Signal Sciences Documentation Archive

Edge Deployment

The Edge deployment method allows you to add the Signal Sciences as an edge security service onto Fastly's Edge Cloud Platform without needing to make any modifications to your own hosting environment.

Limitations and caveats

This feature works with backends defined in VCL services using the API, CLI, or web interface. Backend definitions defined manually in VCL or snippets can be supported by redefining them using the API, CLI, or web interface. This will offer validation and enable a number of features not available to VCL-defined backends, including shielding. Learn more about defining backends in our integration documentation.

We automatically support VCL directors as long as they are defined using the Fastly API. We do not, however, currently support Hash or Client directors. This feature requires a one-time configuration change that needs to be performed by Fastly. Contact sales@fastly.com for more information.

Deploying at the edge

To deploy at the edge, you will need a Signal Sciences corp and at least one site to protect. Setup involves making calls to the Signal Sciences API. These API calls will add privileged dynamic VCL snippets to your service that enable inspection.

Creating the edge security service

Create a new edge security service by calling the edge deployment API endpoint. This API call creates a new edge security service associated with your corp and site. You will need to replace `${corpName}` and `${siteName}` with those of the corp and site you are adding the edge security service to. Your `${corpname}` and `${siteName}` are both present in the address of your Signal Sciences console, such as `https://dashboard.signalsciences.net/corps/${corpName}/sites/${siteName}`.

```
curl -H "x-api-user:${SIGSCI_EMAIL}" -H "x-api-token:${SIGSCI_TOKEN}" \
-H "Content-Type: application/json" -X PUT \
"https://dashboard.signalsciences.net/api/v0/corps/${corpName}/sites/${siteName}/edgeDeployment"
```

Run this API call again for each site you want to deploy on.

Mapping to the Fastly service

To map your corp and site to an existing Fastly service and synchronize the origins, follow these steps:

1. Using the curl command line tool, call the PUT edgeDeployment/${fastlySID} API endpoint in a terminal application:

   ```
   curl -H "x-api-user:${SIGSCI_EMAIL}" -H "x-api-token:${SIGSCI_TOKEN}" \
   -H "Fastly-Key: ${FASTLY_KEY}" -H 'Content-Type: application/json' -X PUT \
   "https://dashboard.signalsciences.net/api/v0/corps/${corpName}/sites/${siteName}/edgeDeployment/${fastlySID}"
   ```

   This API call will create and activate a new service version with dynamic VCL snippets automatically added to the service. By default, the service will be activated and set to 0% traffic ramping. You can override those defaults by providing parameters in the JSON body:

   - `activateVersion` - activate Fastly service version after clone. Possible values are `true` or `false` (unquoted). If not specified, defaults to `true`.
   - `percentEnabled` - percentage of traffic to send to the Next-Gen WAF. Possible values are integers values `0` to `100` (unquoted). If not specified, defaults to `0`. This can be adjusted later. Check out Traffic ramping for details.

   For example, to disable initial activation and set initial traffic ramping to 10%, add the curl parameter `-d '{"activateVersion": false, "percentEnabled": 10}'` to the usual call:

   ```
   curl -H "x-api-user:${SIGSCI_EMAIL}" -H "x-api-token:${SIGSCI_TOKEN}" \
   -H "Fastly-Key: ${FASTLY_KEY}" -H 'Content-Type: application/json' -X PUT \
   -d '{"activateVersion": false, "percentEnabled": 10}' \
   "https://dashboard.signalsciences.net/api/v0/corps/${corpName}/sites/${siteName}/edgeDeployment/${fastlySID}"
   ```

   This API call requires the Fastly-Key header for authentication. The Fastly API key must have write access to the Fastly service ID. This API call will create and activate a new service version with dynamic VCL snippets automatically added to the service.

backends.

To re-assign the Fastly service to another site, follow these steps:

1. Using the curl command line tool, call the DELETE edgeDeployment/${fastlySID} API endpoint in a terminal application:

   ```
   curl -v -H "x-api-user: ${SIGSCI_EMAIL}" -H "x-api-token: ${SIGSCI_TOKEN}" \
   -H "Fastly-Key: ${FASTLY_KEY}" -H 'Content-Type: application/json' -X DELETE \
   "https://dashboard.signalsciences.net/api/v0/corps/${corpName}/sites/${siteName}/edgeDeployment/${fastlySID}"
   ```

   This API call requires Fastly-Key header for authentication. The Fastly API key must have write access to the Fastly service ID. In the Fastly app, a draft version of the service is created. The draft version removes the mapping to the old site.

2. Using the curl command line tool, call the PUT edgeDeployment/${fastlySID} API endpoint in a terminal application:

   ```
   curl -H "x-api-user:${SIGSCI_EMAIL}" -H "x-api-token:${SIGSCI_TOKEN}" \
   -H "Fastly-Key: ${FASTLY_KEY}" -H 'Content-Type: application/json' -X PUT \
   "https://dashboard.signalsciences.net/api/v0/corps/${corpName}/sites/${siteName}/edgeDeployment/${fastlySID}"
   ```

   In the Fastly app, a new `sigsci_config` custom VCL file that maps the corp and site name to the service and synchronizes the origins is added to the existing draft version of the service.

3. Using the curl command line tool, call the PUT edgeDeployment/${fastlySID} API endpoint in a terminal application as above, but with the new `${siteName}`. For example:

   ```
   curl -H "x-api-user:${SIGSCI_EMAIL}" -H "x-api-token:${SIGSCI_TOKEN}" \
   -H "Fastly-Key: ${FASTLY_KEY}" -H 'Content-Type: application/json' -X PUT \
   "https://dashboard.signalsciences.net/api/v0/corps/${corpName}/sites/${siteName}/edgeDeployment/${fastlySID}"
   ```

Synchronizing origins

> **IMPORTANT:** Failure to synchronize origins may result in your traffic not being inspected properly. Requests sent to a backend that does not exist in the edge security service will be served a `503 Unknown wasm backend` error. You can correct this issue by running an API call to properly sync origins after any changes.

Some conditions cause origin syncing to occur automatically:

- Site configuration changes
- Agent mode changes (e.g., blocking, not blocking)
- Enabling or disabling IP Anonymization
- Rule changes (e.g., request rules, signal exclusion rules, CVE rules)
- Rule list changes (only if the list is being used by a rule)
- IP addresses flagged

If you change your origins in the Fastly Console, you will need to take additional action to synchronize your changes using an API call. The API call makes sure origin changes applied in the Fastly Console are reflected in the edge security service. For example:

```
curl -v -H "x-api-user:${SIGSCI_EMAIL}" -H "x-api-token:${SIGSCI_TOKEN}" \
-H "Fastly-Key: $FASTLY_KEY" -H "Content-Type:application/json" -X PUT \ "https://dashboard.signalsciences.net/ap
```

WAF execution

Once both API calls are completed, your service will automatically be set up with dynamic VCL snippets that control the execution of the Next-Gen WAF. A new service version will be created and activated containing the additional VCL snippets.

The edge security service runs in the `vcl_miss` and `vcl_pass` subroutines. Execution priority is set to a high value to enable compatibility with any other VCL snippets that may be in use.

Traffic ramping

You can control the amount of traffic inspected by the edge security service using the `Enabled` dictionary key. This value is available in the `Edge_Security` dictionary and is automatically created when you attach a delivery service.

The default value is 0, with numbers greater than zero representing a percentage of the traffic being inspected. This means that unless you change the value of the `Edge_Security` Edge dictionary, your WAF will be enabled but won't inspect any traffic. If the value is set to 100, all

Note: The `Edge_Security` Edge dictionary no longer uses The `DISABLED` field. To control blocking and logging behavior of an edge security service or turn off agent configurations entirely, use the web interface instead.

Health checks

The edge security service includes a health check inside the `edge_security` function. Using the `backend.health` property, this health check will skip security processing entirely if the edge security service is unhealthy for any reason. The edge security service is modeled as an origin using the backend type and uses the same health check feature.

The health check works by sending a periodic probe every 15 seconds and checks for an HTTP status code 200 as an expected response. Should a check indicate an unhealthy service, all security processing will be skipped until the service becomes healthy again. It may take up to 60 seconds for all security processing to be skipped.

Determining if you already use health check logic

You can check if your service already uses health check logic by inspecting the value of the `x-sigsci-edgemodule` HTTP header, which is added to the request prior to being sent to the edge security service. If the value is greater than or equal to `1.6.0`, then your VCL includes the health check logic.

Enabling and testing health check logic

To enable the health check for an existing service, make sure your VCL is updated to the latest version by re-running the steps in our instructions for mapping to the Fastly service. Then, test the health check logic by toggling the Agent mode to **Off**. This will simulate an unhealthy state for the edge security service and processing will be skipped.

Java Module Overview

The Signal Sciences Java module can be deployed in several ways:

- As a Servlet filter
- As a Jetty handler
- As a Netty handler
- With Dropwizard
- On WebLogic servers

Kubernetes Installation Overview

About Signal Sciences on Kubernetes

We recommend starting with the most common deployment scenario Agent + Module if you are unsure what module to start with. After installing Agent + Module, try out the other options listed below.

Get Started

To start installing Signal Sciences on Kubernetes, choose your deployment option:

- Reverse Proxy
- Agent + Module
- Agent + Ingress Controller + Module
- Envoy
- Istio
- Ambassador
- Pivotal Container Services (PKS)
- AWS Elastic Container Service (ECS)

Upgrading Introduction

- Upgrading an Agent
- Upgrading the NGINX Module
- Upgrading the Apache Module
- Upgrading the IIS Module

Cloud WAF Overview

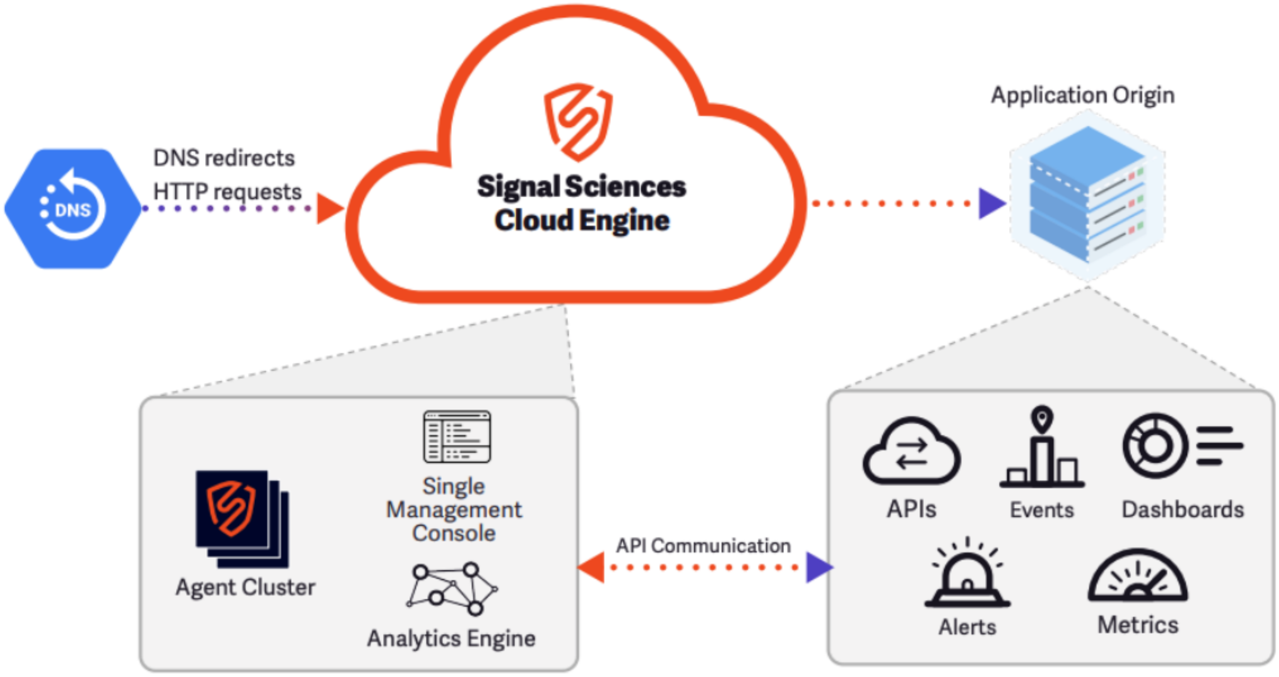
install a Signal Sciences agent and module into their respective environments.

For environments such as these, Cloud WAF is an easily deployable option that provides the same security capabilities of other Signal Sciences agent-based deployment options.

Cloud WAF shares a unified management console with all other deployment options thus providing actionable information and key metrics quickly in a single centralized interface for your entire organization.

How does it work?

Cloud WAF uses the same technology as our other agent-based deployment options under the hood, which means that as a customer, you have full flexibility to deploy wherever your application operates.



For additional information about how the Cloud WAF solution works, see our Cloud WAF product page and data sheet.

Getting started

To deploy a Cloud WAF instance:

1. **Secure communication between Cloud WAF and the client.** Upload a TLS certificate for the domain you are protecting with Cloud WAF to ensure communication between the Cloud WAF instance and the client is encrypted and secure. If your requests will be coming from Fastly's Edge, you can use a Fastly-managed TLS certificate instead and uploading a TLS certificate is optional.

2. **Create the Cloud WAF instance.** Create a new Cloud WAF instance in a geographic location close to the location of your origin. Only Owner users can create and edit Cloud WAF instances. All other user roles will have visibility into your Cloud WAF instances but will not be able to create or edit Cloud WAF instances.

3. **Point DNS to your Cloud WAF instance.** After the DNS change propagates, confirm that Cloud WAF is protecting your applications by viewing the request data populated in the console.

    > **Note:** Ensure that your DNS registrar has the ability to create aliases or CNAME records at the apex (or root) of the domain. If your DNS provider does not support this, we can recommend several DNS providers based on your implementation. Reach out to our support team for more information.

Announcements

Announcing new protection for CVE-2023-34362

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. From the **Rules** menu, select **Templated Rules**.
4. In the search bar, enter `CVE-2023-34362` and then click the **View** link for the CVE-2023-34362 templated rule.
5. Click the **Configure** button.
6. Click the **Add trigger** button, and select the **Block requests from an IP immediately if the CVE-2023-34362 signal is observed** checkbox.
7. Click the **Update rule** button.

Edge deployment now available for the Premier platform

Advanced rate limiting rules and the Site Flagged IP signal have been added for Premier plan customers using edge deployment. Customers who have upgraded from the Professional plan to the Premier plan will now be able to author new site rules that rate limit requests. The Site Flagged IP signal will also be available for use in all types of site rules.

Check out the details about edge deployment and about advanced rate limiting rules to learn more.

Edge deployment setup changes

The setup process for the edge deployment has been changed from custom VCL to dynamic VCL snippets. This change is expected to simplify the onboarding process for all customers using the edge deployment. In particular, if you are using custom VCL for other features of the delivery platform, your setup will become simpler.

This setup change to dynamic VCL snippets applies to new Signal Sciences Corps using the edge deployment and the Edge Deployment documentation has been updated accordingly.

IMPORTANT: Existing edge-deployed sites require a backend configuration change to begin using VCL snippets. Without the configuration change, custom VCL will continue to be used. If you would like to switch to using VCL snippets for your corp and sites reach out to support@fastly.com.

Custom response codes

We've expanded the functionality of our custom response codes feature. Custom response codes allow you to specify the HTTP status code that is returned when a request to your web application is blocked.

Specifically, you can now change the site default blocking response code from `406` to an alternative response code. Blocking actions use the site default blocking response code unless a different response code is specified in a rule.

We also now support the `301` and `302` custom response codes and allow you to specify a redirect URL.

Advanced rate limiting user experience

We've updated the advanced rate limiting user workflow to simplify rate limiting rule configuration. Advanced rate limiting rules put a cap on how often an individual client can send requests that meet set conditions before some or all of the requests from that same client are blocked or logged.

Specifically, the Add form for advanced rate limiting rules has been redesigned. It now includes the **Match type** field. With this field, you can define which requests from the client should be blocked or logged after the threshold has been passed. Options include:

- **Rule conditions:** rate limit requests from the client that match the rule's conditions.
- **Other signal:** rate limit requests from the client that are tagged with a specific signal.
- **All requests:** rate limit all requests from the client.

We've updated and expanded several other areas of the Add form as well. Specifically:

- The **Actions section** now has two subsections: **Tracking** and **Rate Limiting**. In the **Tracking** section, you specify a signal that should be applied to requests that meet the rule's condition set and define the threshold. In the **Rate limiting** section, you define how a client that exceeds the threshold should be rate limited.
- The **Counting signal** field has been renamed to the **Threshold signal**.
- **Action type** menu options have been renamed from **Log request** and **Block signal** to **Log** and **Block**.

Finally, you can use the new `ratelimited` field to search for requests that have been tagged with a specific threshold signal and that have been rate limited.

development workflow. You may continue to use the modules at your own discretion but Fastly will not update or provide technical support for the modules.

Agent management functionality (Beta)

We've expanded our agent management functionality to include:

- a service that automatically updates agent versions.
- a plugin for HashiCorp Vault that stores and rotates agent keys.

When the agent auto-update service is enabled, the service checks the Signal Sciences package downloads site for a new version of the agent and updates the agent when a new version is available. By default, the check for a new version is performed on the second Thursday of the month. The agent auto-update service is only compatible with agents on Debian 8 or higher, Red Hat CentOS 7 or higher, and Ubuntu 18.04 or higher.

With the Signal Sciences plugin for HashiCorp Vault, you can use Vault to store and rotate the Agent Access Key and Agent Secret Key for your site. Vault is an identity-based secrets and encryption management system.

These features are now available for all Signal Sciences customers as part of a beta release.

Professional Plan Edge Deployment Updates

Custom signals, dashboards, lists, templated rules, and custom response codes are now available for Professional plan customers using edge deployment. Customers who have upgraded from the Essential plan to the Professional plan will find that some features now appear in different locations. Specifically:

- Virtual patching (CVE) rules can be found in the Templated Rules menu.
- Threshold functionality can be found in the Rules menu under Site Alerts.

The edge security service includes a health check inside the `edge_security` function. Using the `backend.health` property, this health check will skip security processing entirely if the edge security service is unhealthy for any reason. The edge security service is modeled as an origin using the backend type and uses the same health check feature.

Learn more about the edge deployment by visiting our documentation site.

Announcing New Protection for CVE-2022-42889

A code execution vulnerability affecting the Apache Commons Text library has recently been identified and assigned **CVE-2022-42889**. Fastly has created a virtual patch for it that is now available within your account. To activate it and add protection to your services:

1. Navigate to the Signal Sciences console and select **Templated Rules** from the **Rules** menu.
2. Search the templated rules for `CVE-2022-42889` and then click **View**.
3. Click **Configure** and then click **Add trigger** to configure the rule's thresholds and actions.
4. Select **Block requests from an IP immediately if the CVE-2022-42889 signal is observed** and then click **Update rule**.

LOG4J-JNDI attack signal

SmartParse has been extended to allow for advanced and precise detection of Log4Shell payload attacks with minimal-to-no false positives. SmartParse is our proprietary detection method that analyzes request parameters to determine whether code is actually executable. It requires no manual tuning or configuration because it does not rely on ever-expanding regex pattern matching.

When SmartParse detects Log4Shell attacks, the requests are tagged with the new LOG4J-JNDI attack signal. You should begin seeing requests that match this signal in your requests feed immediately. We've enabled it by default along with default threshold rules. You can adjust these thresholds using site alerts or by creating an instant blocking rule.

To learn more about this new attack signal, check out our blog post.

Agent and module end-of-support plan

Beginning January 31, 2023, agent versions will enter a two year support cycle with versions older than two years being retired or deprecated on a quarterly cadence. Retiring older versions with fewer features will enable us to focus our resources on supporting and developing newer versions that provide more value to our customers. At the end of January, we will support Agent v4.16.0 and above.

Support for our Python and PHP modules will be moving to self-service in March 2023. At that time, you may continue to use the modules at your own discretion, but we will no longer update and provide technical support for the modules. Until the self-service transition occurs, we will fully support both modules. More information about this transition will be posted at a later date.

AWS Lambda Integration is now GA

We've expanded the Fastly Next-Gen WAF (powered by Signal Sciences) capabilities to include protection for serverless and FaaS traffic. Our support for AWS Lambda can help companies grow their web applications without requiring supporting infrastructure and provisioning servers. This support is now generally available to all Signal Sciences and Fastly Next-Gen WAF customers. For more information, see our install guide.

Additionally, we have received the AWS Service Ready designation for our Lambda support, which means our Fastly Next-Gen WAF has gone through full technical evaluations with the AWS team and has tight integration into the AWS marketplace. You can find us listed as an official AWS Lambda partner.

Enabling and disabling logging via the rule builder

We have introduced a new feature in the site and corp rule builder that allows you to select whether to store the logs for requests that match a rule's criteria.

Historically, when creating a custom site or corp rule without a custom signal attached, requests matching the rule's criteria would not be logged. Not logging matching requests reduced the potential noise in your request logs. In cases where you did want to store the logs for matching requests, you had to create a custom signal for the rule.

Now, you no longer have to create a custom signal to log requests that match a rule's criteria. The new Request Logging menu in the site and corp rule builder allows you to select whether matched requests are logged (data storage policy applies).

For new rules, logging is enabled by default, but you can disable logging for a rule by setting the Request Logging menu to **None**. Existing rules that have a custom signal will continue to log matching requests, and existing rules that do *not* have a custom signal will *not* log them.

Announcing the AWS Lambda Integration (Beta)

We're expanding the Fastly Next-Gen WAF (powered by Signal Sciences) capabilities to include protection for serverless and FaaS traffic. We now support AWS Lambda, which is helping companies grow their web applications without having to worry about supporting infrastructure and provisioning servers.

This feature is available now as part of a beta release and will require a configuration change to enable it via feature flag. It is available for all Signal Sciences and Fastly Next-Gen WAF customers, and you can learn how to set it up in our install guide.

Announcing New Protection for CVE-2022-26134

A remote code execution vulnerability affecting the Atlassian Confluence product has recently been discovered and assigned the identifier **CVE-2022-26134**. Fastly has created a virtual patch for it that is now available within your account. To activate it and add protection to your services:

1. Navigate to the Signal Sciences console and select **Templated Rules** from the **Rules** menu.
2. Search the templated rules for `CVE-2022-26134` and then click **View**.
3. Click **Configure** and then click **Add trigger** to configure the rule's thresholds and actions.
4. Select **Block requests from an IP immediately if the CVE-2022-26134 signal is observed** and then click **Update rule**.

Essential Plan Updates

Common Vulnerabilities and Exposures (CVE) signals are now supported for Essential plan customers to help protect you against known exploits and threats. The new functionality can be configured through the web interface from the Signals menu or through the templated rules section of the API.

We've also included a number of enhancements to the edge cloud deployment for the Fastly Next-Gen WAF: new APIs have been added for deprovisioning, origin syncing has been improved, and a percentage ramp up feature is now supported to control the amount of traffic through the edge security service. Learn more about this by visiting our documentation site.

Announcing Fastly Security Labs

We're happy to announce the launch of Fastly Security Labs, a new program that empowers customers to continuously innovate by being the first to test new detection and security features — helping shape the future of security.

Fastly Security Labs provides our customers with an open line of communication directly to the Security Product team and bolsters our feedback loops that bring our customers directly into our product lifecycle process for the Fastly Next-Gen WAF (powered by Signal Sciences), helping us create stronger products. We'll use the program to release a wide range of features from new Signals and Templated Rules to new inspection protocols. You can read more about it in our blog.

we can apply our current set of WAF detections to GraphQL requests, which include protection against OWASP-style attacks. The ability to inspect GraphQL requests means you can apply customs rules to specifically handle those requests. We've also added GraphQL-specific attack and anomaly Signals to address certain targeted attacks. With many common attack vectors at play in GraphQL, we've added new signals out of the box so that any specific routing can be applied to them.

To track GraphQL API requests, your agents must be on version 4.33.0 or above and you need to enable the GraphQL API Query templated rule. GraphQL Inspection is available for all Next-Gen WAF customers. Reach out to your account manager or sales@fastly.com to learn more.

Support for ARM Processors

We're expanding the Fastly Next-Gen WAF (powered by Signal Sciences) capabilities to include more deployment models than ever before. We now support processors using ARM architecture, which are gaining popularity in web applications due to the potential speed gains and overall cost-savings compared to other processors.

The new set of ARM-compatible Agent and Module will sit alongside our existing packages made for other processors, which includes a new ARM Agent and a complementary NGINX-native Module that supports NGINX v1.18.0 and above. It is available for all Signal Sciences / Fastly Next-Gen WAF customers, and you can read more about it in our blog.

Custom Response Codes

We've introduced custom response codes for site rules that block requests. This feature provides you with tighter integration between upstream services and your agents. It is especially powerful for connecting the Fastly edge and the Fastly Next-Gen WAF (powered by Signal Sciences).

You can use this feature to override the default 406 response code from Signal Sciences to enable additional security enforcement in programmable layers. In Fastly, you can use VCL to help you accomplish enhanced enforcement actions such as edge rate limiting or tarpitting.

The feature is available for Professional and Premier platform customers. Learn more about custom response codes by visiting our documentation site.

Renamed - Observed IPs and Rate Limited IPs pages

The Observed IPs page has been renamed to Observed Sources. In addition, the Rate Limited IPs tab has been renamed to Rate Limited Sources. To learn more about Observed Sources, read our announcement or visit our documentation site.

New Identity Provider Integration - Manage users with Okta

We have updated our official Okta integration to support automated provisioning, de-provisioning, and management of users. If you use Okta as your Identity Provider, you can easily install or update the Signal Sciences integration from the Okta Integration Marketplace.

After configuring the integration, any existing Signal Sciences users will be automatically matched to existing Okta users that have identical email accounts.

Customers can use Okta "groups" to assign Signal Sciences roles and site memberships to users in that group.

From Okta, you can:

- Create users in Signal Sciences
- Delete users from Signal Sciences
- Edit users' site memberships
- Edit users' role

Learn more by visiting our official documentation site.

Moved - Rate Limited IPs list

As of February 24, the Rate Limited IPs list, previously available as a tab on the **Events** page (under the **Monitor** menu), is now available on the brand-new **Observed IPs** page (also under **Monitor** menu).

You can also find new Suspicious IP and Flagged IP lists on the Observed IPs page. To learn more about Observed IPs, read our announcement or visit our documentation site.

New Observed IPs page

We've introduced a new Observed IPs page in the Signal Sciences console, found underneath the **Monitor** menu.

Important note: The Rate Limited IPs tab on the Events page has now moved to the Observed IPs page.

Learn more about Observed IPs by visiting our documentation site.

New Dashboards and Templated Rules Page

We are excited to announce today the launch of API and ATO Protection Dashboards, a new set of features dedicated to identifying, blocking, and analyzing malicious behavior that attackers use against web applications and APIs. Now available on the Signal Sciences console, these new dashboards surface security telemetry from over 20 new signals for advanced attack scenarios such as account takeover, credit card validation, and password reset.

For more information, view our blog post about the features.

To configure and activate your new templated rules, login to the management console and select templated rules, or navigate directly to the new dashboards from any site's home dashboard.

New Request Volume Graph

A new Request Volume graph is included in the first position of the default Overview system dashboard on every site. The graph represents the number of requests hitting a site over a given timeframe, along with average RPS. The graph can also be added to any custom dashboard.

To learn more about your site's Overview Page and how to customize dashboards, head over to the relevant docs page.

Deprecated - Weekly Summary Page

The Weekly Summary page is no longer available as of September 9. The summary's information and functionality can now be accessed from site-level dashboards (with the release of the new Request Volume card) Any existing links to the Weekly Summary will be redirected to the site's Overview dashboard with a seven-day look back.

Learn more about dashboards and how to customize them by visiting the relevant docs page.

New Client IP Headers setting

You can now set the real client IP of incoming requests across all agents via the console web interface. The new setting replaces the need to update the `/etc/sigsci/agent.conf` file on each agent to specify the real client IP.

To use the new feature, visit site settings > agent configurations in your console and scroll down to the Client IP Headers section. Learn more

New request to site rule converter

Our latest introduction to the console makes it easier than ever to use data from a request to create a new site rule. To use the tool, click “View request detail” for any request in the requests page, then look for the new “Convert to rule” button. With the new menu, you can select from the available request data to jump-start the process of creating a rule.

API Access Token updates

We've made a number of improvements to API Access Token security, management, and visibility for corp Owners.

Security:

- Corp Owners can set an expiration TTL that applies to all tokens. The expiration countdown is based on the token's creation timestamp.
- Corp Owners can create a list of IP or ranges that all tokens needs to be used from (i.e., a corporate network) otherwise API access will result in a 400-error
- Corp Owners can restrict token usage on a user-by-user basis. See below.
- These restrictions can be enabled or disabled from the **Corp Manage > User Authentication** page

Restrictions by user:

- When per-user restrictions are enabled, globally users cannot create or use tokens unless they are given explicit permission by the corp Owner
- **IMPORTANT:** If users have existing tokens when this feature is enabled, these existing tokens will be disabled (not deleted) until permissions are given to their owners, and then they will resume working. Users just need permission once.
- Permission is granted to users from the **Corp Manage > Corp Users > Edit User** page

Visibility and management:

- Corp Owners can see all the tokens created and in use across the corp from the brand new **Corp Manage > API Access Tokens** page

- When they turn on Restrictions by User, a Corp Owner can use this page to see who needs permission and which tokens are disabled
- Corp Owners can delete access tokens
- An individual user's tokens have moved from their account settings page to the new **My Profile > API Access Tokens** page

New rules conditions

We are pleased to announce the introduction of several new rules conditions that will help give you better visibility into abusive or anomalous behavior on your applications.

- **Response Conditions** Use `Response code` or `Response header` as conditions in request rules or signal exclusion rules for finer detail when adding or removing a signal. Combine response conditions with request conditions to gain greater insight into the results of client requests.
- **Custom Signals** Use custom signals as conditions in request rules to improve workflows or create more complex rule logic.

Learn more

SSO Bypass

A couple updates to the feature formerly known as API Users:

1. We're no longer using the term "API Users" in the console or the API. Instead, these are now "users with SSO Bypass." The intent of this attribute is to enable organizations to invite third-parties to access their SigSci instance (for example, a contractor who is outside the organizations SSO setup). While users with SSO Bypass can still connect to the API, we recommend users create API Access Tokens to connect services or automations to our API.

2. Users with SSO Bypass can now use Two-Factor Authentication (2FA). Corps with SSO enabled can continue to invite users from outside their organization's SSO, like contractors, now with the added protection of 2FA.

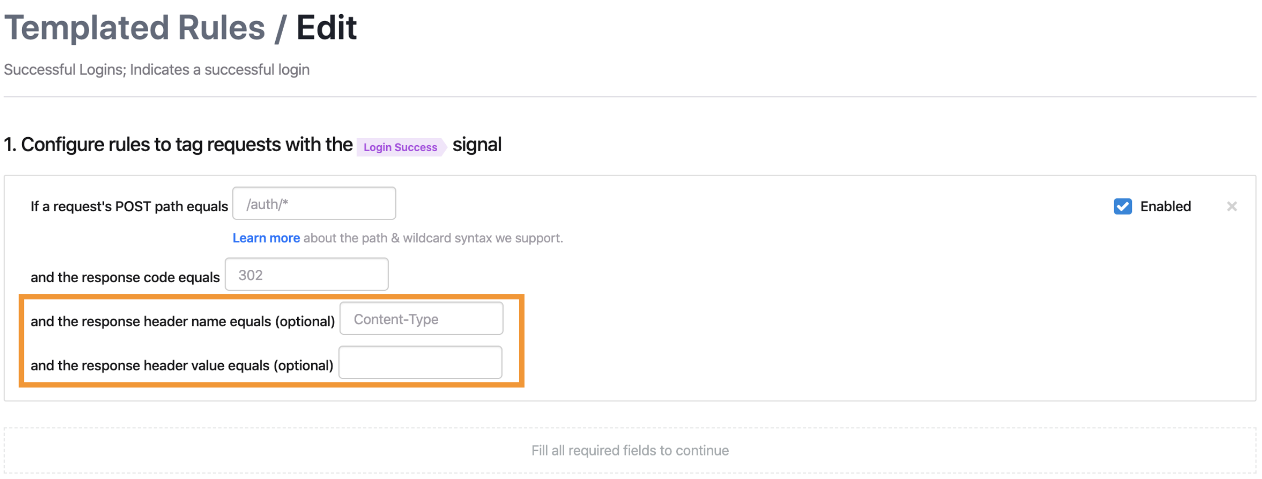
Templated rules response header and value conditions

You can now add optional response header name and value conditions to ATO templated rules, which include:

- `Login Success`
- `Login Failure`
- `Registration Success`
- `Registration Failure`

We're excited to give you these additional levels to protect your apps against ATO and excessive authentication attempts! If you have any questions about these changes, reach out to us at support@signalsciences.com.

Example for the `Login Success` templated rule:



Templated Rules / Edit

Successful Logins; Indicates a successful login

1. Configure rules to tag requests with the Login Success signal

If a request's POST path equals /auth/*

Learn more about the path & wildcard syntax we support.

and the response code equals 302

and the response header name equals (optional) Content-Type

and the response header value equals (optional)

☑ Enabled ×

Fill all required fields to continue

Agent 1x and 2x End-of-Life

We will disable all agents older than 3.0 on March 31, so if you have any agents between 1.x to 2.x please upgrade them before March 31. We've improved our newer agent versions to be much more efficient and secure. If you need help upgrading, let us know at

Multiple custom dashboards

We are excited to announce that we've introduced the ability for users to create and edit **multiple custom dashboards** for each site. Last year, we introduced the ability for users to edit the dashboard found on each site's overview page, by adding custom signal time series graphs and rearranging the layout of those cards. Today, we've introduced the ability to save multiple custom dashboards, each with their own name and card layout. Every card type is moveable, including default cards like the Flagged IPs card. Owners, Admins, and Users can edit and view all of a site's dashboards, and Observers can view them.

Find out more about custom dashboards in our blog post and learn how to create and customize dashboards by visiting our documentation.

Changes to the User API

We've made a few changes to our user roles lately, and we updated the API response for `/api/v0/corps/_/users` to return new values. The new values are already available for use. The old values are still available as well, but they will be deprecated Friday, September 27, 2019.

Old value	New value
`corpOwner`	`owner`
`corpAdmin`	`admin`
`corpUser`	`user`
`corpObserver`	`observer`

Announcing Corp Rules

Take advantage of corp rules in order to create rules that apply to all, or a select number of sites within your corp. In the corp level navigation, simply navigate to Corp Rules > Corp Rules. From this page, manage existing corp rules, or add a new rule with the existing rules builder. Select the global scope to apply the rule to all sites within the corp, or select specific sites that you'd like the rule to apply. Note, this is a corp level feature available to corp owners and admins. For more information on rules look at our documentation

Dashboard navigation changes

We've made some big changes to the dashboard navigation. We've launched a few new features recently, with a focus on elevating some configurations from the site-level to multi-site- or global-level. We wanted to update the nav to make it clearer and more consistent.

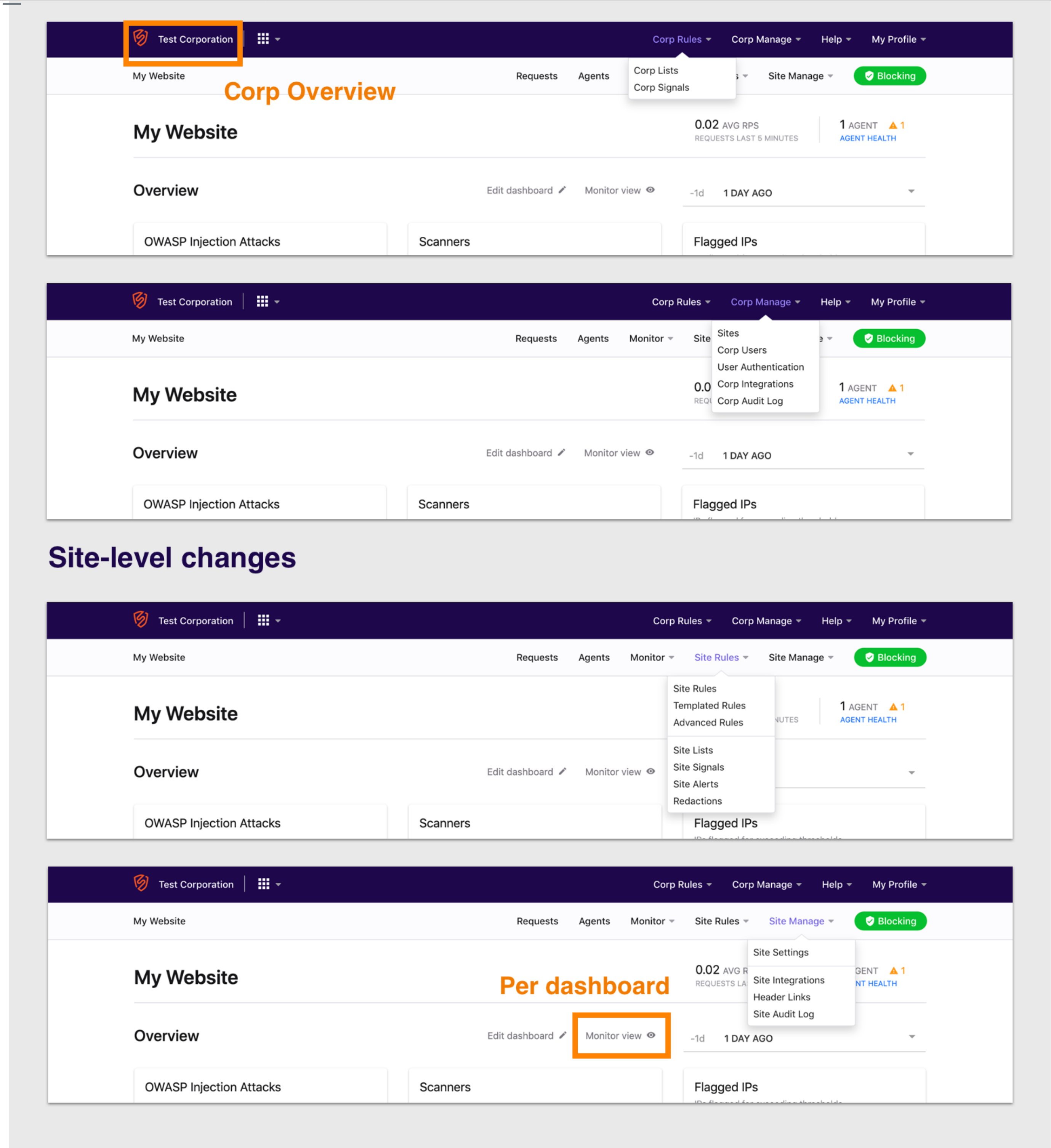
We took a look at making sure each navigation item is in the right menu, and that the menu names are parallel at both the corp- and site-level. Think "Corp Rules" versus "Site Rules." You'll also notice a few items and page names have changed as well. For example, "Activity" is now "Audit log." See a full list of changes below:

Renamed and reorganized categories:

- Library is now "Corp Rules"
- Corp Tools is now "Corp Manage"
- Configure is now split up into "Site Rules" and "Site Manage"
- Corp Rules and Site Rules categories now only contain pages that directly relate to rules.
- We added the words "Corp" and "Site" in front of pages that have a corp/site equivalent to prevent confusion between corp and site levels (e.g., rules, lists, signals, integrations, audit log).
- We removed 2 pages from the navigation to prevent duplicate access points: Corp Overview and Monitor View. Corp Overview was removed since it can be accessed by clicking on your corp name. Monitor View was removed because it can be accessed on the Site Overview page.
- Site Settings is now underneath Site Manage to prevent overcrowding in the nav.
- Site Audit Log (formerly Activity) was moved to Site Manage to stay consistent with Corp Audit Log being underneath Corp Manage

Page nomenclature changes include:

- "Activity" is now "Audit Log"
- "Settings" is now "User Authentication"
- "Week in Review" is now "Weekly Summary"
- "Data Privacy" is now "Redactions"
- "Dashboards" is now "Signals Dashboards"
- "Custom Alerts" is now "Site Alerts"



Event page updates

We have launched some great new improvements to the Events page. Read about the updates below or see them for yourself.

1) We've added filters to the Events page to make it easier to triage and review events. You can filter by IP, signal, and status (Active/Expired).

2) Scrolling and navigation has been improved. First, we've made navigation elements sticky so they follow the user as they scroll up and down the page. Second, we've added a new interaction that automatically scrolls the user to the top of the page when they select a new event, reducing the amount of scrolling you have to do when reviewing multiple events.

3) We also have always-persistent Next Event and Previous Event buttons that make it easy to cycle through and review events. We think this will make it easy to manage the reviewing workflow when there are a lot of events.

Corp Owners and Admins can now assign multiple existing users to a site at once.

Corp Owners and Admins can now assign multiple existing users to a site at once. This provides business unit leaders and site managers an easy way to add their entire team to a new site at once. This feature can be accessed by Owners from the **Corp Users** page (under the **Corp Tools menu**) or by Owners and Admins from the **Site Settings** page. **NOTE:** The flow is restricted to users that are already existing in the corp. New users can't be invited from the flow.

Check out our documentation to learn more.

User Management Updates

The web interface for the corp-level Users Page has been improved to give Owners a better experience when managing and editing users across their entire corp. We've added enhanced filtering so users can now focus on specific sites or roles. This also lays the groundwork for some highly requested user management features.

We have also enhanced the Site Settings Page usability with an easier-to-use tabbed layout. **IMPORTANT:** With this update, the legacy Site Users page has been deprecated and moved to the Users tab.

Announcing Corp Signals

Corp Signals allow you to centrally manage and report on signals that are specific to your business at the corp-level rather than on individual sites! For example, you can create a single corp-level "OAuth Login" signal that can be used in any site rule which will then show up on the Corp Overview page. Learn more.

Stay on top of your corp activity

With corp integrations, you can receive alerts on activity that happens at the corp level of your account. Events relating to authentication, site and user administration, corp rules, and more can be sent to the tools you use for your day-to-day workflow. These are the same events you see in the Corp Activity section of the dashboard.

The following events are available for notification:

- New releases of our agent and module software
- New feature announcements
- Sites created/deleted
- SSO enabled/disabled on your corp
- Corp Lists created/updated/deleted
- Corp Signals created/updated/deleted
- Users invited
- User MFA enabled/updated/disabled
- Users added/removed
- User email bounced
- API access tokens created/updated/deleted

Currently, we offer integrations with Slack, Microsoft Teams, and email. Please visit the Corp Integrations page to configure one today.

Brand new Corp Overview

We have redesigned the Corp Overview page from the ground up to give you better tools to analyze security trends across your entire organization. It has been enhanced to allow you to:

Visualize attack traffic: New request graphs offer a high-level view of traffic across all of your monitored properties, as well as site-by-site breakdowns down of attack traffic and blocked attack traffic.

View corp-level Signal counts: For the first time in the dashboard, you can view the total number of requests tagged with specific Signals across your whole corp using the Signal Trends table. See what security trends are affecting your properties and adjust your security strategy accordingly.

Filter, filter, filter: We've added filtering and pagination tools to just about every aspect of the Corp Overview, allowing you to specify the data you want to see. Filter by site or Signal to zoom in on request data, or use the powerful new time range selector to report day-, week-, or month-over-month.

Visit the Corp Overview page to see for yourself. It can be accessed by clicking on your corp name in the navigation, or by selecting `Corp Tools > Overview`.

tl;dr: Roles and permissions have been updated. Corp Admin is a brand-new role, and existing Corp Owners and Corp Users with multiple site roles experienced some permission updates. Check out the changes below.

What's new?

We've made some changes to our roles and permissions. These changes are designed to make it simpler to manage users across multiple sites at once, and will allow us to introduce some powerful new features in the near future.

Owner has full access and full owner permissions across every site within their corp. This isn't a substantial change; previously Corp Owners could set themselves as members of any and all sites. We're just simplifying the process of granting these permissions.

Admin is a brand new role we created to make it simpler for users to manage multiple sites. The Admin has Site Admin permissions on specific sites, meaning they can invite users and can edit configurations and agent mode (blocking/non-blocking). Admins do not have visibility into sites they do not manage and have limited visibility into corp-level or multi-site features.

User manages specific sites, including configurations and agent mode (blocking/non-blocking). Users do not have visibility into sites they do not manage and have limited visibility into corp-level or multi-site features.

Observer views specific sites in a read-only mode and has limited visibility into corp-level or multi-site features.

Role	Site access	User management privileges	Change agent blocking mode	Configure rules and other settings
Owner	All sites	Invite, edit, delete, security policies	Every site	Every site
Admin	Specific sites	Invite to specific sites	Specific sites	Specific sites
User	Specific sites	No	Specific sites	Specific sites
Observer	Specific sites	No	No	No

How was I affected by the update?

If you were previously a **Corp Owner:** you now have access to every site within your corp and are granted Site Owner permissions by default. Previously, Corp Owners could optionally choose to be members of sites. This option is no longer available.

If you were previously a **Corp User:**

- If you were either a Site Owner or Site Admin on any site in your corp, you are now an **Admin** across all your site memberships.
- If you were a Site User or a Site Observer on sites (and *not* a Site Owner or Site Admin) , you are a **User** on those same sites.
- However, if you only had the Site Observer role across *all* of your site memberships, you are an **Observer** with visibility limited to those same sites.

Questions or concerns? Check out our Customer Support portal.

Updated APT and YUM repository signing keys

Due to a change with our package hosting provider, we have updated the GPG keys for our YUM and APT repositories. Updated GPG URLs are now listed in all relevant installation instructions.

If you have scripts for automated deployment, you will need to update the scripts with the new GPG key URL to ensure they continue to work:

Old URL: `https://yum.signalsciences.net/gpg.key` or `https://apt.signalsciences.net/gpg.key` New URL: `https://yum.signalsciences.net/release/gpgkey` or `https://apt.signalsciences.net/release/gpgkey`

Note: If you're using NGINX 1.9 or earlier, then you will instead want to use the legacy URL of: `https://yum.signalsciences.net/nginx/gpg.key`

Introducing Corp Lists!

Corp Lists are a new feature that allow Corp Owners to manage Lists at the corp-level which can be used by any site-level rule. You can find Corp Lists by going to Library > Corp Lists in the corp-level navigation.

For example, you can centrally manage a list of OFAC-sanctioned countries, or scanner IPs that you may want to block or allow across multiple sites.

Learn more about Lists here.

Customize the Monitor View

Here by popular demand, you can now customize the Monitor View. Previously, the Monitor would display 5-6 default graphs. With the new update, the Monitor now reflects any custom Overview page graphs or arrangements. When displayed as a grid, the Monitor shows the first 6

Custom Signals enable you to gain visibility into traffic that's specific to your application. You can create these signals either on the Custom Signals page (Configure > Custom Signals) or, more commonly, when creating or editing a Rule.

The new Custom Signals page now shows:

1. The number of requests tagged with a particular signal in the past 7 days.
2. The number of Rules that add that signal.
3. The number of Alerts that use that signal.

This additional data makes it easier to determine whether a Custom Signal is working correctly or is no longer used by any Rules or Alerts.

Check out our fresh new status page!

Be sure to subscribe to our new status page at https://www.fastlystatus.com/ so that you can receive alerts in the rare occasion that Signal Sciences has an unexpected event. Please note that you'll need to resubscribe to this new page if you were previously subscribed to the old status page.

Rules Simplification

Starting today, November 8th, we'll be rolling out a new unified Rules page.

Previously Request Rules (rules that allow you block, allow, or tag requests) and Signal Rules (rules that allow you to exclude signals for specific criteria) were managed on two distinct pages. Now Request and Signal Rules can be viewed, managed, and filtered from a single page.

Why are we making this change?

In addition to simplifying the number of pages in the product you need to go to manage rules, this change lays the groundwork for future changes to more easily share rules across sites.

How will this change affect me?

From a user-facing perspective, this change should be minimal — existing URLs will be redirected and you will create and manage rules from a single page.

Where can I learn more about rules?

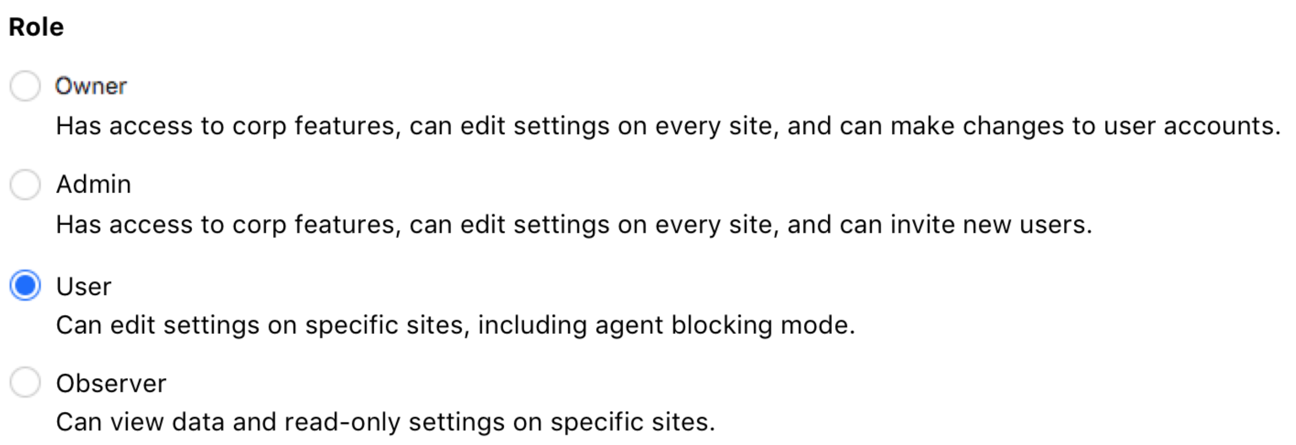
Full documentation for rules is available here.

Coming soon: Updated roles and permissions

tl;dr: Roles and permissions will be changing in January. Corp Admin is a brand-new role, and existing Corp Owners and Corp Users with multiple site roles will experience permission updates. Review the changes below and prepare your organization.

What's new?

We're making some changes to our roles and permissions. These changes are designed to make it simpler to manage users across multiple sites at once, and will allow us to introduce some powerful new features in the near future.



Role

○ Owner
Has access to corp features, can edit settings on every site, and can make changes to user accounts.

○ Admin
Has access to corp features, can edit settings on every site, and can invite new users.

● User
Can edit settings on specific sites, including agent blocking mode.

○ Observer
Can view data and read-only settings on specific sites.

Admin is a brand new role we created to make it simpler for users to manage multiple sites. The Admin has Site Admin permissions on specific sites, meaning they can invite users and can edit configurations and agent mode (blocking/non-blocking). Admins will not have visibility into sites they do not manage and will have limited visibility into corp-level or multi-site features.

User will manage specific sites, including configurations and agent mode (blocking/non-blocking). Users will not have visibility into sites they do not manage and will have limited visibility into corp-level or multi-site features.

Observer will view specific sites in a read-only mode and will have limited visibility into corp-level or multi-site features.

Role	Site access	User management privileges	Change agent blocking mode	Configure rules and other settings
Owner	All sites	Invite, edit, delete, security policies	Every site	Every site
Admin	Specific sites	Invite to specific sites	Specific sites	Specific sites
User	Specific sites	No	Specific sites	Specific sites
Observer	Specific sites	No	No	No

How will I be affected when the roles are updated?

If you are currently a **Corp Owner:** you will have access to every site within your corp and will be granted Site Owner permissions by default. Currently, Corp Owners can optionally choose to be members of sites. This option will no longer be available.

If you are currently a **Corp User:**

- If you are either a Site Owner or Site Admin on any site in your corp, you'll become an **Admin** across all your site memberships.
- If you are a Site User or a Site Observer on sites (and *not* a Site Owner or Site Admin) , you will be a **User** on those same sites.
- However, if you only have the Site Observer role across *all* of your site memberships, you will become an **Observer** with visibility limited to those same sites.

Questions or concerns? Check out our Customer Support portal.

Personal API Access Tokens

Personal API Access Tokens are permanent tokens that can be used instead of passwords to authenticate against the API. This allows SSO and 2FA users to easily access the API without the additional workaround. Furthermore, these tokens can be used directly against API endpoints without having to authenticate and obtain a session token.

Introduction

What is the Signal Sciences architecture?

The Signal Sciences platform is an application security monitoring system that proactively monitors for malicious and anomalous web traffic directed at your web servers. The system is comprised of three key components:

- A web server integration module
- A monitoring agent
- Our cloud-hosted collection and analysis system

The module and agent run on your web servers within your infrastructure, analyzing and acting on malicious traffic in real-time as it is detected. Anomalous request data is collected locally and uploaded to our collectors, allowing us to perform out-of-band analysis of malicious inbound traffic.

Additional details can be found here: Architecture

Installation Process

Getting started with Signal Sciences typically takes less than five minutes and is just a few simple steps depending on your web server (NGINX, Apache).

To get started jump over to our Install Guides

Blocking

Unlike other security products you may have seen before, Signal Sciences' customers actually use our product in blocking mode.

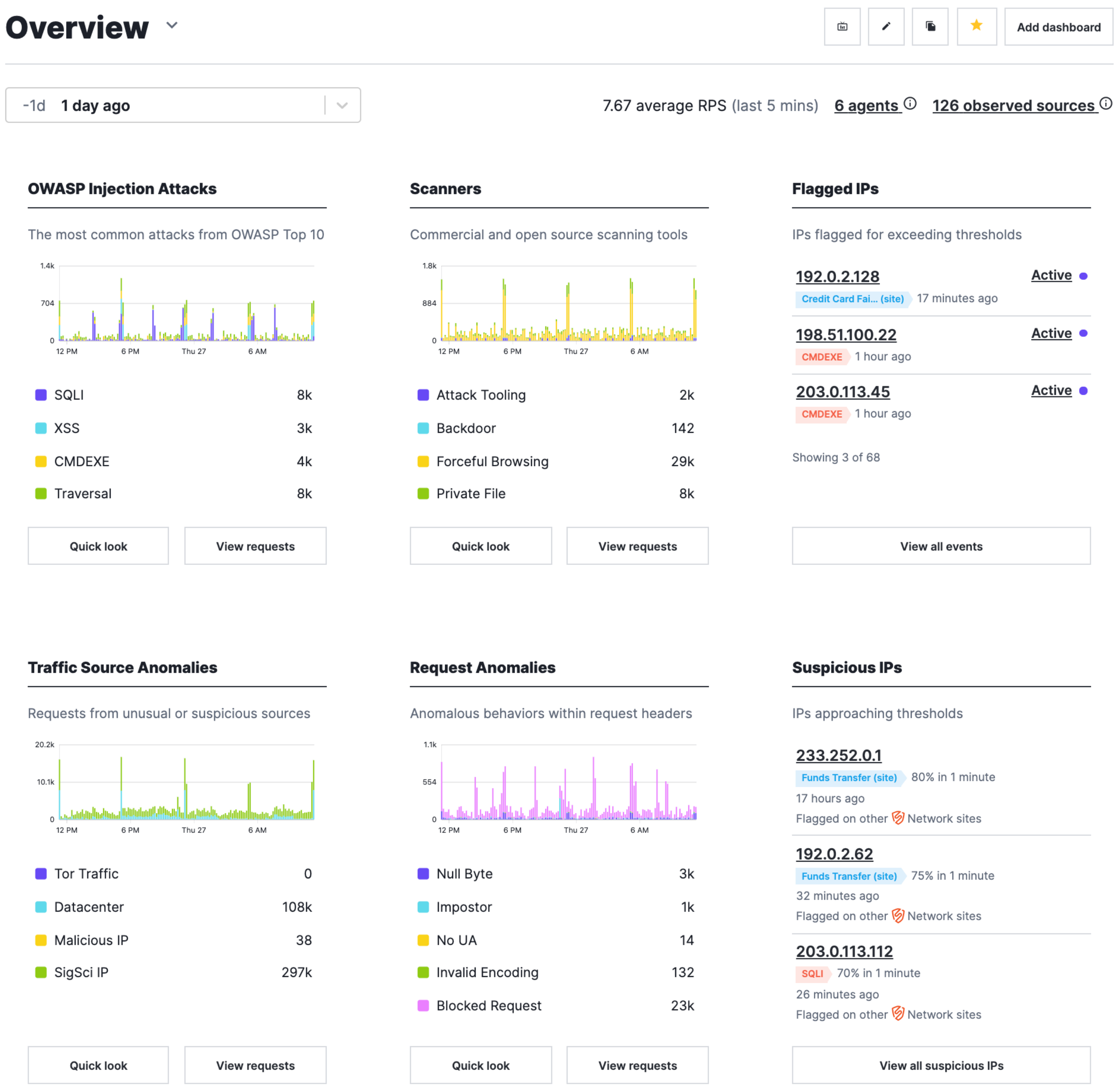
Instead of the legacy approach of blocking any incoming request that matches a regex, Signal Sciences takes an alternative approach by focusing on eliminating attackers' ability to use scripting and tooling. When an incoming request contains an attack, a snippet of that request

Agents will pull these decisions and either log (when the agent mode is set to "not blocking") or block (when set to "blocking") all subsequent requests from that IP that contain attacks.

For more information, see site alerts and agent-modes.

Site Overview page

The Site Overview page gives you an immediate idea about activity for attacks or oddities against the sites that are being managed by Signal Sciences. These include graphs for OWASP Injection Attacks and different types of Anomalies. From any of these graphs you can drill in by clicking requests or highlighting the time period you are interested directly on the graph itself. This page mainly serves as the jumping off point to drill down into more granular detail.



7.67 average RPS (last 5 mins) 6 agents ⓘ 126 observed sources ⓘ

OWASP Injection Attacks

The most common attacks from OWASP Top 10

SQLI	8k
XSS	3k
CMDEXE	4k
Traversal	8k

Scanners

Commercial and open source scanning tools

Attack Tooling	2k
Backdoor	142
Forceful Browsing	29k
Private File	8k

Flagged IPs

IPs flagged for exceeding thresholds

192.0.2.128 Active ●
Credit Card Fai... (site) 17 minutes ago

198.51.100.22 Active ●
CMDEXE 1 hour ago

203.0.113.45 Active ●
CMDEXE 1 hour ago

Showing 3 of 68

Traffic Source Anomalies

Requests from unusual or suspicious sources

Tor Traffic	0
Datacenter	108k
Malicious IP	38
SigSci IP	297k

Request Anomalies

Anomalous behaviors within request headers

Null Byte	3k
Impostor	1k
No UA	14
Invalid Encoding	132
Blocked Request	23k

Suspicious IPs

IPs approaching thresholds

233.252.0.1
Funds Transfer (site) 80% in 1 minute
17 hours ago
Flagged on other Network sites

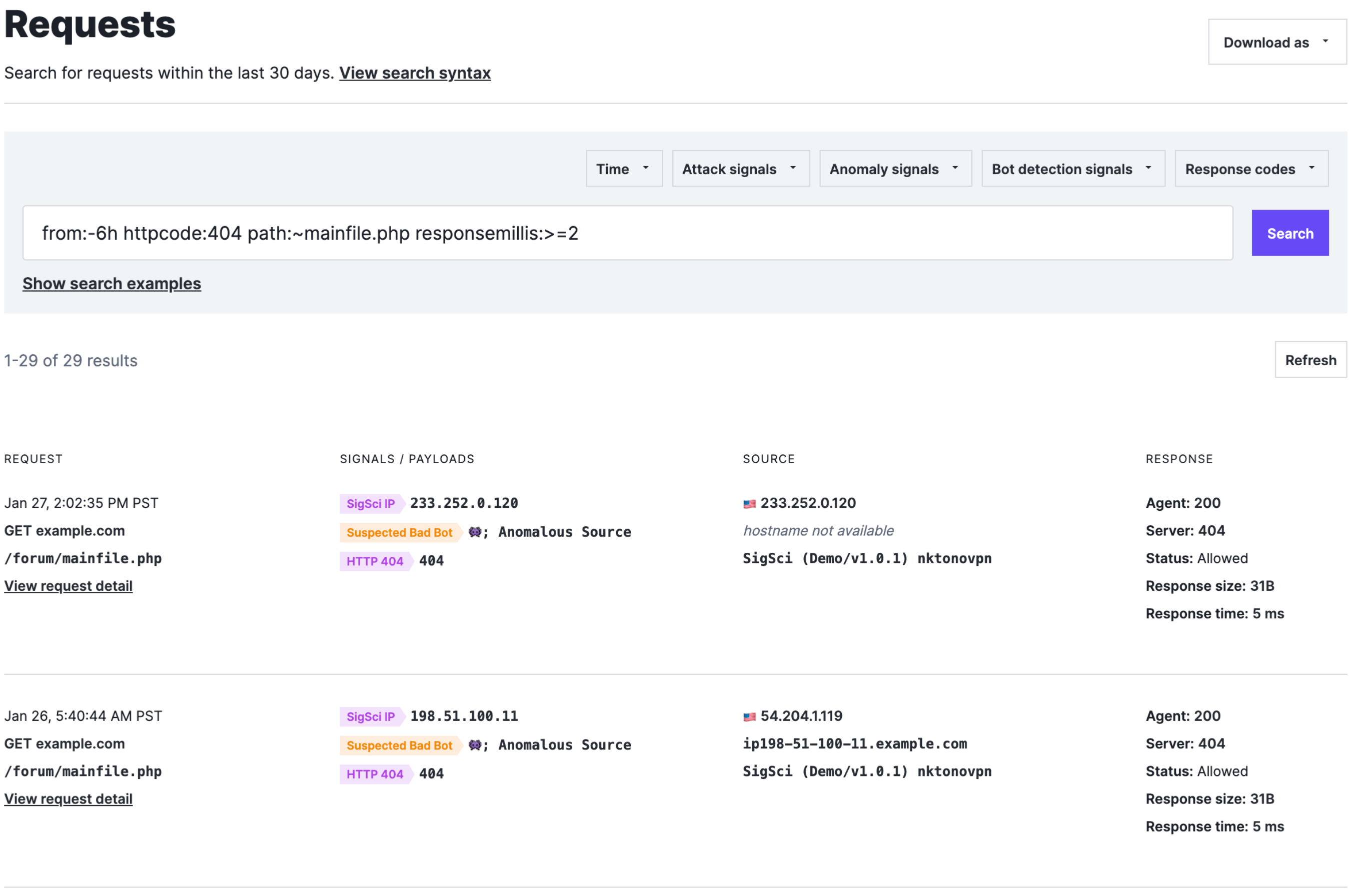
192.0.2.62
Funds Transfer (site) 75% in 1 minute
32 minutes ago
Flagged on other Network sites

203.0.113.112
SQLI 70% in 1 minute
26 minutes ago
Flagged on other Network sites

Requests

The Requests view of Signal Sciences is a very powerful interface for finding information on the different types of requests that are coming through. The requests that are sent to Signal Sciences are going to be either threats or anomalous tagged requests. If you're familiar with the Elastic Search syntax the syntax for Signal Sciences search is very similar. For more advanced search information, see search syntax.

```
from:-6h httpcode:404 path:~mainfile.php responsemillis:>=2
```



Requests

Search for requests within the last 30 days. View search syntax

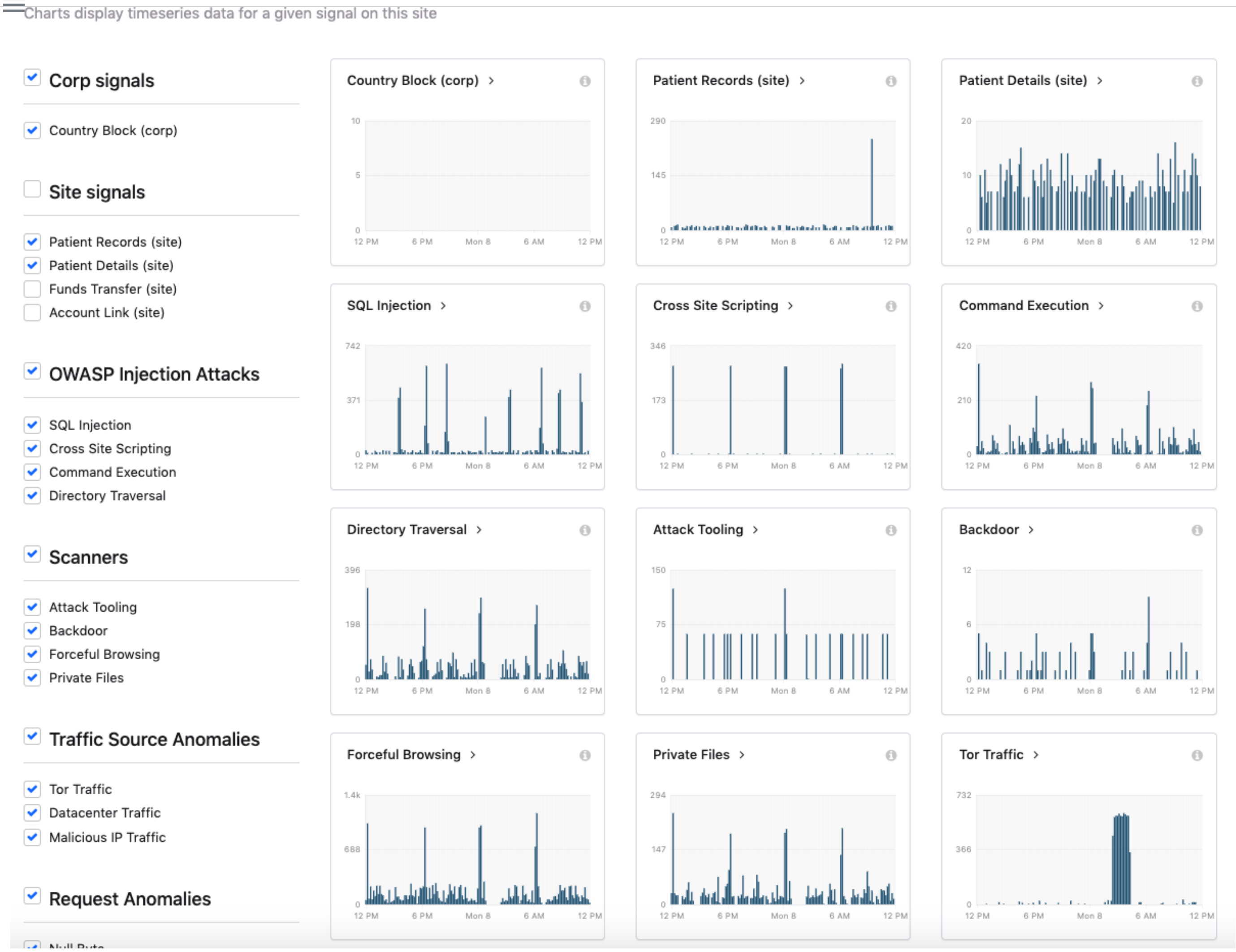
from:-6h httpcode:404 path:~mainfile.php responsemillis:>=2

1-29 of 29 results

REQUEST	SIGNALS / PAYLOADS	SOURCE	RESPONSE
Jan 27, 2:02:35 PM PST GET example.com /forum/mainfile.php View request detail	SigSci IP 233.252.0.120 Suspected Bad Bot 👾; Anomalous Source HTTP 404 404	233.252.0.120 *hostname not available* SigSci (Demo/v1.0.1) nktonovpn	Agent: 200 Server: 404 Status: Allowed Response size: 31B Response time: 5 ms
Jan 26, 5:40:44 AM PST GET example.com /forum/mainfile.php View request detail	SigSci IP 198.51.100.11 Suspected Bad Bot 👾; Anomalous Source HTTP 404 404	54.204.1.119 ip198-51-100-11.example.com SigSci (Demo/v1.0.1) nktonovpn	Agent: 200 Server: 404 Status: Allowed Response size: 31B Response time: 5 ms

Signals Dashboard

In the Signals Dashboard view **Monitor** > **Signals Dashboard** there are breakdowns of the individual signals that are being tracked in your Signal Sciences deployment. There are the out of the box Attacks and Anomalies plus any custom signals that are being tracked. These Dashboards give you a more detailed view into the activity that is happening in your environment.



Charts display timeseries data for a given signal on this site

☑ Corp signals

☑ Country Block (corp)

□ Site signals

☑ Patient Records (site)
☑ Patient Details (site)
□ Funds Transfer (site)
□ Account Link (site)

☑ OWASP Injection Attacks

☑ SQL Injection
☑ Cross Site Scripting
☑ Command Execution
☑ Directory Traversal

☑ Scanners

☑ Attack Tooling
☑ Backdoor
☑ Forceful Browsing
☑ Private Files

☑ Traffic Source Anomalies

☑ Tor Traffic
☑ Datacenter Traffic
☑ Malicious IP Traffic

☑ Request Anomalies

☑ Null Byte

Nginx Module Overview

Choose your NGINX version number followed by your OS to view the correct set of installation instructions. To find your NGINX version run `nginx -v`.

NGINX.org

For NGINX versions 1.14.1 or higher, follow one of these installation instructions:

- Ubuntu
- Red Hat
- Debian
- Amazon Linux
- Alpine Linux

For NGINX versions 1.10.0 - 1.14.1, follow one of these installation instructions:

- Ubuntu
- Red Hat
- Debian
- Amazon Linux

For NGINX versions 1.9 or lower, follow one of these installation instructions:

- Ubuntu
- Red Hat
- Debian

For NGINX Plus releases 17-27, follow one of these installation instructions:

- Ubuntu
- Red Hat
- Debian
- Amazon Linux

Agent Installation Overview

About Agents

The Signal Sciences Agent is a small daemon process which provides the interface between your web server and our analysis platform. An inbound web request is passed to the agent, which then decides whether the request should be permitted to continue, or whether it should take action.

To start installing an agent, choose your OS:

- Ubuntu
- Red Hat
- Debian
- Amazon Linux
- Windows
- Alpine

> **Note:** ARM processors are currently only supported on agent v4.27.0 and higher. Dedicated agent packages are only available for Alpine, Ubuntu, Debian, CentOS, and Red Hat Enterprise.

Agent reliability

The Signal Sciences architecture is intentionally split between the module and the agent in order to optimize performance and improve reliability.

If the agent experiences issues or unavailability, your application will continue to function because the module fails open if it doesn't hear back from the agent within a set time limit. Additionally, agent communication with the Signal Sciences backend is asynchronous, so should the agent lose the ability to communicate with the Signal Sciences cloud backend, the agent will continue to function with the following caveats:

- The agent will continue to detect attacks, anomalies, and any custom rules or signals
- The agent will continue to enforce existing blocking decisions
- The agent will not queue request logs and there will be an outage of data shown in the console. The ability to look at individual requests or aggregate data will be lost until the connection is reestablished.
- The agent will not receive updates for new detections or enforcement decisions

Agent end-of-support policy

Agent versions have a two year support cycle with versions older than two years being retired or deprecated on a quarterly cadence. Retiring older versions with fewer features enables us to focus our resources on supporting and developing newer versions that provide more value to our customers.

Under the agent end-of-support policy, we:

- Support agent versions that are under two years old. Customers who use older versions of the agent are unsupported and don't have access to new features and integrations included with newer versions.

- Deprecate and remove support from agent versions that are older than two years. Unsupported agent versions don't receive rule updates and blocking decisions. Deprecation occurs on a quarterly cadence at the end of January, April, July, and October.

  If you are using an older version, you can upgrade your agent to a supported version. Contact securitysupport@fastly.com if you need help upgrading your agent.

The agent end-of-support policy adheres to the Fastly product lifecycle.

Apache Module Overview

- Ubuntu
- Red Hat
- Debian
- Amazon Linux
- Windows

Installation: Getting Started

Installation Introduction

Signal Sciences supports multiple installation methods. You can use Fastly's Edge Cloud Platform, you can use Signal Sciences' hosted Cloud WAF solution, or you can deploy directly onto your hosting environment via traditional Module-Agent process. Signal Sciences supports traditional, VM-based architectures as well as modern container-based ones. Integrations with several Platforms-as-a-Service (PaaS) are also available. Below are all the installation options available to get Signal Sciences up and running.

Edge Deployment

You can deploy Signal Sciences on Fastly's Edge Cloud Platform by adding it to new or existing Fastly services. Deploying on Fastly's Edge Cloud Platform doesn't require you to install or modify anything on your own hosting environment.

Cloud WAF

Our Cloud WAF solution allows you to deploy Signal Sciences without requiring you to install the Signal Sciences agent and module directly onto your environment.

Module-Agent Installation Process

Signal Sciences can also be deployed directly onto your hosting environment. Getting started deploying Signal Sciences typically takes less than five minutes and is just a few simple steps depending on your web server (NGINX, Apache, etc).

More information about the Signal Sciences Agent and Module can be found in How It Works.

The Signal Sciences installation process is very simple and can be done with three steps:

Step 1: Agent Installation

The Signal Sciences Agent is a small daemon process which provides the interface between your web server and our analysis platform. An inbound web request is passed to the agent, which then decides whether the request should be permitted to continue, or whether it should take action.

Learn how to install an agent

Step 2: Module Installation

The Signal Sciences Module is the architecture component that is responsible for passing request data to the agent. The module deployment is flexible and can exist as a plugin to the web server, a language or framework specific implementation, or can be removed if running the agent in reverse proxy mode.

Learn how to install a module

Step 3: Verify Agent and Module Installation

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. Click **Agents** in the navigation bar near the top of the screen.
4. Check the module version under **Module** to confirm the correct version is listed.

> **Note:** Until there has been at least one request since the agent and module were installed, the module information won't be listed. Once there is traffic the module information will be populated.

Containers and Kubernetes

Signal Sciences supports multiple deployment patterns in Kubernetes. You will likely have to customize configurations for Signal Sciences to work in your own Kubernetes app. The documentation provides several Kubernetes deployment examples, using the Docker sidecar container pattern.

Learn how to install in Kubernetes

following agent-only options are available.

Agent Reverse Proxy Mode

The Agent can be configured to run as a reverse proxy allowing it to interact directly with requests and responses without the need for a module. Running the Agent in reverse proxy mode is ideal when a module for your web service does not yet exist or you do not want to modify your web service configuration - for example, while testing the product. In this mode, the agent sits inline as a service in front of your web service.

Learn how to run the Agent in Reverse Proxy

Envoy Proxy Integration

The Signal Sciences agent can integrate directly with Envoy, a cloud-native reverse proxy, to inspect and protect web traffic. Envoy v1.11.0 or later is recommended, however, Envoy v1.8.0 or later is supported with limited functionality.

Learn how to install Envoy Proxy

Istio Service Mesh Integration

The Signal Sciences agent can integrate with Istio Service Mesh to inspect and protect north/south and east/west traffic in microservices architecture applications. Full Istio integration is only possible in Istio v1.3 or later due to the required extensions to EnvoyFilter introduced in that release.

Learn how to install Istio

AWS Lambda Integration

The Signal Sciences agent can integrate with AWS Lambda. To provide on-demand protection, the agent can be set up to initialize with each function and close out upon function completion.

Learn how to integrate with AWS Lambda

PaaS

The Signal Sciences agent can be easily deployed by Platform as a Service (PaaS). We worked with multiple vendors to integrate our technologies directly into their platforms to simplify deployment.

View PaaS platforms

Using Signal Sciences

Once Signal Sciences is installed, there are no rules or signatures to configure to get immediate visibility and protection against common attack types.

Now that you have Signal Sciences installed, learn how to use Signal Sciences.

Modules Overview

About Modules

Before you begin installing a module, make sure that you've already installed an agent.

The Signal Sciences Module is the architecture component that is responsible for passing request data to the agent. The module deployment is flexible and can exist as a plugin to the web server, a language or framework specific implementation, or can be removed if running the agent in reverse proxy mode.

After you install a module, verify your agent and module installation.

Web Server Module Options

- NGINX Module Install
- Apache Module Install
- IIS Module Install
- HAProxy Module Install
- HAProxy SPOE Module Install
- Kong Plugin Install

Language or Framework Specific Module Options (RASP)

- .Net Module Install
- .Net Core Module Install
- Golang Module Install
- IBM HTTP Server

No Module Option

- Cloud WAF
- Reverse Proxy Mode

Open source modules

The open source modules are as follows:

- PHP module
- Python module

When using an open source module, keep these things in mind:

- The open source modules follow a self-service model. This means that they have a public-only development workflow and that Fastly will not update or provide technical support for the modules.
- Per the MIT license included in each repository, you may use the open source modules without restriction.

PaaS Overview

About Platform as a Service (PaaS)

The Signal Sciences agent can be easily deployed by the PaaS platforms listed below. The installation process is compatible with any of the language buildpacks.

Platforms

- VMware Tanzu
- Heroku
- IBM Cloud
- OpenShift
- Azure App Service
- AWS Lambda

If you prefer to install the agent by OS, refer to the Agent Installation Overview.

Developer Introduction

- API Documentation
- Using Our API
- Terraform Provider
- Extracting Your Data
- Data Flows
- X-SigSci-* Request Headers

FAQ Introduction

Basics

Here are some answers to a few basic Signal Sciences questions.

What platforms does SigSci support for the module/agent?

Our supported platforms are documented on our Compatibility and Requirements page.

If you want to install on another version, OS, or a something new altogether, contact us. Sometimes we can spin up a new version as fast as a day.

Does SigSci provide an API?

Where does Signal Sciences host the Services?

Signal Sciences is hosted across multiple availability zones in Amazon AWS.

What does Signal Sciences need firewall access to?

See Architecture.

What are the default timeouts for the Signal Sciences modules?

When the module receives a request, it sends it to the agent for processing. The module then waits for a decision from the agent (whether or not to block) for a set amount of time before defaulting to allowing the request through. The default timeouts vary by module type and are listed below:

Module	Timeout
Windows IIS	200ms
.NET	200ms
.NET Core	200ms
All other modules	100ms

Account

Here are some answers to a few basic account questions.

How do I add more users?

See User Management.

How do I add a new site?

See Site Management.

How do I install the Signal Sciences module/agent on a new site?

Go to Installation Process and follow the instructions. Any questions? Contact us.

How do I know what version I’m running?

Agent version information can be viewed on the Agents page of the console:

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. Click **Agents** from the navigation bar.

How can I be notified when a new agent or module version is released?

You can subscribe to release notifications through any of the available Corp Integrations. The `releaseCreated` integration event will trigger the integration to notify you when a new agent or module version is released.

Integration Introduction

There are two types of integrations: **Corp Integrations** and **Site Integrations**:

Corp Integrations

Corp integrations notify you about activity within your corp, including changes to users, sites, and settings. Currently only Owners can create and modify Corp Integrations. The following integrations are available as Corp Integrations:

- Mailing List
- Microsoft Teams
- Slack

Note: Corp Integrations are not supported on the Essential platform.

Site Integrations

Site integrations notify you about activity within specific sites, such as IP flagging events, changes to custom rules, and changes to site-level settings. All integrations are available as Site Integrations:

- Cisco Threat Response / SecureX
- Datadog

- Mailing List
- Microsoft Teams
- OpsGenie
- PagerDuty
- Pivotal Tracker
- Slack
- Sumo Logic
- VictorOps

Release Notes Introduction

- Agent
- NGINX
- NGINX C Binary
- Apache
- IIS
- Dotnet
- Dotnet Core
- Java
- Heroku
- IBM Cloud
- Cloud Foundry
- Golang
- Node.js
- HAProxy
- NGINX 1.10 Lua Module
- NGINX 1.11 Lua Module
- NGINX 1.12 Lua Module

Troubleshooting

Apache module fails to load

(*The following information has been confirmed for RHEL/CentOS deployments using the default yum module installation*.)

The default install location for the SigSci Apache module is `/etc/httpd/modules` but some systems may have Apache loading it's config from a non-standard directory. When this happens the `yum` installer will not install `mod_signalsciences.so` to `/etc/httpd/modules` but instead to the following path:

```
/usr/lib64/httpd/modules/mod_signalsciences.so
```

If Apache fails to restart after the module installation because it cannot locate `mod_signalsciences.so` change the LoadModules line in `httpd.conf` to reflect the correct location on the target system.

How do I configure the agent to use a proxy for egress traffic?

The agent can be configured to use a local proxy for egress traffic to the Signal Sciences cloud infrastructure by setting the `HTTPS_PROXY` environment variable. Add the following line to `/etc/default/sigsci-agent`, replacing `IP-OR-HOST-NAME` with the IP address or hostname to proxy traffic to:

```
export HTTPS_PROXY=IP-OR-HOSTNAME
```

Restart the agent and verify the configuration.

How can I view requests that have been blocked or allowed by rules?

When configuring rules with a block or allow action, you can use the Request logging menu to select whether a sample of matched requests are logged or not logged.

When a request is logged, the individual request data and time series data for that request will be available throughout the web interface (e.g., on the Requests page). When a request is not logged, only time series data for that request will be available in the web interface.

two ways:

Command line

Add the `-server-hostname="HOSTNAME"` flag when starting the `sigsci-agent` process via command line:

```
sigsci-agent -server-hostname="HOSTNAME"
```

Config file

Add the following line to your agent configuration file (by default at `/etc/sigsci/agent.conf`):

```
server-hostname = "HOSTNAME"
```

Agent or module is not detected

When the module and agent have been successfully installed you will be able to see them reporting within the Agents page of the console. In many cases, customers first realize there may be a problem with their configuration when they have started the agent and everything appears to be running normally but the agent or module are not listed correctly.

Agent is not detected

Although the agent appears to be running, it's possible for the agent to not be listed in the Agents page of the console. This is typically due to either the agent being misconfigured or a connection issue between the agent and our cloud-hosted backend. Run through the following troubleshooting steps:

1. Check if the agent is running:

   ```
   ps -aef | grep sigsci-agent
   ```

2. Try restarting the agent with:

   ```
   sudo restart sigsci-agent
   ```

3. If the agent is running, ensure communication between the agent and the cloud-hosted backend isn't blocked by your firewall. The Signal Sciences agent communicates with the following endpoints outbound via port 443/TCP:

   - `c.signalsciences.net`
   - `wafconf.signalsciences.net`
   - `sigsci-agent-wafconf.s3.amazonaws.com`
   - `sigsci-agent-wafconf-us-west-2.s3.amazonaws.com`

   Additional information about firewall restrictions can be found in Architecture.

4. Review any log files for error messages:

   ```
   ls -l /var/log/sigsci-agent
   ```

   ```
   tail -n 20 /var/log/sigsci-agent
   ```

5. If the agent is not starting and nothing is written to the log files, check what messages are displayed if you run the agent manually:

   ```
   stop sigsci-agent
   /usr/sbin/sigsci-agent
   ```

6. Run the debug tool and send the output, along with a detailed description of the issue and all log files, to our Support team.

   ```
   /usr/sbin/sigsci-agent-diag
   ```

Module is not detected

Alternatively, although the console may show that the agent is reporting, the module may be listed as "undetected". There are a few possible causes to this scenario and the following steps are intended to help troubleshoot this condition:

1. It is necessary to send a request through the system in order for the module to report to the agent. Generating a manual 404 to the server in question by requesting a page that doesn't exist is the easiest way to start seeing traffic validated on the console. Allow up to 30 seconds from the time of the request for the module to report and the console to display the anomaly.

2. Confirm the steps for module installation specific to your web server, and any optional configuration changes, have been made correctly.

default, the agent and module are configured to use `/var/run/sigsci.sock` as the local domain socket under Linux operating systems and will require sufficient privileges to run properly:

- If using Red Hat/CentOS, check for SELinux:

  ```
  sestatus
  ```

  If SELinux is enabled refer to the SELinux support guide.

- If using Ubuntu check for AppArmor and adjust security profiles if necessary:

  ```
  sudo apparmor_status
  ```

5. If the module is still not reporting, reach out to our Support team with a detailed description of the issue and the following logs:

   - NGINX or Apache `error.log`, IIS error logs (default `%SystemDrive%\inetpub\logs\LogFiles`)
   - If NGINX is your web server, capture the output of:

     ```
     /opt/sigsci/bin/check-nginx
     ```

   - Collect the configuration files `/etc/sigsci/agent.conf` and if running NGINX `/etc/nginx/nginx.conf` or if running Apache your `httpd.conf` normally located in `/etc/httpd/conf/httpd.conf`.

Agent not receiving request data when integrated with Ambassador

The Ambassador configuration may not have AuthService defined, which is required for the Signal Sciences agent to receive request data. AuthService is enabled by default; if the agent is not receiving requests, run `kubectl get authservice` to check on the status of this service.

Why are my F5 load balancer health checks failing when going through the Signal Sciences reverse proxy?

F5 load balancer health checks use HTTP/0.9 by default. However, the SigSci reverse proxy does not support HTTP/0.9 because Go—which the Signal Sciences agent is written in—does not support it. This results in the F5 health checks failing with `400 Bad Request` response codes.

To resolve this, force the F5 health checks to use HTTP/1.0 or HTTP/1.1 instead. Specify the HTTP version in the send string, which will force the monitor to send an HTTP/1.0 or 1.1 request instead.

Below is an example of an HTTP/0.9 GET request:

```
GET /index.html
```

By specifying `HTTP/1.0`, it will instead become an HTTP/1.0 GET request:

```
GET /index.html HTTP/1.0
```

For additional information about altering the F5 health check requests, see F5’s official documentation.

What flags are available for configuring the agent?

The following options were derived from running the command `sigsci-agent -help` and can be used as command line flags, set in `/etc/sigsci/agent.conf` or set as ENV vars.

Refer to our Configuration Options to view all flags.

Generated environment variables:

```
SIGSCI_RPC_ADDRESS
SIGSCI_RPC_VERSION
SIGSCI_ACCESSKEYID
SIGSCI_SECRETACCESSKEY
SIGSCI_MAX_CONNECTIONS
SIGSCI_MAX_BACKLOG
SIGSCI_MAX_PROCS
```

```
SIGSCI_UPLOAD_INTERVAL
SIGSCI_UPLOAD_SEND_EMPTY
SIGSCI_DOWNLOAD_URL
SIGSCI_DOWNLOAD_INTERVAL
SIGSCI_SERVER_HOSTNAME
SIGSCI_CLIENT_IP_HEADER
SIGSCI_REVERSE_PROXY
SIGSCI_REVERSE_PROXY_LISTENER
SIGSCI_REVERSE_PROXY_UPSTREAM
SIGSCI_DEBUG_LISTENER
SIGSCI_DEBUG_RPC_SERIAL
SIGSCI_DEBUG_GC_PERCENT
SIGSCI_DEBUG_DELAY
SIGSCI_DEBUG_ALWAYS_REPLY
SIGSCI_DEBUG_RPC_TEST_HARNESS
SIGSCI_DEBUG_LOG_BLOCKED_REQUESTS
SIGSCI_DEBUG_LOG_RULE_UPDATES
SIGSCI_DEBUG_LOG_WEB_INPUTS
SIGSCI_DEBUG_LOG_WEB_OUTPUTS
SIGSCI_DEBUG_LOG_UPLOADS
SIGSCI_DEBUG_LOG_PROXY_REQUESTS
SIGSCI_DEBUG_LOG_CONNECTION_ERRORS
SIGSCI_DEBUG_LOG_RPC_DATA
SIGSCI_DEBUG_STANDALONE
SIGSCI_DEBUG_LOG_ALL_THE_THINGS
SIGSCI_DEBUG_DISABLE_PROCESSING
SIGSCI_LEGAL
SIGSCI_VERSION
SIGSCI_SITE_KEYS
```

Installing the Java Module as a Servlet Filter

Requirements

- A Servlet 3.x compliant Java servlet container (e.g., Tomcat 7.0.x.+, Jetty 9+, GlassFish 3.0+).

Supported Application Types

The Signal Sciences Java servlet filter module can be deployed to a variety of Servlet 3.0+ Java application servers (e.g., Apache Tomcat, Jetty, Glassfish, Resin).

The module is compatible with application servers deployed on both Linux and Windows servers running the Signal Sciences agent.

Agent Configuration

Like other Signal Sciences modules, the servlet filter supports both Unix domain sockets and TCP sockets for communication with the Signal Sciences Agent. By default, the agent uses Unix domain sockets with the address set to `unix:/var/run/sigsci.sock`. It is possible to override this or specify a TCP socket instead by configuring the `rpc-address` parameter in the Agent.

Additionally, ensure the agent is configured to use the default RPC version: `rpc-version=0`. This can be done by verifying the parameter `rpc-version` is not specified in the agent configuration or if it is specified, ensure that is specified with a value of `0`. Below is an example Agent configuration that overrides the default Unix domain socket value:

```
accesskeyid = "YOUR AGENT ACCESSKEYID"
secretaccesskey = "YOUR AGENT SECRETACCESSKEY"
rpc-address = "127.0.0.1:9999"
```

Download

Download the Signal Sciences Java module manually or access it with Maven.

Access with Maven

```
<repositories>
   <repository>
      <id>sigsci-stable</id>
      <url>https://packages.signalsciences.net/release/maven2</url>
   </repository>
</repositories>

<dependency>
   <groupId>com.signalsciences</groupId>
   <artifactId>sigsci-module-java</artifactId>
   <version>LATEST_MODULE_VERSION</version>
</dependency>
```

Be sure to replace `LATEST_MODULE_VERSION` with the latest release of the Java module. You can find the latest version in our version file at https://dl.signalsciences.net/sigsci-module-java/VERSION.

Download manually

If you aren't using Maven to build or deploy your Java projects, follow these steps to manually download the Signal Sciences Java module:

1. Download the Java module archive from https://dl.signalsciences.net/sigsci-module-java/sigsci-module-java_latest.tar.gz.

2. Extract `sigsci-module-java_latest.tar.gz`.

3. Deploy the jars using one of the following options:

   - Copy `sigsci-module-java-{version}-shaded.jar` (an uber jar with all the dependencies bundled) to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`).
   - Copy `sigsci-module-java-{version}.jar` and its dependencies in the `lib` folder to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`). If you already have any of the dependency jar files in your application classpath folder (i.e., for Tomcat in the `WEB-INF\lib`) then it is not necessary to copy them, even if the version numbers are different. The logging jars are optional based on how `slf4j` is configured.

   Note: If you want coverage across all web applications in your Application Server instance, the jar files must be placed in the server classpath. For example, in Tomcat that would be `%CATALINA_HOME%/lib`.

Installation

1. Update the `web.xml` file of your application with filter and filter-mapping entries.

   The filter supports the use of either Unix domain sockets or TCP sockets for the `rpcServerURI` parameter. Ensure that the value specified here matches the address specified in your Agent configuration. Specify the value using the following formats based on socket type:

   - **TCP Sockets:** `tcp://\<host>:\<port>`
   - **Unix Domain Sockets:** `unix:/\<file path>`

   Add the following lines to your application's deployment descriptor within the existing `<web-app> </web-app>` section.

   Note: If you want coverage across all web applications in your Application Server instance, the filter and filter-mapping entries must be applied to default deployment descriptor for the container. For example, in Tomcat that would be `%CATALINA_HOME%/conf/web.xml`.

```
<web-app>
   <filter>
      <filter-name>SigSciFilter</filter-name>
      <filter-class>com.signalsciences.servlet.filter.SigSciFilter</filter-class>
      <async-supported>true</async-supported>
      <init-param>
         <param-name>rpcServerURI</param-name>
         <param-value>unix:/var/run/sigsci.sock</param-value>
      </init-param>
      <init-param>
```

```
        <init-param>
        <param-name>excludeIpRange</param-name>
          <param-value>192.168.0.1-192.168.0.5,192.169.0.10-192.169.0.12,193.168.0.1,192.168.10.1-192.168.10
        </init-param>
        <init-param>
          <param-name>excludeCidrBlock</param-name>
          <param-value>192.168.14.0/24,193.165.0.0/28,192.168.11.0/24</param-value>
        </init-param>
        <init-param>
          <param-name>excludePath</param-name>
          <param-value>/test/exit,/hello,/bonus</param-value>
        </init-param>
        <init-param>
          <param-name>excludeHost</param-name>
          <param-value>localhost,127.0.0.2</param-value>
        </init-param>
      </filter>
      <filter-mapping>
        <filter-name>SigSciFilter</filter-name>
        <url-pattern>/*</url-pattern>
      </filter-mapping>
    </web-app>
```

2. Restart the Application Server.

Module Configuration

Option	Default	Description
`rpcServerURI`	Required, tcp://127.0.0.1:9999	The Unix domain socket or TCP connection to communicate with the agent.
`rpcTimeout`	Required, 300ms	The timeout in milliseconds that the RPC client waits for a response back from the agent.
`maxResponseTime`	Optional, no default	The maximum time in seconds that the server response time will be evaluated against (i.e. to see if it exceeds this value) to determine if the module should send a post request to the agent.
`maxResponseSize`	Optional, no default	The maximum size in bytes that the server response size will be evaluated against (i.e. to see if it exceeds this value) to determine if the module should send a post request to the agent.
`maxPost`	Optional, no default	The maximum POST body size in bytes that can be sent to the Signal Sciences agent. For any POST body size exceeding this limit, the module will not send the request to the agent for detection.
`asyncStartFix`	Optional, false	This can be set to `true` to workaround missing request body when handling requests asynchronously in servlets.
`altResponseCodes`	Optional, no default	Space separated alternative agent response codes used to block the request in addition to 406. For example "403 429 503".
`excludeCidrBlock`	Optional, no default	A comma-delimited list of CIDR blocks or specific IP addresses to be excluded from filter processing.
`excludeIpRange`	Optional, no default	A comma-delimited list of IP ranges or specific IP addresses to be excluded from filter processing.
`excludePath`	Optional, no default	A comma-delimited list of paths to be excluded from filter processing. If the URL starts with the specified value it will be excluded. Matching is case-insensitive.
`excludeHost`	Optional, no default	A comma-delimited list of host names to be excluded from filter processing. Matching is case-insensitive.

Sample module configuration:

Module configuration changes must be made in the `<!-- Signal Sciences Filter -->` section of your application's `web.xml` file:

```
<!-- Signal Sciences Filter -->
<filter>
    <filter-name>sigSciFilter</filter-name>
    <filter-class>com.signalsciences.servlet.filter.SigSciFilter</filter-class>
```

```
      <param-value>500</param-value>
  </init-param>
    <init-param>
      <param-name>asyncStartFix</param-name>
      <param-value>true</param-value>
  </init-param>
  </filter>
  <filter-mapping>
       <filter-name>sigSciFilter</filter-name>
       <url-pattern>/*</url-pattern>
  </filter-mapping>
  <!-- end Signal Sciences Filter -->
```

IIS Module Install

Requirements

- Windows Server 2008R2 (Windows 7) or higher (64-bit)
- IIS 7 or higher
- Verify you have installed the Signal Sciences Windows Agent. This will ensure the appropriate folder structure is in place on your file system.

Before you begin

- We only support 64-bit and 32-bit application pools on Windows 2012 or higher. We only support 64-bit application pools on Windows Server 2008R2.
- We only support 64-bit OSes. For older or 32-bit versions of Windows, it is possible to deploy the Signal Sciences Agent as a reverse proxy. If you have questions or require assistance with older or 32-bit versions of Windows, reach out to our support team.
- IIS Module v2.0 and higher includes the utility `sigscictl.exe` which outputs diagnostic information. The information provided by this utility is useful for troubleshooting issues and checks, among other things, whether or not 32-bit app pools are enabled on your server.

Download

The latest version of the IIS module can be downloaded as an MSI installer or a legacy ZIP archive from https://dl.signalsciences.net/?prefix=sigsci-module-iis/.

Alternatively, the IIS module is also downloadable via Nuget.

Installation

The IIS Module is available as an MSI installer or as a legacy ZIP archive. The install packages contain a DLL that must be configured as an IIS native module and a configuration schema that must be registered with IIS. This configuration and registration with IIS is done automatically by the MSI package, or must be done manually if using the legacy ZIP archive.

Install using the MSI

Double-click (or right-click and select install) the MSI file to install it.

Alternatively, for unattended installation, use the following command. This command will not display any output, but will install into `%PROGRAMFILES%\Signal Sciences\IIS Module` by default. It will also register the Signal Sciences module and configuration with IIS:

> **Note:** You may be prompted for Administrator credentials if the login session is not already running as an Administrator.

```
msiexec /qn /i sigsci-module-iis_latest.msi
```

If you require an alternative install location, specify it with the `INSTALLDIR=path` option to the `msiexec.exe` command above. For example:

```
msiexec /qn /i sigsci-module-iis_latest.msi INSTALLDIR=D:\Program Files\Signal Sciences\IIS Module
```

Legacy install using the ZIP archive

> **Note:** This method may not be supported in the future. It is recommended to install via MSI even if you previously used the ZIP archive.

1. Extract the ZIP archive contents to the IIS Module install directory (`C:\Program Files\Signal Sciences\IIS Module`).

```
cd "%PROGRAMFILES%\Signal Sciences\IIS Module"
.\SigsciCtl.exe Install
```

If you need to install into an alternative location, then you will need to run the `Register-Module -file DLL-path`, `Register-Config -file XML-path` and optional `Configure-Module` commands with the `SigsciCtl.exe` utility (see `SigsciCtl.exe Help` for more information). Ensure the `SigSciIISModule.dll` is not located under the `C:\Users\` directory or its sub-directories. For security, Windows prevents DLL files from being loaded from any location under `C:\Users\`.

Verify installation

To confirm the module DLL has been registered with IIS, run the following from a terminal running as Administrator to verify the SignalSciences module is listed:

```
"%PROGRAMFILES%\Signal Sciences\IIS Module\SigsciCtl.exe" Get-Modules
```

The output should look similar to the following:

```
IIS Global Modules:

                            Name Image                                                                    Precond:
-------------------------------- ------------------------------------------------------------------------ -------
               HttpLoggingModule %windir%\System32\inetsrv\loghttp.dll
                  UriCacheModule %windir%\System32\inetsrv\cachuri.dll
                 FileCacheModule %windir%\System32\inetsrv\cachfile.dll
                TokenCacheModule %windir%\System32\inetsrv\cachtokn.dll
                 HttpCacheModule %windir%\System32\inetsrv\cachhttp.dll
         StaticCompressionModule %windir%\System32\inetsrv\compstat.dll
           DefaultDocumentModule %windir%\System32\inetsrv\defdoc.dll
          DirectoryListingModule %windir%\System32\inetsrv\dirlist.dll
           ProtocolSupportModule %windir%\System32\inetsrv\protsup.dll
                StaticFileModule %windir%\System32\inetsrv\static.dll
   AnonymousAuthenticationModule %windir%\System32\inetsrv\authanon.dll
          RequestFilteringModule %windir%\System32\inetsrv\modrqflt.dll
               CustomErrorModule %windir%\System32\inetsrv\custerr.dll
 ApplicationInitializationModule %windir%\System32\inetsrv\warmup.dll
                  SignalSciences C:\Program Files\Signal Sciences\IIS Module\SigsciIISModule.dll          bitness
```

To confirm that the module configuration has been registered, run the following from a terminal running as Administrator to output the current configuration:

```
"%PROGRAMFILES%\Signal Sciences\IIS Module\SigsciCtl.exe" Get-Configs
```

The output should look similar to the following but may also list sites individually:

```
C:\WINDOWS\system32\inetsrv\config\schema:

Date                         Size Name
-------------------- ------------ --------------------------------
2020-02-13 03:12:56Z          677 SignalSciences_schema.xml

"SignalSciences" Configuration Section (Global):

                       Attribute Value
-------------------------------- ----------------------------------------------------------------
                       agentHost
                       agentPort 737
                  statusPagePath
                           Debug False
                ReuseConnections False
                     MaxPostSize 100000
                     AnomalySize 524288
```

Full diagnostics information can be displayed with the following command:

```
"%PROGRAMFILES%\Signal Sciences\IIS Module\SigsciCtl.exe" Info
```

Configure

Configuration changes are typically not necessary. By default, the module will use port 737 to communicate with the agent (or in v2.0.0+, if the agent was configured to use an alternate port, it will use that port). The configuration can be set via the MSI installer, the new `SigsciCtl.exe` utility in v2.0.0+, IIS Manager UI, via PowerShell, or using the `appcmd.exe` utility.

> **Note:** Ensure that the same port number is used by the both the module and the agent configurations.

Using the MSI

To set a configuration option when installing the MSI, specify the option on the command line in `option=value` format. For example:

```
msiexec /qn /i sigsci-module-iis_latest.msi agentHost=203.0.113.182 agentPort=737
```

Using SigsciCtl.exe

To set a configuration option via `SigsciCtl.exe` utility after install, use the `Configure-Module` command. For example:

```
"%PROGRAMFILES%\Signal Sciences\IIS Module\SigsciCtl.exe" Configure-Module agentHost=203.0.113.182 agentPort=737
```

To view the active configuration via the `SigsciCtl.exe` utility the `Get-Configs` command:

```
"%PROGRAMFILES%\Signal Sciences\IIS Module\SigsciCtl.exe" Get-Configs
```

This should output something similar to the following:

```
C:\WINDOWS\system32\inetsrv\config\schema:

Date                         Size Name
-------------------- ------------ --------------------------------
2020-02-13 03:12:56Z          677 SignalSciences_schema.xml

"SignalSciences" Configuration Section (Global):

                        Attribute Value
-------------------------------- ----------------------------------------------------------------
                        agentHost
                        agentPort 737
                  statusPagePath
                           Debug False
                  ReuseConnections False
                     MaxPostSize 100000
                     AnomalySize 524288
            AnomalyDurationMillis 1000
                     TimeoutMillis 200
```

Using PowerShell

To set a configuration option via PowerShell (modern Windows only) use the `-SectionPath "SignalSciences"` option such as follows:

```
Set-IISConfigAttributeValue -ConfigElement (Get-IISConfigSection -SectionPath "SignalSciences") -AttributeName "a
```

To list the configuration using PowerShell, run the following:

```
(Get-IISConfigSection -SectionPath "SignalSciences").RawAttributes
```

To reset the configuration to defaults using PowerShell, run the following:

```
Clear-WebConfiguration -Filter SignalSciences -PSPath 'IIS:\'
```

Using the appcmd.exe

To set a configuration option via the `appcmd.exe` command line tool use the `-section:SignalSciences` option. For example:

```
"%SYSTEMROOT%\system32\inetsrv\appcmd.exe" list config -section:SignalSciences
```

To reset the configuration to defaults using `appcmd.exe`, run the following:

```
"%SYSTEMROOT%\system32\inetsrv\appcmd.exe" clear config -section:SignalSciences
```

Uninstall

1. Open a terminal running as Administrator.
2. Run the following in the terminal:

   ```
   cd "\%PROGRAMFILES%\Signal Sciences\IIS Module"
   .\SigsciCtl.exe Uninstall
   ```

Upgrade

To upgrade the IIS module, you will need to download and install the latest version of the module and verify the configuration is still valid.

If you previously used the ZIP archive to install, then it is recommended that you upgrade via the MSI package. The MSI v1.10.0 or later can be installed over top of an older ZIP file installation following the instructions above.

Cloud WAF Certificate Management

Before you begin

Before uploading your TLS/SSL certificate, ensure that your private key is not password protected and your certificate information is PEM formatted. Any number of certificates can be uploaded, but no more than 48 unique certificates can be applied to a single Cloud WAF instance.

Viewing certificates and their details

To view a summary of all TLS certificates protecting your site with Cloud WAF:

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. From the **Corp Manage** menu, select **Cloud WAF Certificates**. The Certificates page for your site's Cloud WAF appears displaying a summary table that lists the name, domains, status, and expiration details for all certificates at your site.

To view more specific details about a particular TLS certificate, follow the steps above and then click the **View** button at the right of a specific site in the summary table.

Adding certificates

> **Note:** If TLS connections terminate at the Edge before requests are sent to Cloud WAF, then uploading a TLS certificate is optional. Always upload and use certificates if traffic is direct to the Cloud WAF using HTTPS.

To add a certificate, upload it by following the steps below:

1. On the **Certificates** page, click **Add certificate**. A page where you can add certificate details appears.
2. Fill out the certificate details as follows:
   - In the **Name** field, enter a meaningful name that can help you manage the certificate and distinguish it from any others that may exist.
   - In the **Certificate body** field, enter the body of the unencrypted, PEM-formatted server certificate provided by your certification authority. RSA 2048 and 4096 certificates can be used.
   - In the **Certificate chain** field, enter the certificate chain, which is also known as the *intermediate certificate*. The certificate chain is not required for self-signed certificates.
   - In the **Private key** field, enter your certificate's private key.
3. Click the **Upload certificate** button. The newly uploaded certificate appears on the Certificates page in the summary table.

After uploading your certificate, be sure to create a Cloud WAF instance to protect your origin. Keep in mind that, for requests coming from Fastly's Edge, you can use a Fastly-managed TLS certificate instead when you create a Cloud WAF instance. In this case, uploading a TLS certificate is optional.

Deleting a certificate

2. Click **Remove certificate** in the upper-right corner of the page.

Signal Sciences Agent Container Image

The official `signalsciences/sigsci-agent` container image is available from the Signal Sciences account on Docker Hub.

You can pull this image with `signalsciences/sigsci-agent:latest` (or replace `latest` with a version tag).

If you need to modify this image or want to build it locally, then follow the instructions below.

Custom sigsci-agent Dockerfile

You can build on top of the existing `sigsci-agent` container image using `FROM`. However, some care needs to be taken as the Dockerfile is set up to run commands as the `sigsci` user instead of `root`. If you use the recommended Dockerfile, then you may need to change to the `root` user, then back to the `sigsci` user after any system modifications are done.

Example: Installing an Additional Package

```
dockerfile
# Start from the official sigsci-agent container
FROM signalsciences/sigsci-agent:latest

# Change to root to install a package
USER root
RUN apk --no-cache add mypackage

# Change back to the sigsci user at the end for runtime
USER sigsci
```

Build the Signal Sciences agent Docker container image

The recommended `sigsci-agent` Dockerfile is included in the `sigsci-agent` distribution `.tar.gz` archive.

To build the image, download and unpack this archive and follow the instructions in the README.md included in the archive.

The following example commands:

- Download the `sigsci-agent_latest.tar.gz` archive.
- Unpack the archive into a `./sigsci-agent` directory.
- Build the image tagged with `signalsciences/sigsci-agent:latest` and `signalsciences/sigsci-agent:<version>`.

```
curl -O https://dl.signalsciences.net/sigsci-agent/sigsci-agent_latest.tar.gz
mkdir sigsci-agent && tar zxvf sigsci-agent_latest.tar.gz -C sigsci-agent
cd sigsci-agent
make docker
```

You can use a custom name for the tags by setting `IMAGE_NAME` (e.g., `make IMAGE_NAME=custom-prefix/sigsci-agent docker`).

To build manually, run the following command, replacing `YOUR-TAG` and `YOUR-VERSION`:

```
docker build . -t your-tag:your-version
```

Ubuntu NGINX 1.14.1+

Add the package repositories

Add the version of the Ubuntu package repository that you want to use.

Ubuntu 22.04 - jammy

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
```

Ubuntu 20.04 - focal

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ focal main" | sudo tee /etc/apt/sources.list.d/sigs
```

Ubuntu 18.04 - bionic

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ bionic main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 16.04 - xenial

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ xenial main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 14.04 - trusty

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ trusty main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 12.04 - precise

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ precise main" | sudo tee /etc/apt/sources.list.d/si
```

Install the NGINX module

1. Install the Signal Sciences NGINX module by running the following command, replacing “`NN.NN`” with your NGINX version number:

   ```
   sudo apt-get install nginx-module-sigsci-nxo=1.NN.NN*
   ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the `pid /run/nginx.pid;` line:

   ```
   load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
   ```

3. Restart the NGINX service to initialize the new module.

   ```
   sudo service nginx restart
   ```

Console

My data is not showing in the console but the agent and module are running

If both the agent and module are reporting as active within the console, but no data is displayed when requests are processed, then the system time on the agent is likely out of sync. This can cause events to be reported at times significantly in the past or future. This is especially likely in a dev environment using a VM or container that gets in a paused state and is not updated via cron.

To determine whether this condition is occurring:

1. Click **Agents** in the navigation bar. The agents page appears.
2. Click on the name of the agent. The agent metrics page appears.
3. Inspect the graph for **Agent clock skew (seconds)**. The agent clock skew should not be more than a few seconds. If this is a large value updating the system time and maintaining ntpd should rectify the issue.

Requests in the console aren’t reporting any signals

Confirm your OS and web server are supported

See supported versions to confirm what OS and web server versions are supported.

2. In the **Status** column, confirm the agent is listed as online.
3. In the **Module** column, confirm the module is listed as detected.
4. Click on the name of the agent. The agent metrics page appears.
5. Review the listed agent metrics to confirm the console is receiving telemetry from the agent. If the console is not receiving telemetry from the agent, some metrics will be listed as `Unknown` or `0 ms`.
6. Confirm agent clock skew.

Check NGINX

If NGINX is your web server confirm NGINX, the agent, and the module are configured correctly by running

```
/opt/sigsci/bin/check-nginx
```

Contact Support

If you have confirmed any issues with the previous steps, please gather any necessary data and reach out to our Support team for assistance.

1. Enable verbose debug logging by adding the following line to your agent configuration file (by default at `/etc/sigsci/agent.conf`):

   ```
   debug-log-all-the-things = true
   ```

2. Restart the agent and collect the verbose log entries.

3. Generate an agent diagnostic package by running

   ```
   sigsci-agent-diag
   ```

4. Collect the agent configuration file located by default at `/etc/sigsci/agent.conf`.

5. Collect server configuration files:

   - NGINX: `/etc/nginx/nginx.conf`
   - Apache: `/etc/httpd/conf/httpd.conf`
   - IIS: `%SystemDrive%\System32\inetsrv\config\applicationHost.config`
6. Collect server error log files (if applicable):

   - NGINX: `/var/log/nginx/error`
   - Apache: `/var/log/apache2/error.log`
   - IIS: `%SystemDrive%\inetpub\logs\LogFiles`
7. If NGINX is your web server, collect the output of:

   ```
   /opt/sigsci/bin/check-nginx
   ```

8. Reach out to our Support team with a detailed description of the issue and all collected logs and configuration files.

Why am I seeing target hosts in the console for domains I do not own?

This can happen if the requester is using a modified hosts file or forged host header. This is done to make it appear as though the target is a foreign host when it has actually been configured to point to one of your IP addresses directly.

How do I report on the right most X-Forwarded-For IP address?

When multiple IP addresses are appended to the `X-Forwarded-For` header, by default the console reports on the left-most IP address. In some situations (e.g., users of Amazon ELB) you may want to report on the right-most IP address instead. To report on the right-most IP address, make sure you are running the latest version of the Signal Sciences module and agent and then follow the instructions for configuring the X-Forwarded-For header.

Ubuntu Agent Installation

This guide explains how to install the Signal Sciences agent on Ubuntu.

Prerequisites

Before you begin, determine the version of Ubuntu you want to use.

Add the package repository

To add the Ubuntu 22.04 - Jammy package, run the following script:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/ubuntu/ jammy ma
sudo apt-get update
```

Ubuntu 20.04 - Focal

To add the Ubuntu 20.04 - Focal package, run the following script:

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ focal main" | sudo tee /etc/apt/sources.list.d/sigs
```

Ubuntu 18.04 - Bionic

To add the Ubuntu 18.04 - Bionic package, run the following script:

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ bionic main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 16.04 - Xenial

To add the Ubuntu 16.04 - Xenial package, run the following script:

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ xenial main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 14.04 - Trusty

To add the Ubuntu 14.04 - Trusty package, run the following script:

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ trusty main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 12.04 - Precise

To add the 12.04 - Precise package, run the following script:

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ precise main" | sudo tee /etc/apt/sources.list.d/si
```

Install and configure the Signal Sciences Agent package

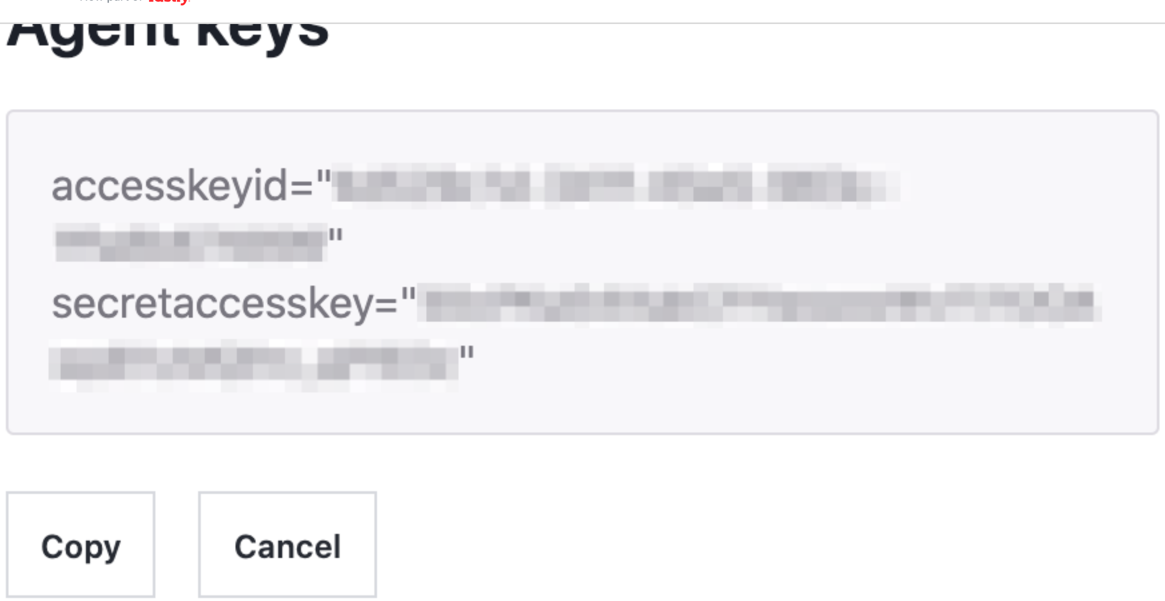
Now that you’ve downloaded the Ubuntu package repository, you can install the Signal Sciences Agent package.

Run the following command to install the Signal Sciences Agent package.

```
sudo apt-get install sigsci-agent
```

Once the agent package is installed, you must create an agent configuration file and add the **Agent Access Key** and **Agent Secret Key**:

1. Create an empty agent configuration file in the following directory: `/etc/sigsci/agent.conf`.
2. Log in to the Signal Sciences console.
3. From the **Sites** menu, select the site that you want to give the agent access to.
4. Click the **Agents** link in the site navigation bar. The agents page appears.
5. Click the **View agent keys** button. The agent keys window appears.



Agent Keys

```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

7. Navigate to the agent configuration file and paste the **Agent Access Key** and **Agent Secret Key** into the file.

   ```
   accesskeyid = "AGENTACCESSKEYHERE"
   secretaccesskey = "AGENTSECRETACCESSKEYHERE"
   ```

8. Save the agent configuration file.

Start the Signal Sciences Agent

Now that you've installed and configured the agent package, you can start the Signal Sciences agent.

For Ubuntu versions 18.04 and above, run the following command to start the Signal Sciences agent:

```
sudo service sigsci-agent start
```

For Ubuntu versions 15.04 through 17.10, run the following command to start the Signal Sciences agent:

```
sudo systemctl start sigsci-agent
```

For Ubuntu versions 14.04 and below, run the following command to start the Signal Sciences agent:

```
sudo start sigsci-agent
```

Optionally, enable the agent auto-update service. The service checks the Signal Sciences package downloads site for a new version of the agent and updates the agent when a new version is available.

Next Steps

Explore our module options and install the Signal Sciences module.

Ubuntu Apache Module Install

1. Install the Signal Sciences Apache module.

   ```
   sudo apt-get install sigsci-module-apache
   ```

2. Add the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the **Dynamic Shared Object (DSO) Support** section to enable the Signal Sciences Apache module:

   ```
   LoadModule signalsciences_module /usr/lib/apache2/modules/mod_signalsciences.so
   ```

3. Restart the Apache web service.

   ```
   sudo service apache2 restart
   ```

Next Steps

Using Our API

Our entire console is built API-first — this means that anything we can do, you can do as well via our API, which is fully documented here.

We've seen customers use our API a number of ways, but a common use case is importing our request data into a SIEM like Splunk or Kibana which can allow you to more easily correlate our security data with your internal data.

About API Access Tokens

Users can connect to the API by creating and using personal API Access Tokens. Authenticate against our API using your email and access token.

By default, all users have the ability to create and use API Access Tokens. However, Owners can choose to restrict API Access Token creation and usage to specific users. All plans allow you to create up to 5 access tokens per user.

Managing API Access Tokens

Follow these steps when managing API access tokens.

Creating API Access Tokens

1. From the **My Profile** menu, select **API Access Tokens**. The API Access Tokens menu page appears.
2. Click the **Add API access token** button. The Add API Access Tokens menu page appears.
3. In the **Token name** field, enter a name to identify the access token.
4. Click the **Create API access token** button. The new token appears.
5. Record the token in a secure location for your use.

    > **Note:** This is the only time the token will be visible. Record the token and keep it secure. For your security, it will not appear in the console.

6. Click the **Continue** button to finish creating the token.

Restricting User Permission to Create and Use API Access Tokens

Owners can restrict all users from creating and using API Access Tokens. After doing so, Owners can then manually grant specific users permission to create and use API Access Tokens.

API Access Tokens that were created before restrictions were activated will not be deleted. However, the users with existing tokens will need to be given permission to use API Access Tokens. Until a user is again granted permission to use API Access Tokens, the token will remain in a disabled state. After a user has been granted permission, the console will remember that permission moving forward.

Owners can enable API Access Token restrictions by following these steps:

1. From the **Corp Manage** menu, select **User Authentication**. The User Authentication menu page appears.
2. Navigate to the **API Access Tokens** section.
3. In the **Access token permissions** field, select the **Restrict access by user** option.
4. A message will be displayed warning you about this setting and its restrictions. Click the **Continue** button to proceed.
5. Click the **Update API Access Tokens** button to save this change.

Granting Users Permission to Create and Use API Access Tokens

When API Access Token creation and usage is restricted, only Owners can enable other users to create API Access tokens.

> **Note:** After restricting API Access Token usage, Owners will also need to grant themselves permission to create and use API Access Tokens.

1. From the **Corp Manage** menu, select **Corp Users**. The Corp Users menu page appears.
2. Click on the user you want to grant permission to.
3. Click the **Edit corp user** button.
4. Under the **Authentication** section, select the **Allow this user to create API Access Tokens** checkbox.

1. From the **My Profile** menu, select **API Access Tokens**. The API Access Tokens menu page appears.
2. Click the **Delete** link to the right of the token you want to delete. The Delete API Access Token menu page appears.
3. Click the **Delete** button to confirm you want to delete the token.

Viewing Personal API Tokens

Owners can view a table of all access tokens across your corp by going to the **Corp Manage** menu and selecting **API Access Tokens**. This table shows the various statuses of each token (active, expired, disabled by owner), their creators, IPs they were used by, and expiration dates.

Managing Corporation-Wide API Access Token Settings

Follow these steps when managing corporation-wide API access token settings.

Setting Automatic Token Expirations

Owners can set API Access Tokens to automatically expire after a set period of time.

1. From the **Corp Manage** menu, select **User Authentication**. The User Authentication menu page appears.
2. Navigate to the **API Access Tokens** section.
3. In the **Access token expiration**, select the **Custom expiration** option. The custom expiration menu appears.
4. Select one of the default periods of time, or select **Custom** to set a specific custom period of time.

   The expiration is based on the creation date of the token itself, not from the start of the expiration policy. For example if there's a 60-day-old token and you set a 30-day expiration policy, the token will instantly be expired. But if you later switch the expiration to 90 days, the token will be un-expired.
5. Click the **Update API Access Tokens** button.

Restricting API Access Token Usage by IP

Owners can restrict the use of API Access Tokens to specific IP addresses.

1. From the **Corp Manage** menu, select **User Authentication**. The User Authentication menu page appears.
2. Navigate to the **API Access Tokens** section.
3. In the **Restrict usage by IP (optional)** field, enter the IP addresses and IP ranges you want to limit token usage to. Enter each IP address on a new line.
4. Click the **Update API Access Tokens** button.

Using Personal API Access Tokens

Golang

```
package main

import (
        "encoding/json"
        "fmt"
        "io/ioutil"
        "log"
        "net/http"
        "os"
        "time"
)

var (
        // Defines the API endpoint
        endpoint = "https://dashboard.signalsciences.net/api/v0"
        email    = os.Getenv("SIGSCI_EMAIL")
```

```
// Corp is a Signal Sciences corp
type Corp struct {
        Name         string
        DisplayName  string
        SmallIconURI string
        Created      time.Time
        SiteLimit    int
        Sites        struct {
                URI string
        }
        AuthType    string
        MFAEncorced bool
}

// CorpResponse is the response from the Signal Sciences API
// containing the corp data.
type CorpResponse struct {
        Data []Corp
}

func main() {
        // No need for timestamps or anything
        log.SetFlags(0)

        // Get corps
        req, err := http.NewRequest("GET", endpoint+"/corps", nil)
        if err != nil {
                log.Fatal(err)
        }

        // Set headers
        req.Header.Set("x-api-user", email)
        req.Header.Set("x-api-token", token)
        req.Header.Set("Content-Type", "application/json")
        req.Header.Add("User-Agent", "SigSci Go-Example")

        // Make request
        var transport http.RoundTripper = &http.Transport{}
        response, err := transport.RoundTrip(req)
        if err != nil {
                log.Fatal(fmt.Sprintf("Error connecting to API: %v", err))
        }
        defer response.Body.Close()

        payload, err := ioutil.ReadAll(response.Body)
        if err != nil {
                log.Fatal(fmt.Sprintf("Unable to read API response: %v", err))
        }

        if response.StatusCode != http.StatusOK {
                log.Fatal(fmt.Sprintf("API request failed, status: %d, resp: %s", response.StatusCode, payload))
        }

        var corpResp CorpResponse
        err = json.Unmarshal(payload, &corpResp)
        if err != nil {
                log.Fatal(err)
        }
}
```

Python

```python
import requests, os

# Initial setup

endpoint = 'https://dashboard.signalsciences.net/api/v0'
email = os.environ.get('SIGSCI_EMAIL')
token = os.environ.get('SIGSCI_TOKEN')

# Fetch list of corps

headers = {
        'Content-type': 'application/json',
        'x-api-user': email,
        'x-api-token': token
}
corps = requests.get(endpoint + '/corps', headers=headers)
print corps.text
```

Ruby

```ruby
require 'net/http'
require 'json'

# Initial setup

endpoint = "https://dashboard.signalsciences.net/api/v0"
email = ENV['SIGSCI_EMAIL']
token = ENV['SIGSCI_TOKEN']

# Fetch list of corps

corps_uri = URI(endpoint + "/corps")

http = Net::HTTP.new(corps_uri.host, corps_uri.port)
http.use_ssl = true

request = Net::HTTP::Get.new(corps_uri.request_uri)
request["x-api-user"] = email
request["x-api-token"] = token
request["Content-Type"] = "application/json"

response = http.request(request)
puts response.body
```

Shell

```
curl -H "x-api-user:$SIGSCI_EMAIL" -H "x-api-token:$ACCESS_TOKEN" -H "Content-Type: application/json" https://das
```

Architecture

What is the Signal Sciences architecture?

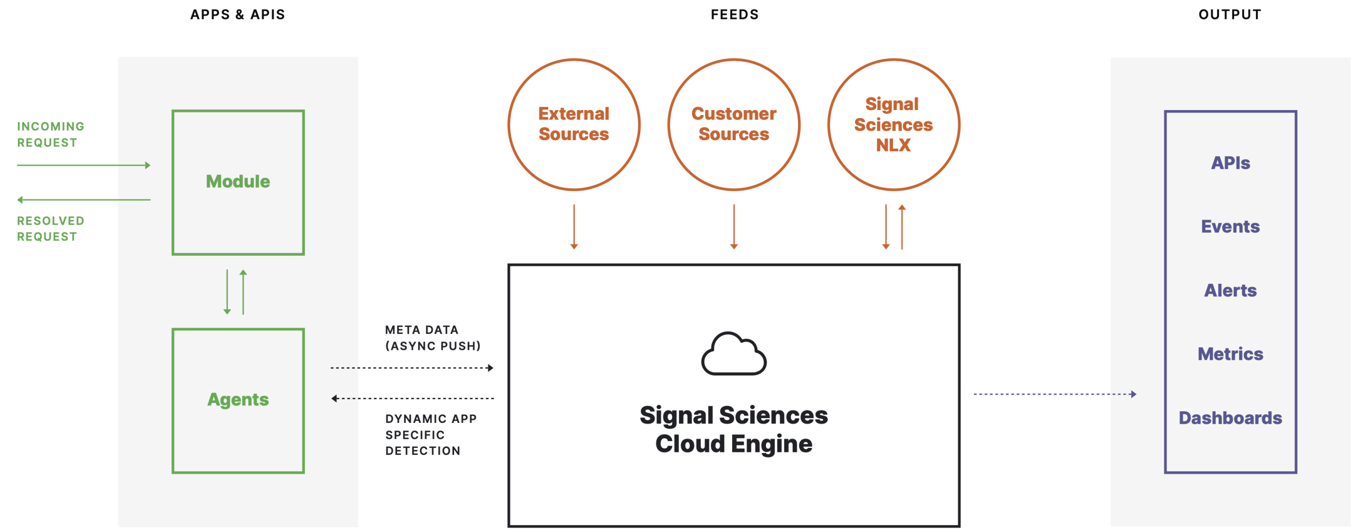
The Signal Sciences platform is an application security monitoring system that proactively monitors for malicious and anomalous web traffic directed at your web servers. The system is comprised of three key components:

- A web server integration module
- A monitoring agent
- Our cloud-hosted collection and analysis system

with that decision. The module can exist as a plugin to the web server or a language-specific implementation.

The agent decides whether to block, allow, or tag requests. When it receives a request from the module, it runs through the rules set up and decides how the request should be handled. The agent then relays the request and its decision back to the module. The agent is also responsible for relaying with the cloud-hosted collection and analysis system; uploading processed request data and downloading new rules and configurations set up in the console.

The cloud-hosted collection and analysis system receives data from the agent and other sources. This includes request data, IP address information, and agent/module performance metrics. This information is then exported and made visible in the Signal Sciences console, through the API, and any third-party integrations you have set up.



What language is the agent written in?

The agent is written in Go. We chose Go because of its combination of performance, ease of deployment, and memory safety guarantees. In other words, it gets very close to native code performance, without the security issues associated with C/C++ (e.g., buffer overflows).

Where is it typically deployed?

Our software is typically installed directly on your web server. It can also be deployed on a reverse proxy or load balancer running Apache/NGINX. Another less common but technically viable approach is to deploy our software at the application layer. We currently provide modules for Node.js, Java, and .NET and can supply documentation to help you write an application layer module in other languages.

Where are you hosting the service?

We are hosting the service in AWS West across multiple availability zones.

What does Signal Sciences need firewall access to?

To download and install Signal Sciences, you will need to ensure your firewall allows access to the following:

- `apt.signalsciences.net`
- `yum.signalsciences.net`
- `dl.signalsciences.net`

The Signal Sciences agent communicates with the following endpoints outbound via port 443/TCP:

- `c.signalsciences.net`
- `wafconf.signalsciences.net`
- `sigsci-agent-wafconf.s3.amazonaws.com`
- `sigsci-agent-wafconf-us-west-2.s3.amazonaws.com`

If the agent is unable to download from the Fastly CDN, it will fall back to downloading directly from an S3 bucket with an additional fallback to a secondary bucket in a second region until it can download from the CDN or primary S3 bucket again.

> **Note:** Because the Signal Sciences endpoints are hosted on AWS, the IP addresses are dynamic with no set ranges. Because there are no set IP ranges, you will need to ensure firewall access via DNS.

50 of the Alexa Traffic Rankings.

Do you support configuration management?

Yes, we support Chef, Puppet, Ansible, and others. It's easy to manage typical deployments with configuration management tools.

Do you support CDNs?

Yes, we can consume the `X-Forwarded-For` or any other header to obtain the true client IP address.

Do you support egress HTTP proxies?

Yes, instructions for configuring the Signal Sciences agent to use a proxy for egress traffic can be found here.

Do you have an API?

Yes, we have a fully documented, RESTful/JSON API so you can pull your Signal Sciences console data into your other systems.

Do you support integrations with SIEMs?

Yes, we support any SIEM via our API.

Cloud WAF Instance Management

Before you begin

To save time before creating a Cloud WAF instance, ensure you have uploaded a TLS certificate. If requests will be coming from Fastly's Edge, you can use a Fastly-managed TLS certificate instead by disabling uploaded certificates.

Viewing Cloud WAF instances

Cloud WAF instances are created and managed directly in the Signal Sciences console. To view an instance:

1. Log in to the Signal Sciences console.
2. From the **Corp Manage** menu, select **Cloud WAF Instances**. The Cloud WAF Instances page appears.

The Cloud WAF Instances page provides a summary table that lists all Cloud WAF instances running on your corp, including names, regions, and statuses. You can view additional details about each Cloud WAF instance by clicking the **View** button to the right of the summary table. Of particular note when viewing these additional details are the DNS entry and Health Check details.

Using health checks

Health checks can be used to assess whether or not the Cloud WAF, or a particular route within the Cloud WAF instance, is up or down. The checks can be used within Fastly or other systems to achieve a redirect failover. There are two methods available for accessing health check endpoints:

- View the details of your Cloud WAF instance and click the **Copy** button to the right of the **Health Check** field. This URL is specific to your Cloud WAF instance and you can use it make health check HTTPS requests.
- Make HTTPS requests to the `/sigsci-healthcheck` path of the fully qualified domain name used in a route for your Cloud WAF instance. For example, if one of your routes uses the domain name `example.com`, you could make a health check request to `https://example.com/sigsci-healthcheck`.

Creating a Cloud WAF instance

Cloud WAF instances contain basic server configuration details and workspace details about the site that those instances will be deployed on. Workspace details specifically include routes information for the paths that requests take from clients to upstream origins.

To create a Cloud WAF instance, follow these steps:

1. On the Cloud WAF instance list menu page, click **Add Cloud WAF Instance**. The Cloud WAF instance creation menu page appears.
2. In the **Server configs** area, supply the following information:
   - In the **Name** field, enter a name for the Cloud WAF instance.
   - In the **Description** field, enter a description for the Cloud WAF instance to make identifying and managing the instance easier.
   - From the **Region** menu, select the geographic region in which the Cloud WAF instance will be deployed. To minimize latency, select the region geographically closest to the location of your origin. The region can't be changed after the Cloud WAF instance is provisioned.
   - From the **Min TLS version** menu, select the minimum TLS version your Cloud WAF instance will use. The minimum TLS version pertains to requests from the client to the Cloud WAF instance. If a request is received with a TLS version lower than the selected

edge, you can optionally set this to **disabled** to use a Fastly-owned certificate instead.

3. In the **Workspaces** section, enter the following information:
    - From the **Site** menu, select the Signal Sciences site on which to deploy the Cloud WAF instance.
    - From the **Instance location** controls, select **Direct** if the Cloud WAF instance will send traffic directly to the upstream origin. In this mode, the source IP address is read from the `X-Forwarded-For` header by default. If the Cloud WAF instances will send traffic to a CDN in the path of the upstream origin, select **Advanced** instead and enter a value for the **Client IP header**.
    - From the **Pass-through protocol** controls, select **HTTPS only** to only allow requests sent over HTTPS through to your origin or select **HTTP and HTTPS** to allow requests sent over either HTTP or HTTPS through to your origin.
4. In the **Routes** section of the **Workspaces** area, enter the following information:
    - In the **Request** field, enter the fully qualified domain name of the property that you'd like to protect with Cloud WAF (e.g., `example.com`). You may include subdomains and paths. The wildcard asterisk (*) can be used to match an entire single path segment between two forward slashes but cannot be used to match partial strings. For example, `www.example.com/foo/*/bar` is valid, but `www.example.com/foo/foo*/bar` is invalid.
    - In the **Origin** field, enter the origin address of the domain name entered in the Request field. Include the protocol (e.g., `https://`) as the first part of the origin address even if you're providing an IP address.
    - From the **Certificates to deploy** menu, select a TLS certificate associated with the request URI. If the appropriate certificate doesn't appear in the list, add it by clicking **Add certificate** and filling out the fields of the window that appears. If you disabled certificate uploads in the **Server configs** area, this section won't be configurable.
    - Leave the **Pass host header** switch disabled if using Server Name Indication (SNI). Enable this setting for the agent to pass the host header to the upstream origin to be used in the TLS handshake. The host header value will take precedence over set values for the host.
    - Leave the **Connection pooling** switch enabled to allow open TCP connections to the origin to be reused. Disable this setting if open TCP connections should not be reused.
    - Leave the **Trust proxy headers** switch disabled to have an agent ignore and drop incoming proxy headers. Enable this setting to allow the agent to trust incoming proxy headers (such as the `X-Forwarded-For` header).
5. Decide whether or not to add more routes to this site. To add another route to this site, click **Add route** and an additional **Routes** section will appear that you can fill out by repeating the above steps.
6. Decide whether or not to add an additional site for this Cloud WAF instance. To add a route to a different Signal Sciences site, click **Add workspace** and an additional **Workspaces** area will appear that you can fill out by repeating the above steps.
7. Click **Create instance** to create the Cloud WAF instance. The Cloud WAF Instances page appears with the new Cloud WAF instance listed with a status of `In progress`. Wait a few minutes for the Cloud WAF instance to be deployed, at which point the status will change to "Deployed".
8. Click **View** to the right of the Cloud WAF instance. The details page for that Cloud WAF instance will appear.
9. Make note of the DNS entry and the egress IP addresses listed. You'll need this information to create a CNAME record for the DNS entry with your DNS registrar. If your origin is not accessible to the public internet, you will also need to configure your origin to allow access from the egress IP addresses provided.

Editing a Cloud WAF instance

1. On the Cloud WAF Instances page, click **View** to the right of the Cloud WAF instance. The details page for that Cloud WAF instance appears.
2. Click **Edit Cloud WAF Instance**. The Cloud WAF instance configuration page appears.
3. Make any changes necessary to the Cloud WAF instance.
4. Click **Update instance**.

Deleting a Cloud WAF instance

1. On the Cloud WAF instance list menu page, click **View** to the right of the Cloud WAF instance. The details page for that Cloud WAF instance appears.
2. Click **Remove Cloud WAF Instance**.
3. Click **Delete**.

Installing the Java Module as a Jetty Handler

Requirements

- Jetty 9.2 or higher

Supported Application Types

For Jetty specific implementations, we support a HandlerWrapper-based install on Jetty 9.2 or higher.

Agent Configuration

Like other Signal Sciences modules, the Jetty Handler supports both Unix domain sockets and TCP sockets for communication with the Signal Sciences Agent. By default, the agent uses Unix domain sockets with the address set to `unix:/var/run/sigsci.sock`. It is possible to override this or specify a TCP socket instead by configuring the `rpc-address` parameter in the Agent.

Additionally, ensure the agent is configured to use the default RPC version: `rpc-version=0`. This can be done by verifying the parameter `rpc-version` is not specified in the agent configuration or if it is specified, ensure that is specified with a value of `0`. Below is an example Agent configuration that overrides the default Unix domain socket value:

```
accesskeyid = "YOUR AGENT ACCESSKEYID"
secretaccesskey = "YOUR AGENT SECRETACCESSKEY"
rpc-address = "127.0.0.1:9999"
```

Download

Download the Signal Sciences Java module manually or access it with Maven.

Download manually

1. Download the Java module archive from https://dl.signalsciences.net/sigsci-module-java/sigsci-module-java_latest.tar.gz.
2. Extract `sigsci-module-java_latest.tar.gz`.
3. Deploy the jars using one of the following options:
   - Copy `sigsci-module-java-{version}-shaded.jar` (an uber jar with all the dependencies bundled) to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`).
   - Copy `sigsci-module-java-{version}.jar` and its dependencies in the `lib` folder to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`). If you already have any of the dependency jar files in your application classpath folder (i.e., for Tomcat in the `WEB-INF\lib`) then it is not necessary to copy them, even if the version numbers are different. The logging jars are optional based on how `slf4j` is configured.

Access with Maven

For projects using Maven for build or deployment, the latest version of Signal Sciences Java modules can be installed by adding XML to the project `pom.xml` file. For example:

```
<repositories>
   <repository>
      <id>sigsci-stable</id>
      <url>https://packages.signalsciences.net/release/maven2</url>
   </repository>
</repositories>

<dependency>
   <groupId>com.signalsciences</groupId>
   <artifactId>sigsci-module-java</artifactId>
   <version>LATEST_MODULE_VERSION</version>
</dependency>
```

Be sure to replace `LATEST_MODULE_VERSION` with the latest release of the Java module. You can find the latest version in our version file at https://dl.signalsciences.net/sigsci-module-java/VERSION.

Install

The installation of the Jetty module varies slightly depending upon whether you deployed Jetty as an embedded or stand alone application.

If you are embedding Jetty within your web application, follow the instructions for “Embedded Jetty”.

Alternatively, if you are deploying your web application to a Jetty instance, follow the instructions for “Standalone Jetty”.

Embedded Jetty

The Signal Sciences Jetty module is currently implemented as a Handler. Edit your application to wrap your existing Handlers with the Signal Sciences Handler.

A typical Jetty based application will add all of the Handlers to a HandlerList, similar to this:

```
ServletHolder defHolder = new ServletHolder("default", DefaultServlet.class);
HandlerList handlers = new HandlerList();

// Servlet: /
defHolder.setInitParameter("pathInfoOnly", "true");
defHolder.setInitParameter("dirAllowed", "true");
defHolder.setInitParameter("acceptRanges", "true");
context.addServlet(defHolder, "/*");
context.addServlet(defHolder, "/");

// Existing App Handlers
handlers.addHandler(context);
handlers.addHandler(new DefaultHandler());

// Add the existing handlers as the server handler
server.setHandler(handlers);

try {
server.start();
server.join();
} catch (Exception e) {
e.printStackTrace();
} finally {
server.stop();
}
```

Add the Signal Sciences Handler around your primary handler. For example:

```
Server server = new Server(InetSocketAddress.createUnresolved("0.0.0.0", 8800));
ServletContextHandler context = new ServletContextHandler();
ServletHolder defHolder = new ServletHolder("default", DefaultServlet.class);
HandlerList handlers = new HandlerList();

// Servlet: /
defHolder.setInitParameter("pathInfoOnly", "true");
defHolder.setInitParameter("dirAllowed", "true");
defHolder.setInitParameter("acceptRanges", "true");
context.addServlet(defHolder, "/*");
context.addServlet(defHolder, "/");

// Existing App Handlers
handlers.addHandler(context);
handlers.addHandler(new DefaultHandler());

// REMOVED: This is replaced by wrapping with the sigsci handler below
//server.setHandler(handlers);

//////////////////////////////////////////////////////////////////////////
// BEGIN ADDITION: Signal Sciences Handler
// Need to also add these imports for SignalSciencesHandler and Timeout:
//   import com.signalsciences.jetty.SignalSciencesHandler;
//   import com.signalsciences.rpc.util.Timeout;
//////////////////////////////////////////////////////////////////////////
// 1. Create a new SignalSciencesHandler
SignalSciencesHandler sigsciHandler = new SignalSciencesHandler();
// 2. Specify the URI of the sigsci-agent rpc-address (Unix)
sigsciHandler.getSigSciConfig().setRpcServerURI(URI.create("unix:/var/run/sigsci.sock"));
// 3. Specify a timeout
sigsciHandler.getSigSciConfig().setRpcTimeout(new Timeout(300, TimeUnit.MILLISECONDS));
sigsciHandler.setHandler(handlers);
// 6. Set the SignalSciencesHandler (wrapper) as the server handler
server.setHandler(sigsciHandler);
///////////////////////////////////////////////////////////////////////////
// END ADDITION
///////////////////////////////////////////////////////////////////////////

try {
server.start();
server.join();
} catch (Exception e) {
e.printStackTrace();
} finally {
server.stop();
}
```

Standalone Jetty

The Signal Sciences Jetty module is currently implemented as a Handler. To use this, you will need to follow the steps below to update your server configuration.

Update Jetty Server Configuration File

In a default Jetty installation, the server configuration file can be found under `{jetty.base}/etc/jetty.xml`. You will need to update the configuration file to wrap the existing Handlers with the Signal Sciences Handler. Modify the stanza in the file that specifies the handler collection to include the Signal Sciences Handler. Below is an example using the out of the box `jetty.xml` file:

```
<Set name="handler">
  <New id="Wrapper" class="com.signalsciences.jetty.SignalSciencesHandler">
    <Call name="setRpcServerURI">
      <Arg>
        <New class="java.net.URI">
          <Arg>unix:/var/run/sigsci.sock</Arg>
         </New>
      </Arg>
     </Call>
     <Call name="setRpcTimeout">
       <Arg>
         <New class="com.signalsciences.rpc.util.Timeout">
           <Arg type="long">300</Arg>
             <Arg>
               <Get class="java.util.concurrent.TimeUnit" name="MILLISECONDS"/>
             </Arg>
         </New>
       </Arg>
     </Call>
     <Set name="handler">
       <New id="Handlers" class="org.eclipse.jetty.server.handler.HandlerCollection">
         <Set name="handlers">
           <Array type="org.eclipse.jetty.server.Handler">
             <Item>
               <New id="Contexts" class="org.eclipse.jetty.server.handler.ContextHandlerCollection"/>
             </Item>
             <Item>
               <New id="DefaultHandler" class="org.eclipse.jetty.server.handler.DefaultHandler"/>
             </Item>
           </Array>
         </Set>
       </New>
     </Set>
```

There are two options for deploying the jars:

- Copy `sigsci-module-java-{version}-shaded.jar` (an uber jar with all the dependencies bundled) to your server classpath.
- Copy `sigsci-module-java-{version}.jar` and its dependencies in the `lib` folder to your server classpath.

Although optional, we recommended adding this library to `{jetty.base}/lib/ext`, as Jetty automatically loads libraries in this path to the server classpath.

Simple Example Server

For a more complete example, see the `sigsci-jetty-simple-example` JAR files included in the distribution. This consists of the binaries, source, and javadoc for a simple working example. The binary JAR is executable and can be run with commands similar to the following. These commands will start the simple server and point it at an agent running on TCP port 5000 on the local host, which require an agent started with `rpc-address = "127.0.0.1:5000"`:

```
$ java -jar examples/sigsci-jetty-simple-example-{version}.jar
tcp://127.0.0.1:5000
00:00:00.384 [main] INFO  c.s.example.SimpleExampleServer - WebRoot is jar:file:/x/sigsci-jetty-simple-example-0.
00:00:00.403 [main] INFO  c.s.example.SimpleExampleServer - Signal Sciences WAF: enabled
00:00:00.501 [main] INFO  c.s.example.SimpleExampleServer - Signal Sciences Simple Example Server started (http:/
00:00:00.986 [qtp123456789-12] INFO  c.s.example.RequestLogger - "GET /test/ HTTP/1.1" 302
```

This example test server will respond with a simple HTML page on the root directory. It can also be used to do basic tests using the `/test/` context. In this test context the following parameters are interpreted:

- `response_time`: Time in milliseconds to delay the response - to test timeouts.
- `response_code`: The HTTP response code to return in the response.
- `size`: The size of the response body in bytes.

For example:

```
$ curl -D- "http://127.0.0.1:8800/test/?response_code=302&response_time=10&size=86"
HTTP/1.1 302 Found
Date: Sat, 01 Sep 2016 00:00:00 GMT
Location: /
Content-Length: 86
Server: Jetty(9.2.z-SNAPSHOT)
```

VMware Tanzu Install

The Signal Sciences Service Broker is a service tile for VMware Tanzu that allows you to deploy Signal Sciences within your VMware Tanzu apps.

See the Signal Sciences Service Broker for VMware Tanzu partner documentation for additional information about VMware Tanzu and the Signal Sciences Service Broker service tile.

Installation

1. Download the product file from Pivotal Network.
2. Log into the Ops Manager Installation Dashboard.
3. Click **Import a Product** and select the downloaded Signal Sciences Service Broker tile.
4. In the Ops Manager **Available Products** view, click **Add** next to the uploaded Signal Sciences Service Broker tile to add it to your staging area.
5. Click the newly added **Signal Sciences Service Broker** tile.
6. Click the **Buildpack Settings** tab. The Buildpack Settings menu page appears.
7. Set the `sigsci_buildpack_decorator` Buildpack Order to zero.
8. Click **Save**.
9. Return to the Ops Manager Installation Dashboard and click **Apply Changes** to install the Signal Sciences Service Broker for VMware Tanzu tile.

For additional information regarding installing the Signal Sciences Service Broker service tile, see the installation instructions provided in our partner documentation.

Introduction

In this example, the Signal Sciences agent runs in a sidecar container and proxies all incoming requests for inspection before sending them upstream to the application container.

Integrating the Signal Sciences Agent

The Signal Sciences Agent can be installed as a sidecar into each pod or as a service for some specialized needs.

The recommended way of installing the Signal Sciences Agent in Kubernetes is by integrating the `sigsci-agent` into a pod as a sidecar. This means adding the `sigsci-agent` as an additional container to the Kubernetes pod. As a sidecar, the agent will scale with the app/service in the pod instead of having to do this separately. However, in some situations, it may make more sense to install the `sigsci-agent` container as a service and scale it separately from the application.

The `sigsci-agent` container can be configured in various ways depending on the installation type and module being used.

You can use the `preStop` container hook to slow the pod's shutdown and ensure drain timeouts are met.

```
preStop:
  exec:
    command:
      - sleep
      - "30"
```

Getting and Updating the Signal Sciences Agent Container Image

An official `signalsciences/sigsci-agent` container image is available from the Signal Sciences account on Docker Hub.

Alternatively, if you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

These instructions reference the `latest` version of the agent with `imagePullPolicy: Always`, which will pull the latest agent version even if one already exist locally. This is so the documentation does not fall out of date and anyone using this will not have an agent that stays stagnant. However, this may not be what if you need to keep installations consistent or on a specific version of the agent. In these cases, you should specify an agent version. Images on Docker Hub are tagged with their versions and a list of versions is available on Docker Hub.

Whether you choose to use the `latest` image or a specific version, there are a few items to consider to keep the agent up-to-date.

Using the `latest` Signal Sciences Container Image

If you do choose to use the `latest` image, then you will want to consider how you will keep the agent up to date.

- If you have used the `imagePullPolicy: Always` option, then the latest image will be pulled on each startup and your agent will continue to get updates.
- Alternatively, you may instead choose to manually update the local cache by periodically forcing a pull instead of always pulling on startup:

  ```
  docker pull signalsciences/sigsci-agent:latest
  ```

  Then, use `latest` with `imagePullPolicy: Never` set in the configuration so that pulls are never done on startup (only manually as above):

  ```
  - name: sigsci-agent
      image: signalsciences/sigsci-agent:latest
      imagePullPolicy: Never
      ...
  ```

Using a Versioned Signal Sciences Container Image

To use a specific version of the agent, replace `latest` with the agent version (represented here by `x.xx.x`). You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:x.xx.x
    imagePullPolicy: IfNotPresent
    ...
```

Using a Custom Tag for the Signal Sciences Container Image

It is also possible to apply a custom tag to a local agent image. To do this, pull the agent image (by version or use `latest`), apply a custom tag, then use that custom tag in the configuration. You will need to specify `imagePullPolicy: Never` so local images are only updated manually. After doing so, you will need to periodically update the local image to keep the agent up-to-date.

For example:

```
docker pull signalsciences/sigsci-agent:latest
docker tag signalsciences/sigsci-agent:latest signalsciences/sigsci-agent:testing
```

Then use this image tag in the configuration:

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:testing
    imagePullPolicy: Never
...
```

Configuring the Signal Sciences Agent Container

Agent configuration is normally done via the environment. Most configuration options are available as environment variables. Environment variables names have the configuration option name all capitalized, prefixed with `SIGSCI_` and any dashes (-) changed to underscores (_). For example, the max-procs option would become the `SIGSCI_MAX_PROCS` environment variable. For more details on what options are available, see the Agent Configuration documentation.

The `sigsci-agent` container has a few required options that need to be configured:

- Agent credentials (**Agent Access Key** and **Agent Secret Key**).
- A volume to write temporary files.

Agent Credentials

The `sigsci-agent` credentials are configured with two environment variables. These variables must be set or the agent will not start.

- **SIGSCI_ACCESSKEYID**: The **Agent Access Key** identifies which site in the Signal Sciences console that the agent is configured for.
- **SIGSCI_SECRETACCESSKEY**: The **Agent Secret Key** is the shared secret key to authenticate and authorize the agent.

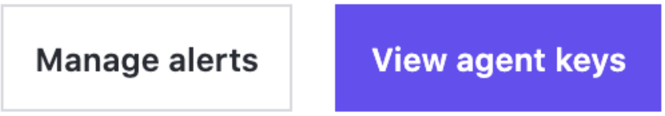
The credentials can be found by following these steps:

1. Log in to the Signal Sciences console.

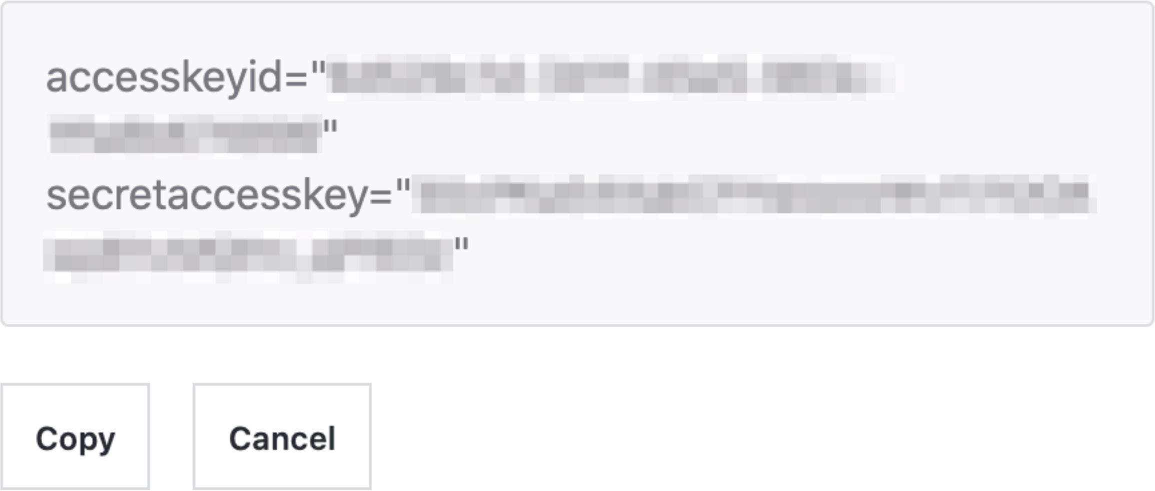
2. From the **Sites** menu, select a site if you have more than one site.

3. Click **Agents** in the navigation bar. The agents page appears.

4. Click **View agent keys**. The agent keys window appears.



5. Copy the **Agent Access Key** and **Agent Secret Key**.



```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

Because of the sensitive nature of these values, we recommend you use the built in `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded values into the deployment configuration. This also makes any desired agent credential rotation easier to manage by having to change them in only one place.

Use the `valueFrom` option instead of the `value` option to use the `secrets` functionality. For example:

```
env:
  - name: SIGSCI_ACCESSKEYID
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: accesskeyid
  - name: SIGSCI_SECRETACCESSKEY
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: secretaccesskey
```

The `secrets` functionality keeps secrets in various stores in Kubernetes. This guide uses the generic secret store in its examples, however any equivalent store can be used. Agent secrets can be added to the generic secret store using YAML similar to the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: sigsci.my-site-name-here
stringData:
  accesskeyid: 12345678-abcd-1234-abcd-1234567890ab
  secretaccesskey: abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

This can also be created from the command line with `kubectl` such as with the following example:

```
kubectl create secret generic sigsci.my-site-name-here \
  --from-literal=accesskeyid=12345678-abcd-1234-abcd-1234567890ab \
  --from-literal=secretaccesskey=abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

Additional information about Kubernetes `secrets` functionality can be found here.

Agent Temporary Volume

For added security, we recommended the `sigsci-agent` container be executed with the root filesystem mounted as read only. However, the agent still needs to write some temporary files such as the socket file for RPC communication and some periodically updated files such as GeoIP data.

The recommended way of creating a writeable volume is to use the builtin `emptyDir` volume type. This is typically configured in the `volumes` section of a deployment, as shown in the following example:

```
volumes:
  - name: sigsci-tmp
    emptyDir: {}
```

Containers will then mount this volume at `/sigsci/tmp`:

```
volumeMounts:
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
```

The default in the official agent container image is to have the temporary volume mounted at `/sigsci/tmp`. If this needs to be moved for the agent container, then the following agent configuration options should also be changed from their defaults to match the new mount location:

- `rpc-address` defaults to `/sigsci/tmp/sigsci.sock`
- `shared-cache-dir` defaults to `/sigsci/tmp/cache`

Signal Sciences agent as a reverse proxy in front of a web application without the Signal Sciences module

If your web application does not support a Signal Sciences Module (or you prefer not to install a module), then you can configure the `sigsci-agent` container to run as a reverse proxy in front of the web application in the same pod.

To configure the Signal Sciences agent to run in reverse proxy mode in a sidecar container, you must:

- Add the `sigsci-agent` container to the pod, configured in reverse proxy mode to:
  - listen for incoming requests (on a new port or by reconfiguring your application or Kubernetes service accordingly)
  - proxy requests to your web application container
- Add an `emptyDir{}` volume as a place for the `sigsci-agent` to write temporary data.

The following configuration exposes an example application (`helloworld`) on port `8000`, adding the `sigsci-agent` as a reverse proxy listener on a new port `8001` with an upstream of the example web application port `8000`.

Add the Signal Sciences agent as a reverse proxy

```
...
      containers:
      # Example helloworld app running on port 8000 without sigsci configured
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
        - localhost:8000
        ports:
        - containerPort: 8000
      # Signal Sciences Agent running in reverse proxy mode (SIGSCI_REVPROXY_LISTENER configured)
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: Always
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              name: sigsci.my-site-name-here
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
              name: sigsci.my-site-name-here
```

```
    - name: SIGSCI_REVPROXY_LISTENER
      value: "http:{listener='http://0.0.0.0:8001',upstreams='http://0.0.0.0:8000',access-log='/dev/stdout'}"
    ports:
    - containerPort: 8001
    securityContext:
      # The sigsci-agent container should run with its root filesystem read only
      readOnlyRootFilesystem: true
    volumeMounts:
    # Default volume mount location for sigsci-agent writeable data
    # NOTE: Also change `SIGSCI_SHARED_CACHE_DIR` (default `/sigsci/tmp/cache`)
    #       if mountPath is changed, but best not to change.
    - name: sigsci-tmp
      mountPath: /sigsci/tmp
```

Note: The above modification assumes that `sigsci` secrets were added to the system.

Adding the Signal Sciences agent temp volume definition to the deployment

You must define the agent temp volume for use by the other containers in the pod. This example uses the builtin `emptyDir: {}` volume type.

```
...
   volumes:
   # Define a volume where sigsci-agent will write temp data and share the socket file,
   # which is required with the root filesystem is mounted read only
   - name: sigsci-tmp
     emptyDir: {}
```

Changing the service definition and adding the Signal Sciences agent as a reverse proxy

In the example above, the `sigsci-agent` reverse proxy listens on a new port, leaving the original application listener in place. You may wish for requests to be routed to the `sigsci-agent` at the original application port to make the agent addition as seamless as possible. One way to do this is to modify the Kubernetes *service definition* to route traffic to the `sigsci-agent` reverse proxy listener port instead of directly to the web application.

Change the service definition to point to the Signal Sciences agent port

Change the service `targetPort` from pointing directly to the application, to instead point to the `sigsci-agent` reverse proxy listener port. The `sigsci-agent` will then proxy to the application port:

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld
  labels:
    app: helloworld
spec:
  ports:
  - name: http
    port: 8000
    # Target is now sigsci-agent on port 8001
    targetPort: 8001
  selector:
    app: helloworld
  type: LoadBalancer
```

Ubuntu NGINX 1.10-1.14

Add the package repositories

Add the version of the Ubuntu package repository that you want to use:

Ubuntu 22.04 - jammy

```
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/ubuntu/ jammy ma
sudo apt-get update
```

Ubuntu 20.04 - focal

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ focal main" | sudo tee /etc/apt/sources.list.d/sigs
```

Ubuntu 18.04 - bionic

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ bionic main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 16.04 - xenial

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ xenial main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 14.04 - trusty

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ trusty main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 12.04 - precise

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ precise main" | sudo tee /etc/apt/sources.list.d/si
```

Enable Lua for NGINX

Some older versions of NGINX don’t support native loading of Lua modules. Therefore, we require NGINX to be built with Lua and LuaJIT support. You must first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

Install the Lua NGINX Module

Install the dynamic Lua NGINX Module appropriate for your NGINX distribution.

NGINX.org distribution

1. Install the Lua NGINX Module.

   - NGINX 1.12.1 or higher

     ```
     sudo apt-get install nginx-module-lua
     ```

   - NGINX 1.11

     ```
     sudo apt-get install nginx111-lua-module
     ```

   - NGINX 1.10

     ```
     sudo apt-get install nginx110-lua-module
     ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the line that starts with `pid`:

   ```
   load_module modules/ndk_http_module.so;
   load_module modules/ngx_http_lua_module.so;
   ```

3. Restart the NGINX service to initialize the new module.

Ubuntu distribution

Enable Lua by installing the `nginx-extras` package.

```
sudo apt-get install nginx-extras && sudo service nginx restart
```

Check that Lua is loaded correctly

Load the following config (e.g., `sigsci_check_lua.conf`) with NGINX to verify that Lua has been loaded properly:

```
# Config just to test for lua jit support
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
# Also, for some nginx-1.10.nn installed from nginx-extras package, you may
# need to specify the load directives.
# Given the above uncomment the following:
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768;
    # multi_accept on;
}
http {
init_by_lua '

local m = {}
local ngx_lua_version = "dev"

if ngx then
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
if not r then
  error("ERROR: No lua jit support: No support for SigSci Lua module")
else

  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
  end

end

';
```

Example of a successfully loaded config and its output

```
$ nginx -t -c <your explicit path>/sigsci_check_lua.conf

nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
nginx: the configuration file <your explicit path>/sigsci_check_lua.conf syntax is ok
nginx: configuration file <your explicit path>/sigsci_check_lua.conf test is successful
```

Install the NGINX module

1. Install the module.

   ```
   apt-get install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX Service to initialize the new module.

   - Ubuntu 15.04 or higher

     ```
     sudo systemctl restart nginx
     ```

   - Ubuntu 14.04 or lower

     ```
     sudo restart nginx
     ```

HAProxy Module Install

Requirements

- HAProxy 1.7 or higher
- Lua module enabled on host
- Signal Sciences agent installed for your OS

> **Note:** The HAProxy module can be used with any OS because it is Lua code.

Installation

Follow these steps to install the HAProxy module.

Configure the agent

> **Note:** This section may not be required for your installation. If you have set HAProxy's chroot directory, you will need to modify the commands below to reflect your custom chroot directory by following the instructions in this section.

If your HAProxy configuration has been modified to set a chroot directory for HAProxy, you will need to update your Signal Sciences agent configuration to reflect this. The default location of the agent socket file (`/var/run/sigsci.sock`) will be inaccessible to the HAProxy module outside of your specified chroot directory.

1. Create the directory structure for the Unix domain socket by running the following command, replacing `HAPROXY-CHROOT-DIRECTORY` with your HAProxy chroot directory:

   ```
   sudo mkdir -p /HAPROXY-CHROOT-DIRECTORY/var/run/
   ```

2. Add the following line to your agent configuration file (located by default at `/etc/sigsci/agent.conf`) to specify the new socket file location under chroot:

   ```
   rpc-address="unix:/haproxy-chroot-directory/var/run/sigsci.sock"
   ```

Module installation

Install the HAProxy module using a package manager or manually.

Install with Package Manager

The HAProxy module can be installed via the package manager of most major OS versions:

- Debian: `sudo apt-get install sigsci-module-haproxy`
- Ubuntu: `sudo apt-get install sigsci-module-haproxy`

Install manually

Alternatively, the HAProxy module can be manually installed.

1. Download the latest version of the HAProxy module.

   ```
   wget https://dl.signalsciences.net/sigsci-module-haproxy/sigsci-module-haproxy_latest.tar.gz
   ```

2. Create the directory the HAProxy module will be moved to.

   ```
   sudo mkdir -p /usr/local/lib/lua/5.3/sigsci/
   ```

3. Extract the HAProxy archive to the new directory.

   ```
   tar xvzf sigsci-module-haproxy_latest.tar.gz -C /usr/local/lib/lua/5.3/sigsci/
   ```

HAProxy configuration changes

After installing the HAProxy module, edit your HAProxy configuration file (located by default at `/etc/haproxy/haproxy.cfg`) to add the following lines:

```
global
    ...
    #Signal Sciences
    lua-load /usr/local/lib/lua/5.3/sigsci/SignalSciences.lua
    pidfile /var/run/haproxy.pid
    ...

frontend http-in
    ...
    #Signal Sciences
    #Required for buffering request body to ensure inspection is performed
    #Can also be set in the defaults section
    option http-buffer-request

    #Signal Sciences
    http-request lua.sigsci_prerequest
    http-response lua.sigsci_postrequest
    ...
```

HAProxy 1.9+

If you are running HAProxy 1.9 or higher, in addition to the HAProxy configuration file edits above, you will also need to add the following line to the `frontend http-in` context:

```
    ...
    # for haproxy-1.9 and above add the following:
    http-request use-service lua.sigsci_send_block if { var(txn.sigsci_block) -m bool }
    ...
```

Configuration

Configuration changes are typically not required for the HAProxy module to work. However, it is possible to override the default settings if needed. To do so, you must create an `override.lua` file in which to add these configuration directives. Then, update the `global` section of your HAProxy config file (`/usr/local/etc/haproxy/haproxy.cfg`) to load this over-ride config file.

Example of configuration

```
global
   ...
   lua-load /path/to/override.lua
   ...
```

Over-ride Directives

`sigsci.agenthost`	The IP address or path to unix domain socket the SignalSciences Agent is listening on, default: `/var/run/sigsci.sock` (unix domain socket).
`sigsci.agentport`	The local port (when using TCP) that the agent listens on, default: `nil`
`sigsci.timeout`	Agent socket timeout (in seconds), default: `1` (`0` means off).
`sigsci.maxpost`	Maximum POST body site in bytes, default: `100000`
`sigsci.extra\_blocking\_resp\_hdr`	User may supply a response header to be added upon 406 responses, default: `""`

Example of over-ride configuration

```
sigsci.agenthost = "192.0.2.243"
sigsci.agentport = 9090
sigsci.extra_blocking_resp_hdr = "Access-Control-Allow-Origin: https://example.com"
```

Upgrading

To upgrade the HAProxy module, download and install the latest version of the module.

After installing, restart HAProxy for the new module version to be detected.

Extracting Your Data

Signal Sciences stores requests that contain attacks and anomalies, with some qualifications. If you would like to extract this data in bulk for ingestion into your own systems, we offer a request feed API endpoint which makes available a feed of recent data, suitable to be called by (for example) an hourly cron.

This functionality is typically used by SOC teams to automatically import data into SIEMs such as Splunk, ELK, and other commercial systems.

Data extraction vs searching

We have a separate API endpoint for searching request data. Its use case is for finding requests that meet certain criteria, as opposed to bulk data extraction:

Searching	Data Extraction
Search using full query syntax	Returns all requests, optionally filtered by signals
Limited to 1,000 requests	Returns all requests
Window: up to 7 days at a time	Window: past 24 hours
Retention: 30 days	24 hours

Time span restrictions

The following restrictions are in effect when using this endpoint:

- The `until` parameter has a maximum of **five minutes** in the past. This is to allow our data pipeline sufficient time to process incoming requests - see below.
- The `from` parameter has a minimum value of **24 hours and five minutes** in the past.
- Both the `from` and `until` parameters must fall on full minute boundaries.
- Both the `from` and `until` parameters require Unix timestamps with second level detail (e.g., `1445437680`).

Delayed data

A five-minute delay is enforced to build in time to collect and aggregate data across all of your running agents, and then ingest, analyze, and augment the data in our systems. Our five-minute delay is a tradeoff between data that is both timely and complete.

Pagination

This endpoint returns data **1,000** requests at a time. If the time span specified contains more than 1,000 requests, a `next` url will be provided to retrieve the next batch. Each `next` url is valid for one minute from the time it's generated.

Sort order

As a result of our data warehousing implementation, the data you get back from this endpoint will be complete for the time span specified, but is not guaranteed to be sorted. Once all data for the given time span has been accumulated, it can be sorted using the `timestamp` field, if necessary.

Rate limiting

- **Five** per corp

Example usage

A common way to use this endpoint is to set up a cron that runs at 5 minutes past each hour and fetches the previous full hour's worth of data. In the example below, we calculate the previous full hour's start and end timestamps and use them to call the API.

Python

```
import sys, requests, os, calendar, json
from datetime import datetime, timedelta

if 'SIGSCI_EMAIL' not in os.environ or 'SIGSCI_TOKEN' not in os.environ or 'SIGSCI_CORP' not in os.environ:
    print ("ERROR: You need to define SIGSCI_EMAIL, SIGSCI_TOKEN and SIGSCI_CORP environment variables")
    print ("Please fix and run again. Exiting....")
    sys.exit(1)

# Initial setup
api_host = 'https://dashboard.signalsciences.net'
email = os.environ.get('SIGSCI_EMAIL')
token = os.environ.get('SIGSCI_TOKEN')
corp_name = os.environ.get('SIGSCI_CORP')
# List of comma-delimited sites that you want to extract data from
site_names = [ 'site123', 'site345' ]

# Calculate UTC timestamps for the previous full hour
# For example, if now is 9:05 AM UTC, the timestamps will be 8:00 AM and 9:00 AM
until_time = datetime.utcnow().replace(minute=0, second=0, microsecond=0)
from_time = until_time - timedelta(hours=1)
until_time = calendar.timegm(until_time.utctimetuple())
from_time = calendar.timegm(from_time.utctimetuple())

# Set up Headers will use
headers = {
    'Content-type': 'application/json',
    'Content-Encoding': 'gzip',
    'x-api-user' : email,
    'x-api-token': token
}

for site_name in site_names:

    url = api_host + ('/api/v0/corps/%s/sites/%s/feed/requests?from=%s&until=%s' % (corp_name, site_name, from_ti
    first = True

    print ("{ \"site_name\": \"%s\", \"data\": [" % (site_name))

    # Loop across all the data and output the data in one big JSON object
    while True:
        response_raw = requests.get(url, headers=headers)
        if response_raw.status_code != 200:
            sys.stderr.write("There was an error fetching requests for site_name=%s.\nURL=%s failed" % (site_name,
            break

        response = json.loads(response_raw.text)

        for request in response['data']:
            data = json.dumps(request)
            if first:
                first = False
        next_url = response['next']['uri']
        if next_url == '':
            break
        url = api_host + next_url

    print ("\n] }")
```

Red Hat Agent Installation

This guide explains how to install the Signal Sciences agent on Red Hat.

Prerequisites

Before you begin, determine the version of Red Hat/CentOS you want to use.

Add the package repository

Begin the agent installation by adding the version of the Red Hat/CentOS package repository that you want to use.

Red Hat/CentOS 8

To add the Red Hat/CentOS 8 package, run the following script:

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/8/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat/CentOS 7

To add the Red Hat/CentOS 7 package, run the following script:

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat/CentOS 6

> **Note:** After Q2 2017, RHEL6 and CentOS 6 will exit Production Phase 2 according to the Red Hat Enterprise Linux Life Cycle. Only limited,critical security fixes will be issued. You will need to review the lifecycle document for details and plan appropriately.

To add the Red Hat/CentOS 6 package, run the following script:

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
```

```
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Install and configure the Signal Sciences Agent package

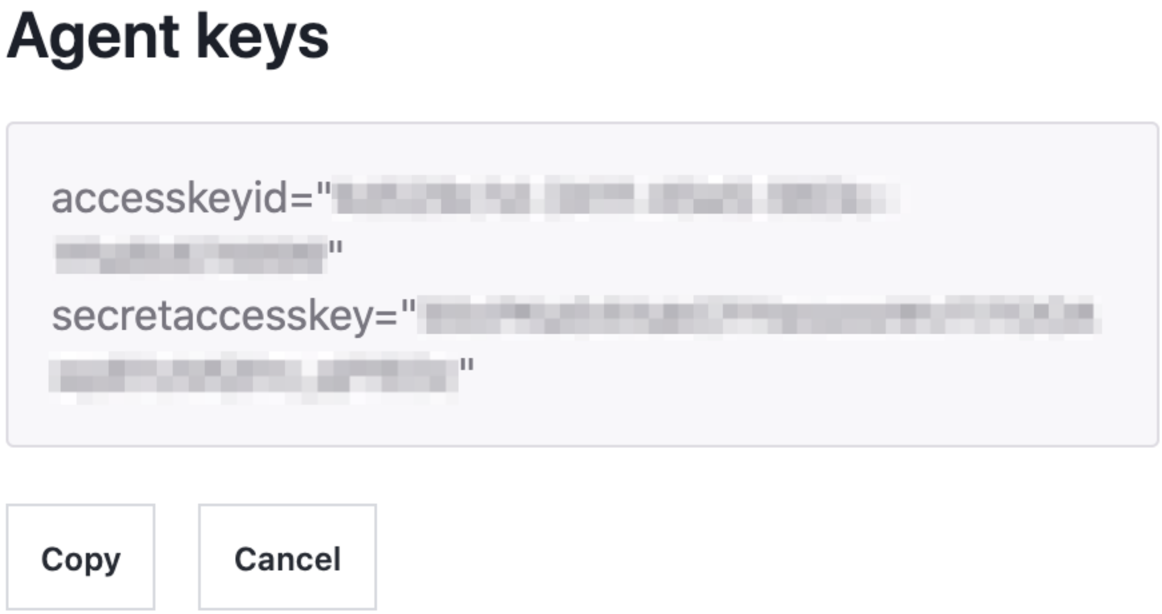
Now that you've downloaded the Red Hat/CentOS package repository, you can install the Signal Sciences Agent package.

Run the following command to install the Signal Sciences Agent package:

```
sudo yum install sigsci-agent
```

Once the agent package is installed, you must create an agent configuration file and add the **Agent Access Key** and **Agent Secret Key**:

1. Create an empty agent configuration file in the following location: `/etc/sigsci/agent.conf`.
2. Log in to the Signal Sciences console.
3. From the **Sites** menu, select the site that you want to give the agent access to.
4. Click the **Agents** link in the site navigation bar. The agents page appears.
5. Click the **View agent keys** button. The agent keys window appears.
6. Click the **Copy** button to copy the **Agent Access Key** and **Agent Secret Key** to your clipboard.



Agent keys

accesskeyid="[redacted]"
secretaccesskey="[redacted]"

7. Navigate to the agent configuration file and paste the **Agent Access Key** and **Agent Secret Key** into the file.

   ```
   accesskeyid = "AGENTACCESSKEYHERE"
   secretaccesskey = "AGENTSECRETACCESSKEYHERE"
   ```

8. Save the agent configuration file.

Start the Signal Sciences Agent

Now that you've installed and configured the agent package, you can start the Signal Sciences agent.

For Red Hat/CentOS versions 7 and above, run the following command to start the Signal Sciences agent:

```
sudo systemctl start sigsci-agent
```

For Red Hat/CentOS 6, run the following command to start the Signal Sciences agent:

```
start sigsci-agent
```

Optionally, enable the agent auto-update service. The service checks the Signal Sciences package downloads site for a new version of the agent and updates the agent when a new version is available.

Red Hat Apache Module Install

1. Install the Signal Sciences Apache module.

   - Red Hat CentOS 8 / RHEL 8

     ```
     sudo yum install sigsci-module-apache
     ```

   - Red Hat CentOS 7 / RHEL 7

     ```
     sudo yum install sigsci-module-apache
     ```

   - Red Hat CentOS 6 / RHEL 6 with Apache 2.4

     ```
     sudo yum install sigsci-module-apache24
     ```

   - Red Hat CentOS 6 / RHEL 6 with Apache 2.2 64-bit

     ```
     sudo yum install sigsci-module-apache
     ```

   - Red Hat CentOS 6 / RHEL 6 with Apache 2.2 32-bit

     ```
     sudo yum install sigsci-module-apache22
     ```

2. Add the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the **Dynamic Shared Object (DSO) Support** section to enable the Signal Sciences Apache module:

   ```
   LoadModule signalsciences_module /etc/httpd/modules/mod_signalsciences.so
   ```

3. Restart Apache.

   - Red Hat CentOS 8 / RHEL 8

     ```
     sudo systemctl restart httpd
     ```

   - Red Hat CentOS 7 / RHEL 7

     ```
     sudo systemctl restart httpd
     ```

   - Red Hat CentOS 6 / RHEL 6

     ```
     sudo service httpd restart
     ```

Next Steps

Verify the agent and module installation and explore module options.

Installing the Java Module as a Netty Handler

The Signal Sciences Netty module is implemented as a handler which inspects `HttpRequest` events before forwarding the event to the next handler in the pipeline.

Download

Download the Signal Sciences Java module manually or access it with Maven.

Download manually

1. Download the Java module archive from https://dl.signalsciences.net/sigsci-module-java/sigsci-module-java_latest.tar.gz.
2. Extract `sigsci-module-java_latest.tar.gz`.
3. Deploy the jars using one of the following options:
   - Copy `sigsci-module-java-{version}-shaded.jar` (an uber jar with all the dependencies bundled) to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`).
   - Copy `sigsci-module-java-{version}.jar` and its dependencies in the `lib` folder to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`). If you already have any of the dependency jar files in your

Access with Maven

For projects using Maven for build or deployment, the latest version of Signal Sciences Java modules can be installed by adding XML to the project `pom.xml` file. For example:

```
<repositories>
   <repository>
      <id>sigsci-stable</id>
      <url>https://packages.signalsciences.net/release/maven2</url>
   </repository>
</repositories>

<dependency>
   <groupId>com.signalsciences</groupId>
   <artifactId>sigsci-module-java</artifactId>
   <version>LATEST_MODULE_VERSION</version>
</dependency>
```

Be sure to replace `LATEST_MODULE_VERSION` with the latest release of the Java module. You can find the latest version in our version file at https://dl.signalsciences.net/sigsci-module-java/VERSION.

Install and configure

Create a new instance of `WafHandler` for every new connection.

- `WafHandler` must be added after `FlowControlHandler`.
- `HttpObjectAggregator` handler should be added before `FlowControlHandler` to inspect HTTP Post body.
- `WafHandler` may send `HttpResponse` for blocked request.

Example deployment

```
// Update configuration
WafHandler.getSigSciConfig().setMaxPost(40000);

// start server and handle requests
new ServerBootstrap()
.group(bossGroup, workerGroup)
.channel(NioServerSocketChannel.class)
.childHandler(
  new ChannelInitializer<SocketChannel>() {
      @Override
      public void initChannel(SocketChannel ch) throws Exception {
    ch.pipeline()
    .addLast(new HttpServerCodec())
    .addLast(new HttpObjectAggregator(6 * (1 << 20)))
    .addLast(new FlowControlHandler())
    .addLast("waf", new WafHandler())
    .addLast(new SimpleChannelInboundHandler<FullHttpRequest>() {

      // send response

    });
      }
  })
.bind(8080)
.sync();
```

Kubernetes Agent + Module

Introduction

The Signal Sciences Agent can be installed as a sidecar into each pod or as a service for some specialized needs.

The recommended way of installing the Signal Sciences Agent in Kubernetes is by integrating the `sigsci-agent` into a pod as a sidecar. This means adding the `sigsci-agent` as an additional container to the Kubernetes pod. As a sidecar, the agent will scale with the app/service in the pod instead of having to do this separately. However, in some situations, it may make more sense to install the `sigsci-agent` container as a service and scale it separately from the application.

The `sigsci-agent` container can be configured in various ways depending on the installation type and module being used.

You can use the `preStop` container hook to slow the pod's shutdown and ensure drain timeouts are met.

```
preStop:
  exec:
    command:
      - sleep
      - "30"
```

Getting and Updating the Signal Sciences Agent Container Image

An official `signalsciences/sigsci-agent` container image is available from the Signal Sciences account on Docker Hub.

Alternatively, if you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

These instructions reference the `latest` version of the agent with `imagePullPolicy: Always`, which will pull the latest agent version even if one already exist locally. This is so the documentation does not fall out of date and anyone using this will not have an agent that stays stagnant. However, this may not be what if you need to keep installations consistent or on a specific version of the agent. In these cases, you should specify an agent version. Images on Docker Hub are tagged with their versions and a list of versions is available on Docker Hub.

Whether you choose to use the `latest` image or a specific version, there are a few items to consider to keep the agent up-to-date.

Using the `latest` Signal Sciences Container Image

If you do choose to use the `latest` image, then you will want to consider how you will keep the agent up to date.

- If you have used the `imagePullPolicy: Always` option, then the latest image will be pulled on each startup and your agent will continue to get updates.
- Alternatively, you may instead choose to manually update the local cache by periodically forcing a pull instead of always pulling on startup:

  ```
  docker pull signalsciences/sigsci-agent:latest
  ```

  Then, use `latest` with `imagePullPolicy: Never` set in the configuration so that pulls are never done on startup (only manually as above):

  ```
  - name: sigsci-agent
      image: signalsciences/sigsci-agent:latest
      imagePullPolicy: Never
      ...
  ```

Using a Versioned Signal Sciences Container Image

To use a specific version of the agent, replace `latest` with the agent version (represented here by `x.xx.x`). You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:x.xx.x
    imagePullPolicy: IfNotPresent
    ...
```

This will pull the specified agent version and cache it locally. If you use this method, then it is recommended that you parameterize the agent image, using Helm or similar, so that it is easier to update the agent images later on.

Using a Custom Tag for the Signal Sciences Container Image

manually. After doing so, you will need to periodically update the local image to keep the agent up-to-date.

For example:

```
docker pull signalsciences/sigsci-agent:latest
docker tag signalsciences/sigsci-agent:latest signalsciences/sigsci-agent:testing
```

Then use this image tag in the configuration:

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:testing
    imagePullPolicy: Never
...
```

Configuring the Signal Sciences Agent Container

Agent configuration is normally done via the environment. Most configuration options are available as environment variables. Environment variables names have the configuration option name all capitalized, prefixed with `SIGSCI_` and any dashes (-) changed to underscores (_). For example, the max-procs option would become the `SIGSCI_MAX_PROCS` environment variable. For more details on what options are available, see the Agent Configuration documentation.

The `sigsci-agent` container has a few required options that need to be configured:

- Agent credentials (**Agent Access Key** and **Agent Secret Key**).
- A volume to write temporary files.

Agent Credentials

The `sigsci-agent` credentials are configured with two environment variables. These variables must be set or the agent will not start.

- **SIGSCI_ACCESSKEYID**: The **Agent Access Key** identifies which site in the Signal Sciences console that the agent is configured for.
- **SIGSCI_SECRETACCESSKEY**: The **Agent Secret Key** is the shared secret key to authenticate and authorize the agent.

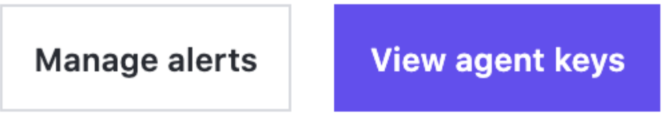
The credentials can be found by following these steps:

1. Log in to the Signal Sciences console.

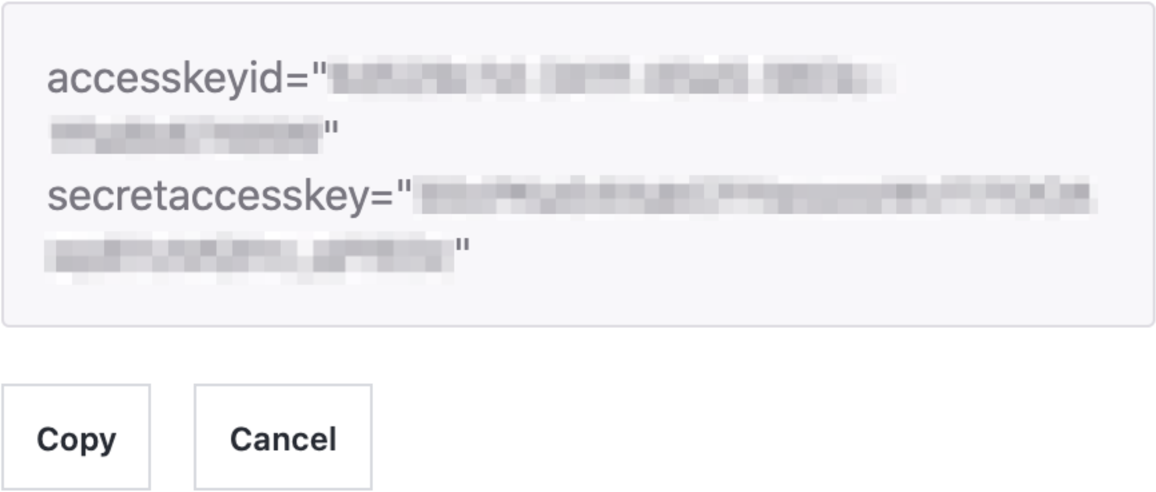
2. From the **Sites** menu, select a site if you have more than one site.

3. Click **Agents** in the navigation bar. The agents page appears.

4. Click **View agent keys**. The agent keys window appears.



5. Copy the **Agent Access Key** and **Agent Secret Key**.



```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

Because of the sensitive nature of these values, we recommend you use the built in `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded values into the deployment configuration. This also makes any desired agent credential rotation easier to manage by having to change them in only one place.

Use the `valueFrom` option instead of the `value` option to use the `secrets` functionality. For example:

```
env:
  - name: SIGSCI_ACCESSKEYID
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: accesskeyid
  - name: SIGSCI_SECRETACCESSKEY
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: secretaccesskey
```

The `secrets` functionality keeps secrets in various stores in Kubernetes. This guide uses the generic secret store in its examples, however any equivalent store can be used. Agent secrets can be added to the generic secret store using YAML similar to the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: sigsci.my-site-name-here
stringData:
  accesskeyid: 12345678-abcd-1234-abcd-1234567890ab
  secretaccesskey: abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

This can also be created from the command line with `kubectl` such as with the following example:

```
kubectl create secret generic sigsci.my-site-name-here \
  --from-literal=accesskeyid=12345678-abcd-1234-abcd-1234567890ab \
  --from-literal=secretaccesskey=abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

Additional information about Kubernetes `secrets` functionality can be found here.

Agent Temporary Volume

For added security, we recommended the `sigsci-agent` container be executed with the root filesystem mounted as read only. However, the agent still needs to write some temporary files such as the socket file for RPC communication and some periodically updated files such as GeoIP data.

The recommended way of creating a writeable volume is to use the builtin `emptyDir` volume type. This is typically configured in the `volumes` section of a deployment, as shown in the following example:

```
volumes:
  - name: sigsci-tmp
    emptyDir: {}
```

Containers will then mount this volume at `/sigsci/tmp`:

```
volumeMounts:
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
```

The default in the official agent container image is to have the temporary volume mounted at `/sigsci/tmp`. If this needs to be moved for the agent container, then the following agent configuration options should also be changed from their defaults to match the new mount location:

- `rpc-address` defaults to `/sigsci/tmp/sigsci.sock`
- `shared-cache-dir` defaults to `/sigsci/tmp/cache`

Signal Sciences agent with a web application and Signal Sciences module installed

This deployment example configures the example `helloworld` application to use the `sigsci-agent` via RPC and deploys the `sigsci-agent` container as a sidecar to process these RPC requests.

To configure Signal Sciences with this deployment type you must:

- Modify your application to add the appropriate Signal Sciences module, configured it to communicate with a `sigsci-agent` via RPC.
- Add the sigsci-agent container to the pod, configured in RPC mode.
- Add an `emptyDir{}` volume as a place for the `sigsci-agent` to write temporary data and share the RPC address.

Modifying and configuring the application container

The `helloworld` example is a language based module (Golang) that has already been modified to enable communication to the `sigsci-agent` via RPC if configured to do so. This configuration is done via arguments passed to the `helloworld` example application as follows:

- Listening Address (defaults to `localhost:8000`).
- Optional Signal Sciences Agent RPC Address (default is to not use the `sigsci-agent`). Other language based modules are similar. Web server based modules must have the Signal Sciences module added to the container.

For this `helloworld` application to work with the `sigsci-agent` it must have the `sigsci-agent` address configured as the second program argument and the `sigsci-tmp` volume mounted so that it can write to the socket file:

```
...
      containers:
      # Example helloworld app running on port 8000 against sigsci-agent via UDP /sigsci/tmp/sigsci.sock
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
          # Address for the app to listen on
          - localhost:8000
          # Address sigsci-agent RPC is listening on
          - /sigsci/tmp/sigsci.sock
        ports:
        - containerPort: 8000
        volumeMounts:
        # Shared mount with sigsci-agent container where the socket is shared via emptyDir volume
        - name: sigsci-tmp
          mountPath: /sigsci/tmp
```

Adding and configuring the Signal Sciences agent container as a sidecar

configured to communicate with the sigsci-agent via this UDS socket. The deployment YAML must be modified from the example above by adding a second argument to specify the `sigsci-agent` RPC address of `/sigsci/tmp/sigsci.sock`.

> **Note:** It is possible to use a TCP based listener for the `sigsci-agent` RPC, but this is not recommended for performance reasons. If TCP is needed (or UDS is not available, such as in Windows), then the RPC address can be specified as `ip:port` or `host:port` instead of a UDS path. In this case, the volume does not have to be shared with the app, but it does need to be created for the `sigsci-agent` container to have a place to write temporary data such as geodata.

Adding the `sigsci-agent` container as a sidecar:

```
...
      containers:
      # Example helloworld app running on port 8000 against sigsci-agent via UDP /sigsci/tmp/sigsci.sock
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
          # Address for the app to listen on
          - localhost:8000
          # Address sigsci-agent RPC is listening on
          - /sigsci/tmp/sigsci.sock
        ports:
        - containerPort: 8000
        volumeMounts:
        # Shared mount with sigsci-agent container where the socket is shared via emptyDir volume
        - name: sigsci-tmp
          mountPath: /sigsci/tmp
      # Signal Sciences Agent running in default RPC mode
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: Always
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: secretaccesskey
        # If required (default is /sigsci/tmp/sigsci.sock for the container)
        #- name: SIGSCI_RPC_ADDRESS
        #  value: /path/to/socket for UDS OR host:port if TCP
        securityContext:
          # The sigsci-agent container should run with its root filesystem read only
          readOnlyRootFilesystem: true
        volumeMounts:
        # Default volume mount location for sigsci-agent writeable data
        # NOTE: Also change `SIGSCI_SHARED_CACHE_DIR` (default `/sigsci/tmp/cache`)
        #       if mountPath is changed, but best not to change.
        - name: sigsci-tmp
          mountPath: /sigsci/tmp
```

> **Note:** The above `sigsci-agent` configuration assumes that `sigsci` secrets were added to the system section above.

Adding the Signal Sciences agent temp volume definition to the deployment

```
...
      volumes:
      # Define a volume where sigsci-agent will write temp data and share the socket file,
      # which is required with the root filesystem is mounted read only
      - name: sigsci-tmp
        emptyDir: {}
```

Ubuntu NGINX 1.9 or lower

Add the package repositories

Add the version of the Ubuntu package repository that you want to use.

Ubuntu 22.04 - jammy

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/ubuntu/ jammy ma
sudo apt-get update
```

Ubuntu 20.04 - focal

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ focal main" | sudo tee /etc/apt/sources.list.d/sigs
```

Ubuntu 18.04 - bionic

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ bionic main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 16.04 - xenial

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ xenial main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 14.04 - trusty

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ trusty main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 12.04 - precise

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ precise main" | sudo tee /etc/apt/sources.list.d/si
```

Enable Lua for NGINX

Some older versions of NGINX don't support native loading of Lua modules. Therefore, we require NGINX to be built with the third party `ngx_lua` module. Because most older versions of NGINX do not support dynamically loadable modules, you will likely need to rebuild NGINX from source.

To assist you, we provide pre-built drop-in replacement NGINX packages already built with the `ngx_lua` module. This is intended for users who prefer not to build from source, or who either use a distribution-provided package or an official NGINX provided package. These pre-built packages are built to support much older distributions and are not gpg signed.

Flavors

Our provided flavors are:

- **Distribution:** the distribution flavor is based off the official distribution-provided NGINX packages. For Debian-based Linux distributions (Red Hat and Debian) these are the based off the official Debian NGINX packages.
- **Stable:** the stable flavor is based off the official NGINX.org stable package releases.
- **Mainline:** the mainline flavor is based off the official NGINX.org mainline package releases.

Flavor version support

The following versions are contained in the various OS and flavor packages:

OS	Distribution	Stable	Mainline
Ubuntu 22.04 (Jammy)	1.18.0	N/A	N/A
Ubuntu 20.04 (Focal)	1.18.0	N/A	N/A
Ubuntu 18.04 (Bionic)	1.14.0	N/A	N/A
Ubuntu 16.04 (Xenial)	1.10.3	N/A	N/A
Ubuntu 15.04 (Vivid)	1.6.2	1.8.1	1.9.10
Ubuntu 14.04 (Trusty)	1.4.6	1.8.1	1.9.10
Ubuntu 12.04 (Precise)	1.1.19	1.8.1	1.9.10

The versions are dependent on the upstream package maintainer’s supported version.

> **Note:** We do not provide a NGINX build for Ubuntu 16.04 and higher since Lua is supported. We only provide our dynamic Lua support modules for those versions.

Apt repository setup for Ubuntu systems

1. Add the repository key:

   ```
   wget -qO - https://apt.signalsciences.net/nginx/gpg.key | sudo apt-key add -
   ```

2. Create a new file `/etc/apt/sources.list.d/sigsci-nginx.list` with the following content based on your OS distribution and preferred flavor:

   - Distribution flavor

OS	`sigsci-nginx.list` content
Ubuntu 15.04 (Vivid)	`deb https://apt.signalsciences.net/nginx/distro vivid main`
Ubuntu 14.04 (Trusty)	`deb https://apt.signalsciences.net/nginx/distro trusty main`
Ubuntu 12.04 (Precise)	`deb https://apt.signalsciences.net/nginx/distro precise main`

   - Stable flavor

OS	`sigsci-nginx.list` content
Ubuntu 15.04 (Vivid)	`deb https://apt.signalsciences.net/nginx/stable vivid main`
Ubuntu 14.04 (Trusty)	`deb https://apt.signalsciences.net/nginx/stable trusty main`
Ubuntu 12.04 (Precise)	`deb https://apt.signalsciences.net/nginx/stable precise main`

   - Mainline flavor

OS	`sigsci-nginx.list` content
Ubuntu 15.04 (Vivid)	`deb https://apt.signalsciences.net/nginx/mainline vivid main`
Ubuntu 14.04 (Trusty)	`deb https://apt.signalsciences.net/nginx/mainline trusty main`
Ubuntu 12.04 (Precise)	`deb https://apt.signalsciences.net/nginx/mainline precise main`

3. Update the `apt` caches.

   ```
   apt-get update
   ```

4. Uninstall the default NGINX.

   ```
   sudo apt-get remove nginx nginx-common nginx-full
   ```

5. Install the version of NGINX provided by Signal Sciences.

Check Lua is loaded correctly

To verify Lua has been loaded properly load the following config (`sigsci_check_lua.conf`) with NGINX:

```
# Config just to test for lua jit support
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
# Also, for some nginx-1.10.nn installed from nginx-extras package, you may
# need to specify the load directives.
# Given the above uncomment the following:
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768;
    # multi_accept on;
}
http {
init_by_lua '

local m = {}
local ngx_lua_version = "dev"

if ngx then
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
if not r then
  error("ERROR: No lua jit support: No support for SigSci Lua module")
else

  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
  end

end

';

}
```

If the config is successfully loaded, the above script will create the following output:

```
nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
nginx: the configuration file <your explicit path>/sigsci_check_lua.conf syntax is ok
nginx: configuration file <your explicit path>/sigsci_check_lua.conf test is successful
```

Install the NGINX module

1. Install the module.

   ```
   apt-get install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX Service to initialize the new module.

   - Ubuntu 15.04 or higher

     ```
     sudo systemctl restart nginx
     ```

   - Ubuntu 14.04 or lower

     ```
     sudo restart nginx
     ```

HAProxy SPOE Module Install

Stream Processing Offload Engine (SPOE) enables HAProxy to send traffic to external programs for out-of-band processing. The HAProxy SPOE Module communicates with the Signal Sciences agent via SPOE, enabling the module to block requests using HAProxy Access Control Lists (ACLs) based on the agent response.

Requirements

- HAProxy 1.8 or higher
- Signal Sciences agent installed for your OS

Installation

Follow these steps to install the HAProxy SPOE module.

Download via package manager

The HAProxy SPOE module can be installed via the package manager of most major OS versions:

- Alpine: `sudo apk add sigsci-module-haproxy`
- CentOS: `sudo yum install sigsci-module-haproxy`
- Debian: `sudo apt-get install sigsci-module-haproxy`
- Ubuntu: `sudo apt-get install sigsci-module-haproxy`

Configure the agent

Add the following line to your agent configuration file (located by default at `/etc/sigsci/agent.conf`) to enable HAProxy SPOE support:

```
haproxy-spoa-enabled=true
```

Chroot directory configuration

> **Note:** This section may not be required for your installation. If you have set HAProxy's chroot directory, you will need to modify the commands below to reflect your custom chroot directory by following the instructions in this section.

If your HAProxy configuration has been modified to set a chroot directory for HAProxy, you will need to update your Signal Sciences agent configuration to reflect this. The default location of the agent socket file (`/var/run/sigsci.sock`) will be inaccessible to the HAProxy module outside of your specified chroot directory.

1. Create the directory structure for the Unix domain socket by running the following command, replacing `HAPROXY-CHROOT-DIRECTORY` with your HAProxy chroot directory:

location under chroot:

```
rpc-address="unix:/haproxy-chroot-directory/var/run/sigsci.sock"
```

Configure HAProxy

Follow these steps to configure HAProxy.

Add SPOA backend

Append the content of `/opt/signalsciences/haproxy-spoe/backend.txt` to your HAProxy configuration file:

```
sed "-i.`date +%F`" -e '$/opt/signalsciences/haproxy-spoe/backend.txt' /etc/haproxy/haproxy.cfg
```

Update frontend section

For HAProxy v2.2 and above, copy the content of `/opt/signalsciences/haproxy-spoe/frontend-2.2.txt` to each HTTP frontend section of your HAProxy configuration file:

```
sed -i -e '/frontend/r/opt/signalsciences/haproxy-spoe/frontend-2.2.txt' /etc/haproxy/haproxy.cfg
```

For HAProxy v1.8 and v2.0, copy the content of `/opt/signalsciences/haproxy-spoe/frontend-1.8.txt` to each HTTP frontend section of your HAProxy configuration file:

```
sed -i -e '/frontend/r/opt/signalsciences/haproxy-spoe/frontend-1.8.txt' /etc/haproxy/haproxy.cfg
```

Upgrading

To upgrade the HAProxy SPOE module:

1. Download and install the latest version of the module.
2. Configure the HAProxy module.
3. Restart HAProxy for the new module version to be detected.

Heroku Install

The Signal Sciences agent can be deployed with Heroku. The installation process is compatible with any of the language buildpacks.

Installation

1. Log in to Heroku.

   ```
   heroku login
   ```

2. Add the Signal Sciences buildpack to your application settings.

   ```
   heroku buildpacks:add --index 1 https://dl.signalsciences.net/sigsci-heroku-buildpack/sigsci-heroku-buildpack
   ```

   > **Note:** The Signal Sciences buildpack must run first or before your application’s primary buildpack.

3. In your `Procfile` file, add `sigsci/bin/sigsci-start` so it precedes your existing start command:

   ```
   web: sigsci/bin/sigsci-start YOUR-APPLICATION[redacted]S-START-COMMAND
   ```

   Example:

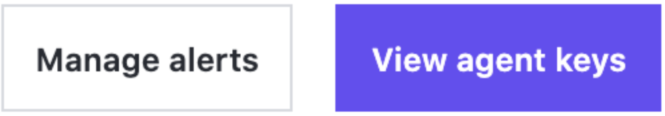
   ```
   web: sigsci/bin/sigsci-start node index.js
   ```

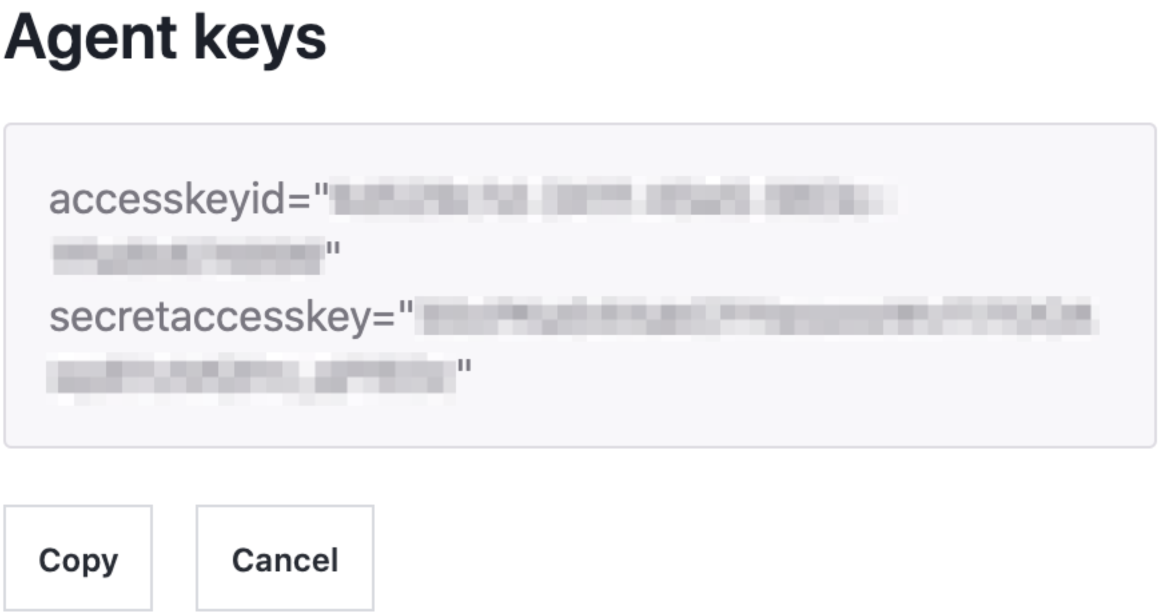
4. Locate the **Agent Keys** for your Signal Sciences site:

   1. Log in to the Signal Sciences console.

   2. From the **Sites** menu, select a site if you have more than one site.

   3. Click **Agents** in the navigation bar. The agents page appears.





Agent keys

```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

5. Add the Signal Sciences agent keys to your application's environment variables.

```
heroku config:set SIGSCI_ACCESSKEYID=access-key-goes-here
heroku config:set SIGSCI_SECRETACCESSKEY=secret-key-goes-here
```

6. Deploy your application. Heroku applications are typically deployed with the following commands:

```
git add .
git commit -m "my comment here"
git push heroku main
```

Configuration

- Each time you deploy your application, Heroku will automatically assign a new random name for the agent. An agent name for each deployment can be specified by setting the `SIGSCI_SERVER_HOSTNAME` environment variable:

  ```
  heroku config:set SIGSCI_SERVER_HOSTNAME=agent-name
  ```

- Agent access logging can be enabled by setting the `SIGSCI_REVERSE_PROXY_ACCESSLOG` environment variable:

  ```
  heroku config:set SIGSCI_REVERSE_PROXY_ACCESSLOG /tmp/sigsci_access.log
  ```

- The buildpack will install the latest version of the Signal Sciences agent by default. You can specify which agent version to install by setting the `SIGSCI_AGENT_VERSION` environment variable:

  ```
  heroku config:set SIGSCI_AGENT_VERSION=1.15.3
  ```

Additional configuration options are listed on the agent configuration page.

Debian Agent Installation

This guide explains how to install the Signal Sciences agent on Debian.

Prerequisites

Before you begin, determine the version of Debian you want to use.

Add the package repository

Begin the agent installation by adding the version of the Debian package repository that you want to use.

Debian 11 - Bullseye

To add the Debian 11 - Bullseye package, run the following script:

```
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ bullseye
sudo apt-get update
```

Debian 10 - Buster

To add the Debian 10 - Buster package, run the following script:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ buster ma
sudo apt-get update
```

Debian 9 - Stretch

To add the Debian 9 - Stretch package, run the following script:

```
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ stretch main
EOF
sudo apt-get update
```

Debian 8 - Jessie

To add the Debian 8 - Jessie package, run the following script:

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ jessie main
EOF
sudo apt-get update
```

Debian 7 - Wheezy

To add the Debian 7 - Wheezy package, run the following script:

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ wheezy main
EOF
sudo apt-get update
```

Install and configure the Signal Sciences Agent package

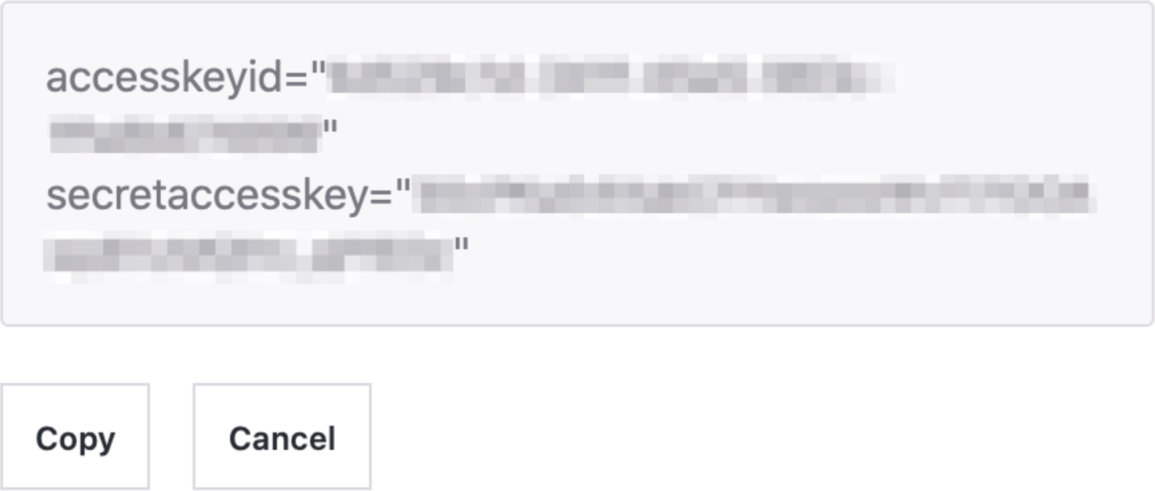
Now that you’ve downloaded the Debian package repository, you can install the Signal Sciences Agent package.

Run the following command to install the Signal Sciences Agent package.

```
sudo apt-get install sigsci-agent
```

Once the agent package is installed, you must create an agent configuration file and add the **Agent Access Key** and **Agent Secret Key**:

1. Create an empty agent configuration file in the following directory: `/etc/sigsci/agent.conf`.
2. Log in to the Signal Sciences console.
3. From the **Sites** menu, select the site that you want to give the agent access to.
4. Click the **Agents** link in the site navigation bar. The agents page appears.
5. Click the **View agent keys** button. The agent keys window appears.
6. Click the **Copy** button to copy the **Agent Access Key** and **Agent Secret Key** to your clipboard.



```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

7. Navigate to the agent configuration file and paste the **Agent Access Key** and **Agent Secret Key** into the file.

   ```
   accesskeyid = "AGENTACCESSKEYHERE"
   secretaccesskey = "AGENTSECRETACCESSKEYHERE"
   ```

8. Save the agent configuration file.

Start the Signal Sciences Agent

Now that you've installed and configured the agent package, you can start the Signal Sciences agent.

For Debian versions 8 and above, run the following command to start the Signal Sciences agent:

```
sudo systemctl start sigsci-agent
```

For Debian 7, run the following command to start the Signal Sciences agent:

```
sudo service sigsci-agent start
```

Optionally, enable the agent auto-update service. On a set schedule, the service checks the Signal Sciences package downloads site for a new version of the agent and updates the agent when a new version is available.

Next Steps

Explore our module options and install the Signal Sciences module.

Debian Apache Module Install

1. Install the Apache module.

   ```
   sudo apt-get install sigsci-module-apache
   ```

2. Add the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the **Dynamic Shared Object (DSO) Support** section to enable the Signal Sciences Apache module:

   ```
   LoadModule signalsciences_module /usr/lib/apache2/modules/mod_signalsciences.so
   ```

3. Restart the Apache web service.

   ```
   sudo service apache2 restart
   ```

Next Steps

Verify the agent and module installation and explore module options.

Search Syntax

Free Text

`/a/path/here sqli -7h`	Show all SQLI in last 7 hours with this particular path
`RU`	All recent requests from Russia
`cn 500`	All recent requests from China that had a 500 error
`404 233.252.0.23`	Recent requests from an IP that had a 404 error

Let us know if a free-text query did something you didn’t expect.

Explicit queries are made through the use of keys and operators. The previous sample queries can be made with keys and operators:

Free Text	Explicit Keys
`/a/path/here sqli -7h`	`path:/a/path/here sqli from:-7h`
`RU`	`country:ru`
`cn 500`	`country:cn httpcode:500`
`404 233.252.0.23`	`httpcode:404 ip:233.252.0.23`

Operators

- All values below can be quoted to allow for spaces.
- Adding – (minus) before any key negates the operation.
- Different key names function as an AND operator (`from:-1h path:/foo`).
- Multiple keys with the same name function as an OR operator (`path:/foo path:/bar` should return paths matching either `/foo` or `/bar`).

Operator	Meaning
`key:value`	equals
`key:=value`	equals, alternate syntax
`-key:value`	not equals, general negation of all operators
`key:!=value`	not equals, alternate syntax
`key:>value`	greater-than, integers only
`key:>=value`	equals or greater-than, integers only
`key:<value`	less-than, integers only
`key:<=value`	equals or less-than, integers only
`key:value1..value2`	in range between `value1` and `value2`, integers only. For time see `from` and `until`
`key:~value`	search on the field with the terms provided

Time

Time ranges can be specified in a number of ways using the `from` and `until` keys.

Queries on the Requests page of the console are limited to a maximum time range of 7 days. Queries greater than a 7 day period will not yield any results. For example, if you wanted to see results from 2 weeks ago, your query would need to use `from:-21d until:-14d`, which would be a 7 day window. A query of just `from:-21d` would not yield any results as that would be a 21 day window.

Relative time

Suffix	Meaning
`-5s`	5 seconds ago (from now)
`-5min`	5 minutes ago
`-5h`	5 hours ago
`-5d`	5 days ago
`-5w`	5 weeks ago
`-5mon`	5 months ago
`-5y`	5 year ago

Example:

- `from:-5h` (until now)
- `from:-5h until:-4h` (one hour range)

- Unix UTC Seconds Since Epoch
- Java/JavaScript UTC Milliseconds since Epoch
- ISO Date format `YYYYMMDD`

Example Absolute Time: Unix UTC Seconds

- `from:141384000` (until now)
- `from:141384000 until:1413844691`

Example Absolute Time: Java/JavaScript Milliseconds UTC

- `from:141384000000` (until now)
- `from:141384000000 until:1413844691000`

Example Absolute Date: `YYYYMMDD`

- `from:20141031` (until now)
- `from:20141031 until:20141225`

You can also mix and match time formats:

- `from:20141031 until:-1h`

Fields

Name	Type	Description
`agent`	string	The server hostname (or alias) for the agent (`agent:~hostname`, `agent:~appname`, `agent:hostname.appname`, or `agent:hostname-appname`)
`agentcode`	integer	The agents internal response code
`bytesout`	integer	HTTP response size in bytes
`country`	string	Request estimated country of origin (e.g., US, RU)
`from`	time	Filter output with requests since a particular date
`httpcode`	integer	The response's http response code
`ip`	string	Single IPv4 (`ip:198.51.100.128`) Single IPv6 (`ip:2001:0db8:1681:f16f:d4dc:a399:c00d:0225`) IPv4 CIDR (`ip:198.51.100.0/24`) IPv6 CIDR (`ip:2001:0db8:1681:f16f::/64`) IPv4 range (`ip:198.51.100.0..198.51.100.255`) IPv6 range (`ip:2001:0db8:1681:f16f::..2001:0db8:1681:f16f:ffff:ffff:ffff:ffff`)
`method`	string	HTTP Method (e.g., GET, POST)
`path`	string	Request URL path, does not include query parameters
`payload`	string	The data that triggered a signal (i.e., the attack value)
`protocol`	string	HTTP Request Protocol, typically HTTP/1.1 or HTTP/1.0
`ratelimited`	string	Requests that have been tagged with a specific threshold signal and have been rate limited. The search syntax is `ratelimited: site.<threshold-signal>`. You will need to replace `<threshold-signal>` with the name of the threshold signal that you want to search for.
`responsemillis`	integer	HTTP response time in milliseconds
`remotehost`	string	Remote hostname (`remotehost:www.example.com`) or subdomain match (`remotehost:~example.com`)
`server`	string	Requested server name in the http request (e.g., `example.com` if `http://example.com/name`)
`tag`	string	A particular signal on a request (e.g., SQLI, XSS)
`target`	string	Server + Path
`sort`	string	Sort with `time-asc` (oldest first) or `time-desc` (most recent first)
`until`	time	Filter output with request before a particular date
`useragent`	string	The request's user agent (browser)

policies, including United States Department of Defense–style mandatory access controls (MAC).

All official CentOS Linux builds come pre-configured with SELinux enabled and set to enforcement mode. There are two approaches to running the agent on a system with SELinux enabled:

- Set SELinux to Permissive mode or disable SELinux completely
- Configure SELinux to allow the module and agent to communicate

Determine if SELinux is enabled in enforcement mode

System administrators may not be aware that SELinux is installed until they encounter an error similar to the following when trying to connect the module to the agent:

```
2016/05/11 22:16:29 [crit] 3193#3193: *10 connect()
to unix:/var/run/sigsci.sock failed
(13: Permission denied), client: 192.0.2.209,
server: localhost, request: "GET /ping HTTP/1.1",
host: "192.0.2.209"
```

To check the status of SELinux, run the command `sestatus`, which produces output similar to the following:

```
[centos@ip-10-95-21-104 nginx]$ sestatus
SELinux status: enabled
SELinuxfs mount: /sys/fs/selinux
SELinux root directory: /etc/selinux
Loaded policy name: targeted
Current mode: enforcing
Mode from config file: enforcing
Policy MLS status: enabled
Policy deny_unknown status: allowed
Max kernel policy version: 28
```

Set SELinux to Permissive mode or disable SELinux completely

The main configuration file for SELinux is `/etc/selinux/config`. Run the following command to view its contents:

```
cat /etc/selinux/config
```

The output will look something like this:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# minimum - Modification of targeted policy. Only selected processes are protected.
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

You want to either disable or switch to permissive (logging) mode. A conservative first step may be changing the configuration line to `SELINUX=permissive` if you want to preserve the logging. You will then need to reboot the system entirely for this change to be applied and then verify the new status for SELinux with another `sestatus` command.

Configure SELinux to allow the module and agent to communicate

Assuming the system has SELinux in permissive or enforced mode and assuming the SELinux writes to the `/var/log/audit/audit.log` file (other Unix flavors potentially write it elsewhere):

1. Log in as root to install the SigSci agent and module.

2. Restart the web server and start the agent.

   - If in enforced mode, the same log messages will be appended to the audit log.
4. From your home directory, run the following command to create a `.te` file and a `.pp` (policy package) file: `cat /var/log/audit/audit.log | audit2allow -M sigsci > sigsci.te`.
5. Install the policy package file with `semodule -i sigscilua.pp`.
6. Verify the policy was installed and loaded by running the following command: `semodule -l`. The output will look something like this:

```
## Policy definition for SigSci Agent package on Rocky Linux 8
## Use make sigsci.pp (with a link to the SELinux policy devel Makefile)
## Requires policycoreutils-devel package
## make -f /usr/share/selinux/devel/Makefile sigsci.pp
## to create a module. Then run semodule -i sigsci.pp to install it

policy_module(sigsci, 1.0)
require {
type httpd_t;
}
#============= httpd_t =============
files_write_generic_pid_sockets(httpd_t)
```

7. Restart the web server and Signal Sciences agent and it should be working properly.

Installing the Java Module with Dropwizard

The Signal Sciences Java module can be deployed through Dropwizard.

Download

Download the Signal Sciences Java module manually or access it with Maven.

Download manually

1. Download the Java module archive from https://dl.signalsciences.net/sigsci-module-java/sigsci-module-java_latest.tar.gz.
2. Extract `sigsci-module-java_latest.tar.gz`.
3. Deploy the jars using one of the following options:
   - Copy `sigsci-module-java-{version}-shaded.jar` (an uber jar with all the dependencies bundled) to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`).
   - Copy `sigsci-module-java-{version}.jar` and its dependencies in the `lib` folder to your application's classpath (e.g., `%CATALINA_HOME%\webbapps\<APP_FOLDER>\WEB-INF\lib`). If you already have any of the dependency jar files in your application classpath folder (i.e., for Tomcat in the `WEB-INF\lib`) then it is not necessary to copy them, even if the version numbers are different. The logging jars are optional based on how `slf4j` is configured.

Access with Maven

For projects using Maven for build or deployment, the latest version of Signal Sciences Java modules can be installed by adding XML to the project `pom.xml` file. For example:

```
<repositories>
   <repository>
      <id>sigsci-stable</id>
      <url>https://packages.signalsciences.net/release/maven2</url>
   </repository>
</repositories>

<dependency>
   <groupId>com.signalsciences</groupId>
   <artifactId>sigsci-module-java</artifactId>
   <version>LATEST_MODULE_VERSION</version>
</dependency>
```

Install and configure

Dropwizard supports standard Java servlet filters, but you will need to register the filter class.

Additional information about Dropwizard servlet filter support can be found in the Dropwizard documentation.

The Dropwizard framework internally uses the Jetty servlet engine. The Signal Sciences Java module provides `servlet filters`.

Example run method inside class extending Dropwizard Application class

```
import com.signalsciences.servlet.filter.SigSciFilter;
@Override
public void run(final DwizExampleConfiguration configuration, final Environment environment) {
    environment.servlets().addFilter("SigSciFilter", new SigSciFilter()).addMappingForUrlPatterns(EnumSet.of(Dispa
    final HelloWorldResource resource = new HelloWorldResource(
        "%s",
        "Demo value"
    );
    environment.jersey().register(resource);
}
```

Kubernetes Agent + Ingress Controller + Module

Introduction

In this example, the Signal Sciences agent is installed as a Docker sidecar, communicating with a Signal Sciences native module for NGINX installed on an `ingress-nginx` Kubernetes ingress controller.

Integrating the Signal Sciences agent into an ingress controller

In addition to installing Signal Sciences per application, it is also possible to install Signal Sciences into a Kubernetes ingress controller that will receive all external traffic to your applications. Doing this is similar to installing into an application with a Signal Sciences module:

- Install and configure the Signal Sciences Module into the ingress controller.
- Add the `sigsci-agent` container to the ingress pod and mount a sigsci-agent volume.
- Add an `emptyDir{}` volume as a place for the `sigsci-agent` to write temporary data.

Kubernetes NGINX ingress controller

The Kubernetes NGINX Ingress Controller is an NGINX based implementation for the ingress API. Signal Sciences supports a native module for NGINX. This enables you to easily wrap the existing `ingress-nginx` controller to install the Signal Sciences module.

Wrap the base `nginx-ingress-controller` to install the Signal Sciences module

Wrapping the `nginx-ingress-controller` is done by using the base controller and installing the Signal Sciences native NGINX module. An example can be found here and here

A prebuilt container can be pulled from Docker Hub with: `docker pull signalsciences/sigsci-nginx-ingress-controller:latest`

Installation

There are two methods for installing:

- Install via Helm Using Overrides
- Install with Custom File

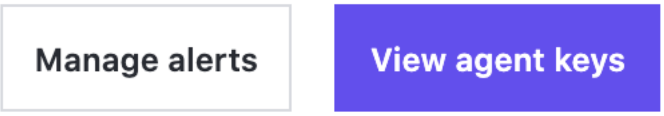
Install via Helm using overrides

The following steps cover installing `sigsci-nginx-ingress-controller` + `sigsci-agent` via the official `ingress-nginx` charts with an override file.

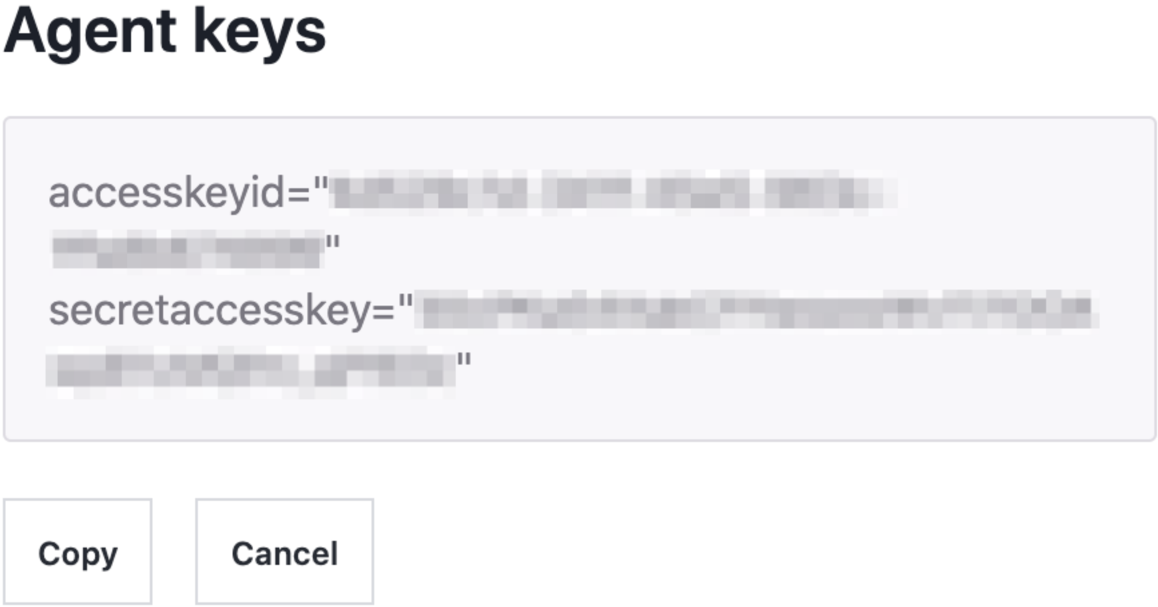
1. Add the `ingress-nginx` repository:

   ```
   helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
   ```

2. From the **Sites** menu, select a site if you have more than one site.
3. Click **Agents** in the navigation bar. The agents page appears.



4. Click **View agent keys**. The agent keys window appears.
5. Copy the **Agent Access Key** and **Agent Secret Key**.



```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

3. In the sigsci-values.yaml file, add the **Agent Keys** as `SIGSCI_ACCESSKEYID` and `SIGSCI_SECRETACCESSKEY`.
4. Install with the release name `my-ingress` in the `default` namespace:

   ```
   helm install -f values-sigsci.yaml my-ingress ingress-nginx/ingress-nginx
   ```

   You can specify a namespace with `-n` flag:

   ```
   helm install -n NAMESPACE -f values-sigsci.yaml my-ingress ingress-nginx/ingress-nginx
   ```

5. After a few minutes, the agent will be listed in your Signal Sciences console.
6. Create an Ingress resource. This step will vary depending on setup and supports a lot of configurations. Official documentation can be found regarding Basic usage - host based routing.

   Here is an example Ingress file:

   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   metadata:
     annotations:
       kubernetes.io/ingress.class: nginx
       nginx.ingress.kubernetes.io/rewrite-target: /
     name: hello-kubernetes-ingress
     #namespace: SET THIS IF NOT IN DEFAULT NAMESPACE
   spec:
     rules:
     - host: example.com
       http:
   ```

```
          backend:
            service:
              name: NAME OF SERVICE
              port:
                number: 80
```

Helm upgrade with override file

1. In the sigsci-values.yaml file, update the `sigsci-nginx-ingress-controller` to the latest version to update the `ingress-nginx` charts:

   ```
   controller:
       # Replaces the default nginx-controller image with a custom image that contains the Signal Sciences Nginx
       image:
         repository: signalsciences/sigsci-nginx-ingress-controller
         tag: "latest"
         pullPolicy: IfNotPresent
   ```

2. Run `helm upgrade` with the override file. This example is running helm upgrade against the `my-ingress` release created in the previous section:

   ```
   helm upgrade -f sigsci-values.yaml my-ingress ingress-nginx/ingress-nginx
   ```

   or

   ```
   helm upgrade -f sigsci-nginxinc-values.yaml my-ingress ingress-nginx/ingress-nginx
   ```

   If ingress is not in default namespace, use `-n` to specify namespace:

   ```
   helm upgrade -n NAMESPACE -f sigsci-values.yaml my-ingress ingress-nginx/ingress-nginx
   ```

   or

   ```
   helm upgrade -n NAMESPACE -f sigsci-nginxinc-values.yaml my-ingress ingress-nginx/ingress-nginx
   ```

Uninstall release

1. Uninstall release `my-ingress`.

   ```
   helm uninstall my-ingress
   ```

2. If it's not in the default namespace, use `-n` to specify the namespace:

   ```
   helm uninstall -n NAMESPACE my-ingress
   ```

Install with custom file

Integrating the Signal Sciences Agent

The Signal Sciences Agent can be installed as a sidecar into each pod or as a service for some specialized needs.

The recommended way of installing the Signal Sciences Agent in Kubernetes is by integrating the `sigsci-agent` into a pod as a sidecar. This means adding the `sigsci-agent` as an additional container to the Kubernetes pod. As a sidecar, the agent will scale with the app/service in the pod instead of having to do this separately. However, in some situations, it may make more sense to install the `sigsci-agent` container as a service and scale it separately from the application.

The `sigsci-agent` container can be configured in various ways depending on the installation type and module being used.

You can use the `preStop` container hook to slow the pod's shutdown and ensure drain timeouts are met.

```
preStop:
  exec:
    command:
      - sleep
      - "30"
```

Alternatively, if you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

These instructions reference the `latest` version of the agent with `imagePullPolicy: Always`, which will pull the latest agent version even if one already exist locally. This is so the documentation does not fall out of date and anyone using this will not have an agent that stays stagnant. However, this may not be what if you need to keep installations consistent or on a specific version of the agent. In these cases, you should specify an agent version. Images on Docker Hub are tagged with their versions and a list of versions is available on Docker Hub.

Whether you choose to use the `latest` image or a specific version, there are a few items to consider to keep the agent up-to-date:

Using the `latest` Signal Sciences Container Image

If you do choose to use the `latest` image, then you will want to consider how you will keep the agent up to date.

- If you have used the `imagePullPolicy: Always` option, then the latest image will be pulled on each startup and your agent will continue to get updates.
- Alternatively, you may instead choose to manually update the local cache by periodically forcing a pull instead of always pulling on startup:

  ```
  docker pull signalsciences/sigsci-agent:latest
  ```

  Then, use `latest` with `imagePullPolicy: Never` set in the configuration so that pulls are never done on startup (only manually as above):

  ```
  - name: sigsci-agent
      image: signalsciences/sigsci-agent:latest
      imagePullPolicy: Never
      ...
  ```

Using a Versioned Signal Sciences Container Image

To use a specific version of the agent, replace `latest` with the agent version (represented here by `x.xx.x`). You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:x.xx.x
    imagePullPolicy: IfNotPresent
    ...
```

This will pull the specified agent version and cache it locally. If you use this method, then it is recommended that you parameterize the agent image, using Helm or similar, so that it is easier to update the agent images later on.

Using a Custom Tag for the Signal Sciences Container Image

It is also possible to apply a custom tag to a local agent image. To do this, pull the agent image (by version or use `latest`), apply a custom tag, then use that custom tag in the configuration. You will need to specify `imagePullPolicy: Never` so local images are only updated manually. After doing so, you will need to periodically update the local image to keep the agent up-to-date.

For example:

```
docker pull signalsciences/sigsci-agent:latest
docker tag signalsciences/sigsci-agent:latest signalsciences/sigsci-agent:testing
```

Then use this image tag in the configuration:

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:testing
    imagePullPolicy: Never
...
```

Configuring the Signal Sciences Agent Container

Agent configuration is normally done via the environment. Most configuration options are available as environment variables. Environment variables names have the configuration option name all capitalized, prefixed with `SIGSCI_` and any dashes (-) changed to underscores (_). For example, the max-procs option would become the `SIGSCI_MAX_PROCS` environment variable. For more details on what options are available, see the Agent Configuration documentation.

- A volume to write temporary files.

Agent Credentials

The `sigsci-agent` credentials are configured with two environment variables. These variables must be set or the agent will not start.

- **SIGSCI_ACCESSKEYID**: The **Agent Access Key** identifies which site in the Signal Sciences console that the agent is configured for.
- **SIGSCI_SECRETACCESSKEY**: The **Agent Secret Key** is the shared secret key to authenticate and authorize the agent.

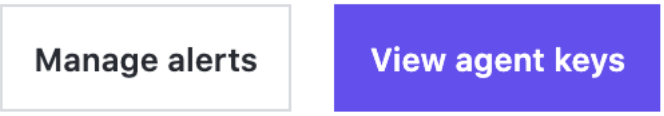
The credentials can be found by following these steps:

1. Log in to the Signal Sciences console.

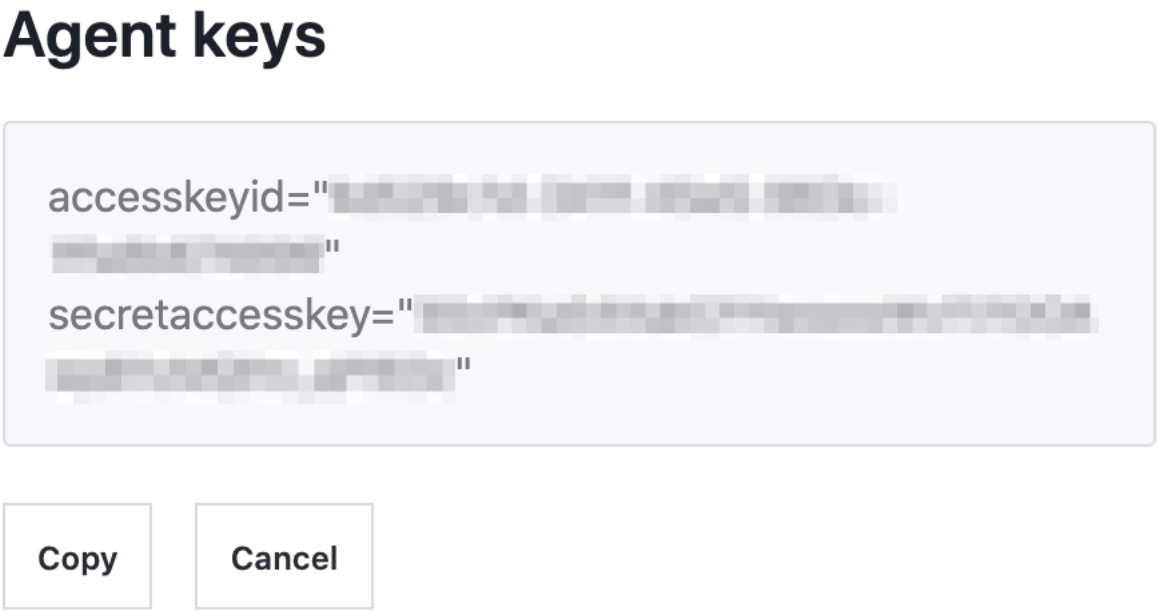
2. From the **Sites** menu, select a site if you have more than one site.

3. Click **Agents** in the navigation bar. The agents page appears.

4. Click **View agent keys**. The agent keys window appears.



5. Copy the **Agent Access Key** and **Agent Secret Key**.



Agent keys

accesskeyid="[redacted]"
secretaccesskey="[redacted]"

Because of the sensitive nature of these values, we recommend you use the built in `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded values into the deployment configuration. This also makes any desired agent credential rotation easier to manage by having to change them in only one place.

Use the `valueFrom` option instead of the `value` option to use the `secrets` functionality. For example:

```
env:
  - name: SIGSCI_ACCESSKEYID
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: accesskeyid
  - name: SIGSCI_SECRETACCESSKEY
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
```

The `secrets` functionality keeps secrets in various stores in Kubernetes. This guide uses the generic secret store in its examples, however any equivalent store can be used. Agent secrets can be added to the generic secret store using YAML similar to the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: sigsci.my-site-name-here
stringData:
  accesskeyid: 12345678-abcd-1234-abcd-1234567890ab
  secretaccesskey: abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

This can also be created from the command line with `kubectl` such as with the following example:

```
kubectl create secret generic sigsci.my-site-name-here \
  --from-literal=accesskeyid=12345678-abcd-1234-abcd-1234567890ab \
  --from-literal=secretaccesskey=abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

Additional information about Kubernetes `secrets` functionality can be found here.

Agent Temporary Volume

For added security, we recommended the `sigsci-agent` container be executed with the root filesystem mounted as read only. However, the agent still needs to write some temporary files such as the socket file for RPC communication and some periodically updated files such as GeoIP data.

To accomplish this with a read only root filesystem, there needs to be a writeable volume mounted. This writeable volume can also be shared to expose the RPC socket file to other containers in the same pod.

The recommended way of creating a writeable volume is to use the builtin `emptyDir` volume type. This is typically configured in the `volumes` section of a deployment, as shown in the following example:

```
volumes:
  - name: sigsci-tmp
    emptyDir: {}
```

Containers will then mount this volume at `/sigsci/tmp`:

```
volumeMounts:
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
```

The default in the official agent container image is to have the temporary volume mounted at `/sigsci/tmp`. If this needs to be moved for the agent container, then the following agent configuration options should also be changed from their defaults to match the new mount location:

- `rpc-address` defaults to `/sigsci/tmp/sigsci.sock`
- `shared-cache-dir` defaults to `/sigsci/tmp/cache`

The NGINX ingress controller is installed with the mandatory.yaml file. This file contains a modified template of the Generic Ingress Controller Deployment as described here. The main additions are:

1. Change the ingress container to load the custom Signal Sciences Module/ingress container and add Volume mounts for socket file communication between the Module/ingress container and Agent sidecar container:

```
...
    containers:
      - name: nginx-ingress-controller
        image: signalsciences/sigsci-nginx-ingress-controller:latest
        ...
        volumeMounts:
          - name: sigsci-tmp
            mountPath: /sigsci/tmp
...
```

```
apiVersion: v1
data:
  main-snippet: load_module /usr/lib/nginx/modules/ngx_http_sigsci_nxo_module-1.17.7.so;
  http-snippet: sigsci_agent_host unix:/sigsci/tmp/sigsci.sock;
metadata:
  name: nginx-configuration
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
```

3. Add a container for the Signal Sciences Agent:

```
...
    containers:
    ...
      # Signal Sciences Agent running in default RPC mode
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: IfNotPresent
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: secretaccesskey
        securityContext:
          # The sigsci-agent container should run with its root filesystem read only
          readOnlyRootFilesystem: true
        volumeMounts:
        # Default volume mount location for sigsci-agent writeable data (do not change mount path)
        - name: sigsci-tmp
          mountPath: /sigsci/tmp
...
```

4. Define the volume used above:

```
...
      volumes:
      # Define a volume where sigsci-agent will write temp data and share the socket file,
      # which is required with the root filesystem is mounted read only
      - name: sigsci-tmp
        emptyDir: {}
...
```

Setup

The `mandatory.yaml` file creates the resources in the `ingress-nginx` namespace. If using Kubernetes Secrets to store the agent access keys, you will need to create the namespace and access keys before running the mandatory.yaml file.

1. Set the name for the secrets for the agent keys in `mandatory.yaml`.

```
...
    env:
          # This secret needs added (see docs on sigsci secrets)
          name: sigsci.my-site-name-here
          key: accesskeyid
    - name: SIGSCI_SECRETACCESSKEY
      valueFrom:
        secretKeyRef:
          # This secret needs added (see docs on sigsci secrets)
          name: sigsci.my-site-name-here
          key: secretaccesskey
...
```

2. Pull or build the **NGINX ingress + Signal Sciences Module** container. Set any preferred registry and repository name, and set the image to match in `mandatory.yaml`:

   ```
   docker pull signalsciences/sigsci-nginx-ingress-controller:latest
   ```

3. Deploy using modified Generic Deployment:

   ```
   kubectl apply -f mandatory.yaml
   ```

4. Create the service to expose the Ingress Controller. The steps necessary are dependent on your cloud provider. Official instructions can be found at https://kubernetes.github.io/ingress-nginx/deploy/#environment-specific-instructions.

   Below is an example `service.yaml` file:

   ```
   kind: Service
   apiVersion: v1
   metadata:
     name: ingress-nginx
     namespace: ingress-nginx
   spec:
     externalTrafficPolicy: Cluster
     selector:
       app.kubernetes.io/name: ingress-nginx
     type: LoadBalancer
     ports:
       - name: http
         port: 80
         targetPort: http
       - name: https
         port: 443
         targetPort: https
   ```

5. Create the Ingress Resource. Below is an example Ingress Resource:

   ```
   apiVersion: extensions/v1
   kind: Ingress
   metadata:
     name: test-ingress
     namespace: ingress-nginx
     annotations:
       nginx.ingress.kubernetes.io/rewrite-target: /
   spec:
     rules:
     - http:
         paths:
         - path: /testpath
           backend:
             serviceName: nginx
             servicePort: 80
   ```

Add the package repositories

Add the version of the Ubuntu package repository that you want to use.

Ubuntu 22.04 - jammy

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/ubuntu/ jammy ma
sudo apt-get update
```

Ubuntu 20.04 - focal

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ focal main" | sudo tee /etc/apt/sources.list.d/sigs
```

Ubuntu 18.04 - bionic

```
sudo apt update
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ bionic main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 16.04 - xenial

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ xenial main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 14.04 - trusty

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ trusty main" | sudo tee /etc/apt/sources.list.d/sig
```

Ubuntu 12.04 - precise

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo echo "deb https://apt.signalsciences.net/release/ubuntu/ precise main" | sudo tee /etc/apt/sources.list.d/si
```

Install the NGINX module

1. Install the Signal Sciences NGINX module that supports the NGINX version that you want to use:

   - NGINX Plus 29

     ```
     sudo apt-get install nginx-module-sigsci-nxp=1.23.4*
     ```

   - NGINX Plus 28

     ```
     sudo apt-get install nginx-module-sigsci-nxp=1.23.2*
     ```

   - NGINX Plus 27

     ```
     sudo apt-get install nginx-module-sigsci-nxp=1.21.6*
     ```

   - NGINX Plus 26

     ```
     sudo apt-get install nginx-module-sigsci-nxp=1.21.5*
     ```

   - NGINX Plus 25

     ```
     sudo apt-get install nginx-module-sigsci-nxp=1.21.3*
     ```

   - NGINX Plus 24

```
sudo apt-get install nginx-module-sigsci-nxp=1.19.5*
```

- NGINX Plus 22

```
sudo apt-get install nginx-module-sigsci-nxp=1.19.0*
```

- NGINX Plus 21

```
sudo apt-get install nginx-module-sigsci-nxp=1.17.9*
```

- NGINX Plus 20

```
sudo apt-get install nginx-module-sigsci-nxp=1.17.6*
```

- NGINX Plus 19

```
sudo apt-get install nginx-module-sigsci-nxp=1.17.3*
```

- NGINX Plus 18

```
sudo apt-get install nginx-module-sigsci-nxp=1.15.10*
```

- NGINX Plus 17

```
sudo apt-get install nginx-module-sigsci-nxp=1.15.7*
```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the `pid /run/nginx.pid;` line:

```
load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
```

3. Restart the NGINX service to initialize the new module.

```
sudo service nginx restart
```

Amazon Linux Agent Installation

This guide explains how to install the Signal Sciences agent on Amazon Linux.

Prerequisites

Before you begin, determine the version of Amazon Linux you want to use: Amazon Linux 2023, Amazon Linux 2, or Amazon Linux 2015.09.01. Amazon Linux 2 is most similar to CentOS 7 and reuses the same configuration. Amazon Linux 2015.09.01 is most similar to CentOS 6 and reuses the same configuration. Amazon Linux 2023 does not mirror CentOS as closely as it did in the past. It is a combination of multiple versions of Fedora and CentOS Stream 9. See Relationship to Fedora. As a result, the `baseurl` of the yum repository will use `amazon/2023` as the distribution name and version number for Amazon Linux 2023.

Add the package repository

Begin the agent installation by adding the version of the Amazon Linux package repository that you want to use. Add the version of the Amazon Linux package repository that you want to use.

Amazon Linux 2023

To add the Amazon Linux 2023 package, run the following script:

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/amazon/2023/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Amazon Linux 2015.09.01

To add the Amazon Linux 2015.09.01 package, run the following script:

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Install and configure the Signal Sciences Agent package

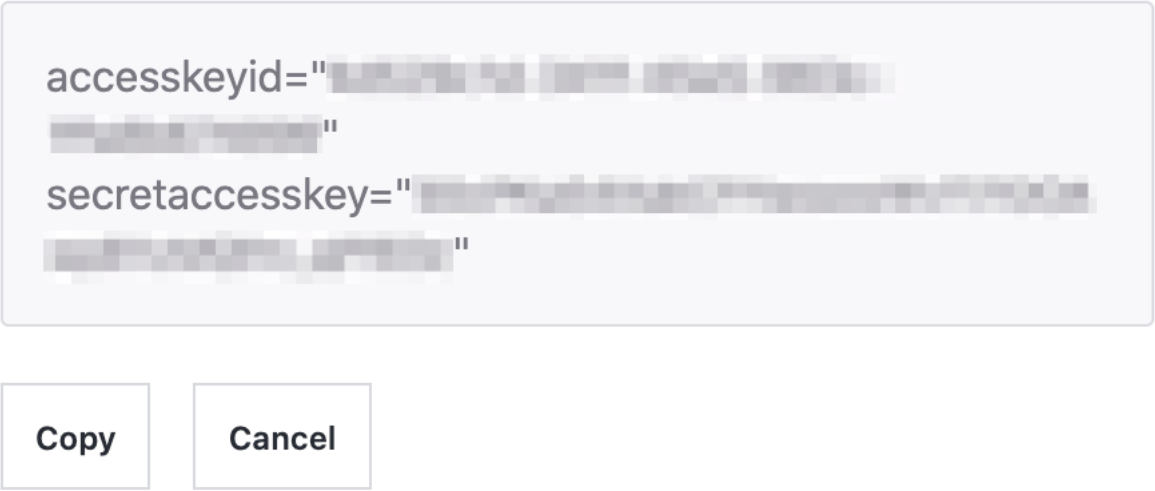
Now that you’ve downloaded the Amazon Linux package repository, you can install the Signal Sciences Agent package.

Run the following command to install the Signal Sciences Agent package.

```
sudo yum install sigsci-agent
```

Once the agent package is installed, you must create an agent configuration file and add the **Agent Access Key** and **Agent Secret Key**:

1. Create an empty agent configuration file in the following directory: `/etc/sigsci/agent.conf`.
2. Log in to the Signal Sciences console.
3. From the **Sites** menu, select the site that you want to give the agent access to.
4. Click the **Agents** link in the site navigation bar. The agents page appears.
5. Click the **View agent keys** button. The agent keys window appears.
6. Click the **Copy** button to copy the **Agent Access Key** and **Agent Secret Key** to your clipboard.



```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

7. Navigate to the agent configuration file and paste the **Agent Access Key** and **Agent Secret Key** into the file.

   ```
   accesskeyid = "AGENTACCESSKEYHERE"
   secretaccesskey = "AGENTSECRETACCESSKEYHERE"
   ```

8. Save the agent configuration file.

Start the Signal Sciences Agent

Now that you've installed and configured the agent package, you can start the Signal Sciences agent.

If you added the Amazon Linux 2 package, run the following command to start the Signal Sciences agent:

```
sudo systemctl start sigsci-agent
```

If you added the Amazon Linux 2015.09.01 package, run the following command to start the Signal Sciences agent:

```
start sigsci-agent
```

Next Steps

Explore our module options and install the Signal Sciences module.

Amazon Linux Apache Module Install

The Signal Sciences Apache module supports Amazon Linux 2015.09.01 or higher.

1. Install the Signal Sciences Apache Module.

   - Amazon Linux 2

     ```
     sudo yum install sigsci-module-apache
     ```

   - Amazon Linux 2015.09.01 with Apache 2.4

     ```
     sudo yum install sigsci-module-apache24
     ```

   - Amazon Linux 2015.09.01 with Apache 2.2

     ```
     sudo yum install sigsci-module-apache
     ```

2. Add the following line to your Apache configuration after the **Dynamic Shared Object (DSO) Support** section to enable the Signal Sciences Apache module:

   ```
   LoadModule signalsciences_module /etc/httpd/modules/mod_signalsciences.so
   ```

3. Restart Apache.

   - Amazon Linux 2

```
sudo service httpd restart
```

Next Steps

Verify the agent and module installation and explore module options.

Kong Plugin Install

About the Kong plugin

The Kong plugin is a feature of the NGINX module, which allows it to function as a Kong plugin. Accordingly, the process for installing the Kong plugin involves installing the Signal Sciences agent and NGINX module, and modifying the NGINX module configuration to enable it for use with Kong.

Installation

1. Install the Signal Sciences agent for your environment.
2. Edit the agent configuration file located at `/etc/sigsci/agent.conf` to add the following lines. Replace `<AGENT-LISTENER-IP>` with the host IP address (usually `127.0.0.1`) and `<AGENT-LISTENER-PORT>` with the TCP port on which the agent will listen for connections from the module. There is no default, but we suggest port `737` to minimize the chance of conflicts with other services:

   ```
   rpc-address=<AGENT-LISTENER-IP>:<AGENT-LISTENER-PORT>
   ```

3. Download and extract the latest Signal Sciences NGINX module.

   ```
   curl -O  https://dl.signalsciences.net/sigsci-module-nginx/sigsci-module-nginx_latest.tar.gz
   sudo mkdir -p /opt/sigsci/nginx
   sudo tar -xf sigsci-module-nginx_latest.tar.gz -C /opt/sigsci/nginx
   ```

4. If you are on Kong 3.0.x, override the `handler.lua` and `schema.lua` files in `/opt/sigsci/nginx/sigsci-module-nginx/kong/plugins/signalsciences` with the `handler.lua` and `schema.lua` files in `/opt/sigsci/sigsci-module-nginx/nginx/kong/plugins/signalsciences-3.0.x`.
5. Edit the following lines in `/opt/sigsci/nginx/sigsci-module-nginx/kong/plugins/signalsciences/handler.lua` to reflect the host IP address and the port used for communication with the agent. Replace "`localhost`" and `12345` with the host IP address and port:

   ```
   sigsci.agenthost = "localhost"
   sigsci.agentport = 12345
   ```

6. In the Kong configuration file at `/etc/kong/kong.conf`, add the following lines:

   ```
   plugins=signalsciences
   lua_package_path=/opt/sigsci/nginx/sigsci-module-nginx/?.lua
   ```

7. Enable the Kong plugin by running the following command. Replace `<KONG-GATEWAY-IP:PORT>` with the Kong IP address and port (for example, `127.0.0.1:1234`):

   ```
   curl -i -X POST --url http://<KONG-GATEWAY-IP:PORT>/plugins/ --data 'name=signalsciences'
   ```

IBM Cloud Install

The Signal Sciences agent can be deployed with IBM Cloud application runtimes. The installation process is compatible with any of the language buildpacks.

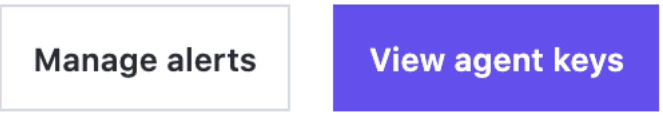
This is a supply-buildpack for Cloud Foundry that provides integration with the Signal Sciences agent for any programming language supported by the platform, and requiring zero application code changes.

Installation

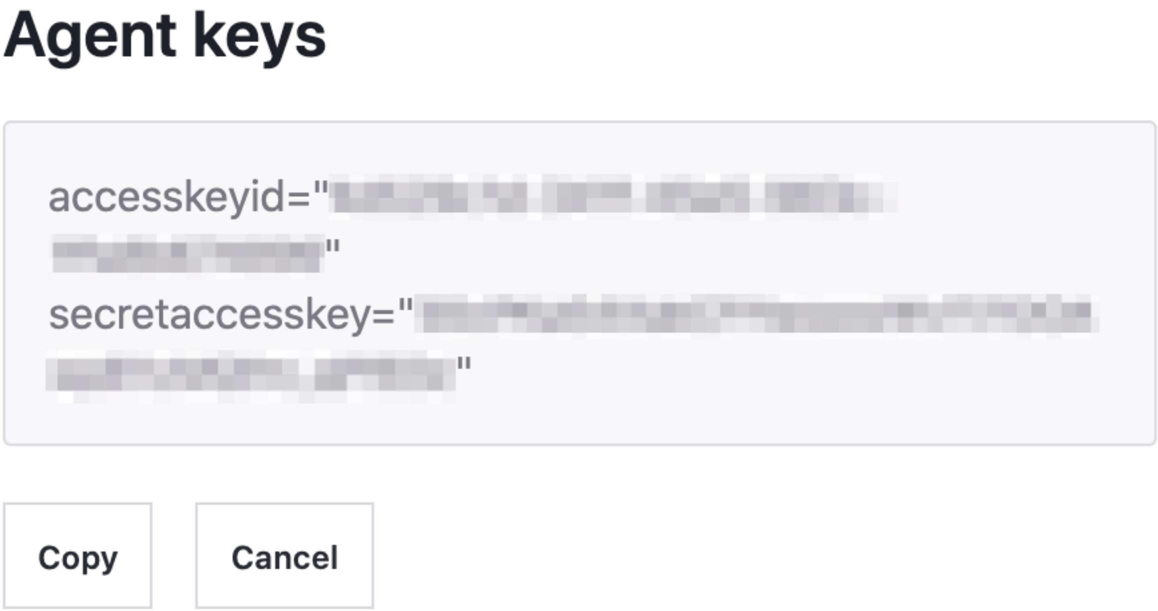
1. Application developers will need to specify the buildpack with the `cf push` command:

   ```
   cf push YOUR-APP -b https://github.com/signalsciences/sigsci-cloudfoundry-buildpack.git -b APP_BUILDPACK
   ```

   2. From the **Sites** menu, select a site if you have more than one site.
   3. Click **Agents** in the navigation bar. The agents page appears.



   4. Click **View agent keys**. The agent keys window appears.
   5. Copy the **Agent Access Key** and **Agent Secret Key**.



Agent keys

```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

3. Set your agent's access key and secret using the `cf set-env` command. Replace `your-application-name` with the name of your application and replace `access-key-goes-here` and `secret-key-goes-here` with your agent keys:

```
cf set-env your-application-name SIGSCI_ACCESSKEYID access-key-goes-here
cf set-env your-application-name SIGSCI_SECRETACCESSKEY secret-key-goes-here
```

4. Run `cf push` as you normally would to deploy your application.

Additional configuration options

The Signal Sciences agent can be configured with environment variables using the `cf command`, replacing `OPTION` and `VALUE` with the agent configuration option and its value:

```
cf set-env your-application-name OPTION "VALUE"
```

To have these changes take effect, you must at least re-stage your app:

```
cf restage your-application-name
```

Server hostname

Each time you deploy your application, IBM Cloud will automatically assign a new random name for the agent. To specify an agent name for each deployment, set the `SIGSCI_SERVER_HOSTNAME` environment variable:

```
cf set-env your-application-name SIGSCI_SERVER_HOSTNAME agent-name
```

Reverse proxy upstream

To define upstream hosts that the Agent will proxy requests to, use the `SIGSCI_REVERSE_PROXY_UPSTREAM` option, replacing `ip:port` with the upstream host IP address and port. This variable is optional with a default value of `127.0.0.1:8081`:

```
cf set-env your-application-name SIGSCI_REVERSE_PROXY_UPSTREAM ip:port
cf set-env your-application-name SIGSCI_REVERSE_PROXY_ACCESSLOG /tmp/sigsci_access.log
```

Agent version

By default the buildpack will install the latest version of the Signal Sciences agent. To specify which agent version to install, set the `SIGSCI_AGENT_VERSION` environment variable, replacing `version-number` with the specific version number to install:

```
cf set-env <application name> SIGSCI_AGENT_VERSION version-number
```

Health checks

Currently, IBM Cloud does not support HTTP health checks native to Cloud Foundry. If the application process crashes while the Signal Sciences agent is still running, IBM Cloud may not detect that the application is in an unhealthy state. The latest release of the Signal Sciences Cloud Foundry installer script can be configured to implement health checking that will stop the agent process if the application process is in an unhealthy state.

There are two environment variables that enable/configure health checking:

Set `SIGSCI_HC` to `true` to enable health checking:

```
cf set-env your-application-name SIGSCI_HC true
```

Set `SIGSCI_HC_CONFIG` to configure the health check. If you do not set this environment variable the default settings will be used.

The default settings configure the health check to:

- Check the `/` path every 5 seconds.
- If the agent listener returns a `502` for 5 sequential checks, then the health check fails.
- If the application process does not return a `200` response for 3 sequential tries, then the health check fails.

To specify custom health check settings, the `SIGSCI_HC_CONFIG` value is a string that consists of several fields delimited by `:`.

`SIGSCI_HC_CONFIG` fields:

```
<frequency>:<endpoint>:<listener status>:<listener warning>:<upstream status>:<upstream warning>
```

Field	Description
`frequency`	How often to perform the check in seconds (e.g., every 5 seconds)
`endpoint`	Which endpoint to check for both the listener and upstream process
`listener status`	The status code that not healthy and will trigger stopping the agent
`listener warning`	The number of times the check can fail before stopping the agent
`upstream status`	The status code that is healthy, any other code will trigger stopping the agent
`upstream warning`	The number of times the check can fail before stopping the agent

As an example, the default settings looks like:

```
5:/:502:5:200:3
```

Example custom health check settings

These example settings configure the health check to:

- Check the `/health.html` path every 10 seconds.
- If the agent listener returns a `502` for 10 sequential tries the health check fails.
- If the application process does not return a `200` for 5 sequential tries, the health check fails.

```
10:/health.html:502:10:200:5
```

Require agent

By default the installer script will allow the application to start even if the Signal Sciences agent fails to start. To ensure that your application never starts without being protected by the Signal Sciences agent, use the `SIGSCI_REQUIRED` environment variable:

```
cf set-env your-application-name SIGSCI_REQUIRED true
```

Additional configuration options

Installing the Java Module on Weblogic

Compatibility

The Signal Sciences Java module is compatible with WebLogic version 12c (12.2.1) or higher.

Installation

To deploy the Signal Sciences Java module on WebLogic servers, you must first add it to your application as a servlet filter.

Then, deploy your application to your WebLogic server through the same process you would deploy any other Web Application.

Module Configuration

Option	Default	Description
`rpcServerURI`	Required, `tcp://127.0.0.1:9999`	The Unix domain socket or TCP connection to communicate with the agent.
`rpcTimeout`	Required, 300ms	The timeout in milliseconds that the RPC client waits for a response back from the agent.
`maxResponseTime`	Optional, no default	The maximum time in seconds that the server response time will be evaluated against (i.e., to see if it exceeds this value) to determine if the module should send a post request to the agent.
`maxResponseSize`	Optional, no default	The maximum size in bytes that the server response size will be evaluated against (i.e. to see if it exceeds this value) to determine if the module should send a post request to the agent.
`maxPost`	Optional, no default	The maximum POST body size in bytes that can be sent to the Signal Sciences agent. For any POST body size exceeding this limit, the module will not send the request to the agent for detection.
`asyncStartFix`	Optional, false	This can be set to `true` to workaround missing request body when handling requests asynchronously in servlets.
`altResponseCodes`	Optional, no default	Space separated alternative agent response codes used to block the request in addition to 406. For example `403 429 503`.
`excludeCidrBlock`	Optional, no default	A comma-delimited list of CIDR blocks or specific IP addresses to be excluded from filter processing.
`excludeIpRange`	Optional, no default	A comma-delimited list of IP ranges or specific IP addresses to be excluded from filter processing.
`excludePath`	Optional, no default	A comma-delimited list of paths to be excluded from filter processing. If the URL starts with the specified value it will be excluded. Matching is case-insensitive.
`excludeHost`	Optional, no default	A comma-delimited list of host names to be excluded from filter processing. Matching is case-insensitive.

Sample module configuration:

Module configuration changes must be made in the `<!-- Signal Sciences Filter -->` section of your application's `web.xml` file:

```
<!-- Signal Sciences Filter -->
<filter>
    <filter-name>sigSciFilter</filter-name>
    <filter-class>com.signalsciences.servlet.filter.SigSciFilter</filter-class>
     <async-supported>true</async-supported>
<init-param>
   <param-name>rpcTimeout</param-name>
   <param-value>500</param-value>
</init-param>
  <init-param>
   <param-name>asyncStartFix</param-name>
   <param-value>true</param-value>
</init-param>
</filter>
<filter-mapping>
    <filter-name>sigSciFilter</filter-name>
```

Node.js Module Install

Compatibility

The Signal Sciences Node.js module is compatible with Node 0.10 through 12.X. All dependencies are specified in the `npm-shrinkwrap.json` file.

Installation

Install the latest version from npmjs.com:

```
npm install sigsci-module-nodejs
```

For specific releases prior to 1.5.3, installation can be performed from the release archive. Replace `<VERSION>` with the specific version number:

```
npm install https://dl.signalsciences.net/sigsci-module-nodejs/<VERSION>/sigsci-module-nodejs-<VERSION>.tgz
```

See the package archive for a list of available versions.

Usage

How to incorporate the Signal Sciences Node.js module will depend on your application.

Native applications

If your application invokes `http.createServer` directly, use the native API.

1. Above your application code, import the Signal Sciences Node.js module by adding the following lines:

   ```
   var Sigsci = require('sigsci-module-nodejs')

   // Your application code
   ```

2. Below your application code, create a `Sigsci` object:

   ```
   // Your application code

   var sigsci = new Sigsci({
    path: '/var/run/sigsci.sock'
    // Other parameters here
   })
   ```

3. Wrap the dispatcher with `sigsci.wrap`. Replace the `http.createServer(dispatcher).listen(8085, '127.0.0.1')` line with:

   ```
   http.createServer(sigsci.wrap(dispatcher)).listen(8085, '127.0.0.1')
   ```

Example

```
var Sigsci = require('sigsci-module-nodejs')

// Your application code

var sigsci = new Sigsci({
 path: '/var/run/sigsci.sock'
 // Other parameters here
})

http.createServer(sigsci.wrap(dispatcher)).listen(8085, '127.0.0.1')
```

Node.js Express

The Node.js Express module is exposed as Express middleware and is typically inserted as the first middleware, immediately below the `var app = express()` statement. See the Express Using Middleware documentation for more details.

```
// Your application code
```

2. Below your application code, create a `Sigsci` object:

```
// Your application code

var sigsci = new Sigsci({
path: '/var/run/sigsci.sock'
// other parameters here
})
```

3. Below the `var app = express()` line, insert the Node.js module middleware:

```
var app = express()
app.use(sigsci.express())

// You can still call other middleware and routes
app.use(...)
app.get('/route', ...)
```

Example

```
var Sigsci = require('sigsci-module-nodejs')

// Your application code

var sigsci = new Sigsci({
path: '/var/run/sigsci.sock'
// other parameters here
})

var app = express()
app.use(sigsci.express())

// You can still call other middleware and routes
app.use(...)
app.get('/route', ...)
```

Node.js Restify

Installing the Signal Sciences module for Restify is similar to Node.js, except that 404 errors are handled differently in Restify. For best results, Signal Sciences should hook into the `NotFound` event. See the Restify node server api for more details.

Node.js Hapi v17 & v18

At the top of your application, add the following:

```
var Sigsci = require('sigsci-module-nodejs')
const Hapi = require('@hapi/hapi')

var sigsci = new Sigsci({
    path: '/var/run/sigsci.sock'
    // see other options below
})
const init = async() => {
 // Creating a server
 const server = Hapi.Server({
   port: 8085
 });

 server.ext('onRequest', sigsci.hapi17())
 server.events.on('response', sigsci.hapiEnding())
 //   config: {
 //     payload: {
 //       parse: false,
 //       maxBytes: 10 * 1024 * 1024,
 //       output: 'data'
 //     }
 //   },
 //   path: '/response',
 //   handler: responseHandler
 // })
};
init();
```

Node.js Hapi v14

At the top of your application, add the following:

```
var Sigsci = require('sigsci-module-nodejs')

var sigsci = new Sigsci({
    path: '/var/run/sigsci.sock'
    // see other options below
})
// Creating a Server
const Hapi = require('hapi')
const server = Hapi.Server({
    port: 8085
});
// Add SigSci request lifecycle methods, e.g.
// server.route({
//  method: ['GET', 'POST', 'PUT', 'PATCH', 'DELETE'],
//  path: '/dynamic/response',
//  handler: responseHandler
// })

server.ext('onRequest', sigsci.hapi14())
server.on('response', sigsci.hapiEnding())
server.start((err) => {
 if (err) {
   throw err
 }
 console.log('Server running at:', server.info.uri)
})
```

Node.js KOA

At the top of your application, add the following:

```
const Koa = require('koa');
const Router = require('koa-router');
var Sigsci = require('sigsci-module-nodejs')
const server = new Koa();
const router = new Router();
var sigsci = new Sigsci({
    path: '/var/run/sigsci.sock'
// see other options below
})

// add lifecycle methods here
// var dispatcher = async function (ctx) {
//     let req = ctx.req
// setup your endpoints here
// router.all('/response', dispatcher)

server.use(sigsci.koa())
server.use(router.routes())

server.listen(8085);
```

Configuration

You can module configuration options directly in the `Sigsci` object:

```
var sigsci = new Sigsci({
path: '/var/run/sigsci.sock'
...
})
```

Name	Description
`port`	Specifies the port to connect to the agent via TCP.
`host`	Specifies the IP address to connect to the agent via TCP (optional). Default: `localhost`
`path`	Specifies the Unix Domain Socket to connect to the agent via UDS.
`socketTimeout`	Number of milliseconds to wait for a response from the agent. After this time the module allows the original request to pass (i.e. fail open).
`maxPostSize`	Controls the maximum size in bytes of a POST body that is sent to the agent. If the body is larger than this value, the post body is not sent to the agent. This allows control over performance (larger POST bodies take longer to process) and to prevent DoS attacks.
`log`	The function to use to log error messages. By default it will be something to the effect of: `function (msg) { console.log(util.format('SIGSCI %s', msg))`

Additional details and default values are available in the `SigSci.js` file.

Next Steps

Verify the agent and module installation and explore module options.

Kubernetes Envoy

Introduction

In this example, the Signal Sciences agent runs in a Docker sidecar and communicates directly with an Envoy proxy deployed on the application.

Integrating the Signal Sciences Agent

The Signal Sciences Agent can be installed as a sidecar into each pod or as a service for some specialized needs.

The recommended way of installing the Signal Sciences Agent in Kubernetes is by integrating the `sigsci-agent` into a pod as a sidecar. This means adding the `sigsci-agent` as an additional container to the Kubernetes pod. As a sidecar, the agent will scale with the app/service in the pod instead of having to do this separately. However, in some situations, it may make more sense to install the `sigsci-agent` container as a service and scale it separately from the application.

The `sigsci-agent` container can be configured in various ways depending on the installation type and module being used.

You can use the `preStop` container hook to slow the pod's shutdown and ensure drain timeouts are met.

```
preStop:
  exec:
    command:
      - sleep
      - "30"
```

Alternatively, if you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

These instructions reference the `latest` version of the agent with `imagePullPolicy: Always`, which will pull the latest agent version even if one already exist locally. This is so the documentation does not fall out of date and anyone using this will not have an agent that stays stagnant. However, this may not be what if you need to keep installations consistent or on a specific version of the agent. In these cases, you should specify an agent version. Images on Docker Hub are tagged with their versions and a list of versions is available on Docker Hub.

Whether you choose to use the `latest` image or a specific version, there are a few items to consider to keep the agent up-to-date.

Using the `latest` Signal Sciences Container Image

If you do choose to use the `latest` image, then you will want to consider how you will keep the agent up to date.

- If you have used the `imagePullPolicy: Always` option, then the latest image will be pulled on each startup and your agent will continue to get updates.
- Alternatively, you may instead choose to manually update the local cache by periodically forcing a pull instead of always pulling on startup:

  ```
  docker pull signalsciences/sigsci-agent:latest
  ```

  Then, use `latest` with `imagePullPolicy: Never` set in the configuration so that pulls are never done on startup (only manually as above):

  ```
  - name: sigsci-agent
      image: signalsciences/sigsci-agent:latest
      imagePullPolicy: Never
      ...
  ```

Using a Versioned Signal Sciences Container Image

To use a specific version of the agent, replace `latest` with the agent version (represented here by `x.xx.x`). You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:x.xx.x
    imagePullPolicy: IfNotPresent
    ...
```

This will pull the specified agent version and cache it locally. If you use this method, then it is recommended that you parameterize the agent image, using Helm or similar, so that it is easier to update the agent images later on.

Using a Custom Tag for the Signal Sciences Container Image

It is also possible to apply a custom tag to a local agent image. To do this, pull the agent image (by version or use `latest`), apply a custom tag, then use that custom tag in the configuration. You will need to specify `imagePullPolicy: Never` so local images are only updated manually. After doing so, you will need to periodically update the local image to keep the agent up-to-date.

For example:

```
docker pull signalsciences/sigsci-agent:latest
docker tag signalsciences/sigsci-agent:latest signalsciences/sigsci-agent:testing
```

Then use this image tag in the configuration:

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:testing
    imagePullPolicy: Never
...
```

Configuring the Signal Sciences Agent Container

Agent configuration is normally done via the environment. Most configuration options are available as environment variables. Environment variables names have the configuration option name all capitalized, prefixed with `SIGSCI_` and any dashes (-) changed to underscores (_).

The `sigsci-agent` container has a few required options that need to be configured:

- Agent credentials (**Agent Access Key** and **Agent Secret Key**).
- A volume to write temporary files.

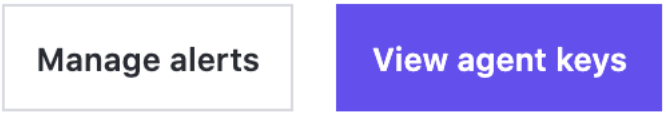
Agent Credentials

The `sigsci-agent` credentials are configured with two environment variables. These variables must be set or the agent will not start.

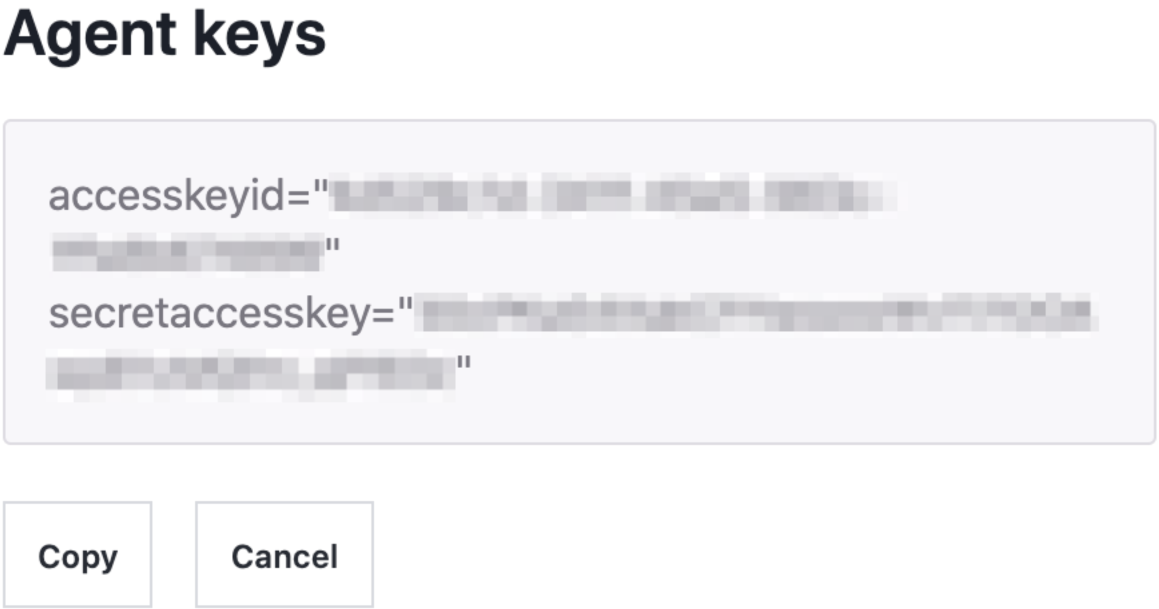
- **SIGSCI_ACCESSKEYID**: The **Agent Access Key** identifies which site in the Signal Sciences console that the agent is configured for.
- **SIGSCI_SECRETACCESSKEY**: The **Agent Secret Key** is the shared secret key to authenticate and authorize the agent.

The credentials can be found by following these steps:

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. Click **Agents** in the navigation bar. The agents page appears.
4. Click **View agent keys**. The agent keys window appears.



5. Copy the **Agent Access Key** and **Agent Secret Key**.



Agent keys

```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

Because of the sensitive nature of these values, we recommend you use the built in `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded values into the deployment configuration. This also makes any desired agent credential rotation easier to manage by having to change them in only one place.

Use the `valueFrom` option instead of the `value` option to use the `secrets` functionality. For example:

```
env:
  - name: SIGSCI_ACCESSKEYID
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: accesskeyid
  - name: SIGSCI_SECRETACCESSKEY
```

```
        name: sigsci.my-site-name-here
        key: secretaccesskey
```

The `secrets` functionality keeps secrets in various stores in Kubernetes. This guide uses the generic secret store in its examples, however any equivalent store can be used. Agent secrets can be added to the generic secret store using YAML similar to the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: sigsci.my-site-name-here
stringData:
  accesskeyid: 12345678-abcd-1234-abcd-1234567890ab
  secretaccesskey: abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

This can also be created from the command line with `kubectl` such as with the following example:

```
kubectl create secret generic sigsci.my-site-name-here \
  --from-literal=accesskeyid=12345678-abcd-1234-abcd-1234567890ab \
  --from-literal=secretaccesskey=abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

Additional information about Kubernetes `secrets` functionality can be found here.

Agent Temporary Volume

For added security, we recommended the `sigsci-agent` container be executed with the root filesystem mounted as read only. However, the agent still needs to write some temporary files such as the socket file for RPC communication and some periodically updated files such as GeoIP data.

To accomplish this with a read only root filesystem, there needs to be a writeable volume mounted. This writeable volume can also be shared to expose the RPC socket file to other containers in the same pod.

The recommended way of creating a writeable volume is to use the builtin `emptyDir` volume type. This is typically configured in the `volumes` section of a deployment, as shown in the following example:

```
volumes:
  - name: sigsci-tmp
    emptyDir: {}
```

Containers will then mount this volume at `/sigsci/tmp`:

```
volumeMounts:
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
```

The default in the official agent container image is to have the temporary volume mounted at `/sigsci/tmp`. If this needs to be moved for the agent container, then the following agent configuration options should also be changed from their defaults to match the new mount location:

- `rpc-address` defaults to `/sigsci/tmp/sigsci.sock`
- `shared-cache-dir` defaults to `/sigsci/tmp/cache`

Integrating the Signal Sciences agent into an Envoy Proxy

You can deploy the Signal Sciences Agent for integration with the Envoy Proxy via the External Authorization (`ext_authz`), HTTP filter. This filter communicates with the `sigsci-agent` via gRPC.

Generic Envoy Proxy

Configuration for Envoy and the Signal Sciences agent are documented with the other modules in the Envoy install guide. This guide is for deploying the Signal Sciences agent as a sidecar to your existing Envoy configuration. Deploying the `sigsci-agent` container as a sidecar to Envoy is similar to a typical module based deployment, but configuration is slightly different.

To deploy the Signal Sciences agent as a sidecar to Envoy, you must:

- Modify your existing Envoy configuration as noted in the Envoy install guide.

Modifying the Envoy Proxy configuration

Modify your existing Envoy configuration as detailed in the Envoy install guide.

Add the Signal Sciences Agent as an Envoy gRPC Service:

```
...
      containers:
      # Example Envoy front proxy running on port 8000
      - name: envoy-frontproxy
        image: signalsciences/envoy-frontproxy:latest
        imagePullPolicy: IfNotPresent
        args:
        - -c
        - /etc/envoy/envoy.yaml
        - --service-cluster
        - front-proxy
        - -l
        - info
        ports:
        - containerPort: 8000
      # Example helloworld app running on port 8080 without sigsci configured (accessed via Envoy proxy)
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
        # Address for the app to listen on
        - localhost:8080
        ports:
        - containerPort: 8080
      # Signal Sciences Agent running in Envoy gRPC mode (SIGSCI_ENVOY_GRPC_ADDRESS configured)
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: IfNotPresent
        # Configure the agent to use Envoy gRPC on port 9999
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: secretaccesskey
        # Configure the Envoy to expect response data (if using a gRPC access log config for Envoy)
        - name: SIGSCI_ENVOY_EXPECT_RESPONSE_DATA
          value: "1"
        # Configure the Envoy gRPC listener address on any unused port
        - name: SIGSCI_ENVOY_GRPC_ADDRESS
          value: localhost:9999
        ports:
        - containerPort: 9999
        securityContext:
          # The sigsci-agent container should run with its root filesystem read only
          readOnlyRootFilesystem: true
```

type:

```
...
      volumes:
      # Define a volume where sigsci-agent will write temp data and share the socket file,
      # which is required with the root filesystem is mounted read only
      - name: sigsci-tmp
        emptyDir: {}
```

Red Hat NGINX 1.14.1+

Add the package repositories

Add the version of the Red Hat CentOS package repository that you want to use.

Red Hat CentOS 8

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/8/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 7

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 6

> **Note:** After Q2 2017, RHEL6 and CentOS 6 will exit Production Phase 2 according to the Red Hat Enterprise Linux Life Cycle. Only limited critical security fixes will be issued. You will need to review the lifecycle document for details and plan appropriately.

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Install the NGINX module

1. Install the Signal Sciences NGINX module by running the following command, replacing `NN.NN` with your NGINX version number:

   ```
   sudo yum install nginx-module-sigsci-nxo-1.NN.NN*
   ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the `pid /run/nginx.pid;` line:

   ```
   load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
   ```

3. Restart the NGINX service to initialize the new module.

   - CentOS 7/RHEL 7 or higher

     ```
     systemctl restart nginx
     ```

   - CentOS 6/RHEL 6

     ```
     restart nginx
     ```

Windows Apache Module Install

Requirements

- Windows 10 or higher (64-bit), Windows Server 2016
- Apache 2.4
- Verify you have installed the Signal Sciences Windows Agent. This will ensure the appropriate folder structure is in place on your file system.

Installation

1. Download the Apache module from:

   https://dl.signalsciences.net/sigsci-module-apache/sigsci-module-apache_latest.zip

2. Extract the Signal Sciences Apache Module from the `.zip` archive to your Apache modules directory, replacing `PATH-TO-APACHE` with the path to your Apache installation:

   ```
   unzip sigsci-module-apache_latest.zip
   copy mod_sigsci.so PATH-TO-APACHE\modules\
   ```

3. Add the following line to your Apache configuration file (`httpd.conf`) after the **Dynamic Shared Object (DSO) Support** section to enable the Signal Sciences Apache module:

   ```
   LoadModule signalsciences_module modules/mod_sigsci.so
   ```

4. Test to confirm the configuration is correct, replacing `MY-SERVICE-NAME` with the name of your service:

   ```
   httpd.exe -n "MY-SERVICE-NAME" -t
   ```

5. Start the Apache service as normal, for example:

   ```
   net start Apache2.4
   ```

   Or restart the Apache service with the following example command, replacing `MY-SERVICE-NAME` with the name of your service:

   ```
   httpd.exe -k restart -n "MY-SERVICE-NAME"
   ```

Next Steps

Verify the agent and module installation and explore module options.

Alpine Linux Agent Installation

This guide explains how to install the Signal Sciences agent on Alpine Linux. This guide includes instructions for Alpine running in a Docker container, virtual machine (VM), or bare-metal server.

- You must have permission to view agent keys in the Signal Sciences web interface.

Add the package repository

Begin the agent installation by adding the version of the Alpine package repository that you want to use.

If you are running Alpine in a Docker container, run the following script to add the package repository:

```
apk update
apk add wget
wget -q https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys/
echo https://apk.signalsciences.net/3.17/main | tee -a /etc/apk/repositories && apk update
```

If you are running Alpine in a VM or on a bare-metal server, run the following script to add the package repository:

```
sudo apk update
sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys/
sudo echo https://apk.signalsciences.net/3.17/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Verify the downloaded key

After you've installed the Alpine package repository, verify the downloaded key contains the proper key by running the following command:

```
openssl rsa -pubin -in /etc/apk/keys/sigsci_apk.pub -text -noout
```

If the downloaded key contains the proper key, the expected output looks like the following:

```
Public-Key: (2048 bit)
Modulus:
    00:bb:23:1a:ef:0d:61:8f:8d:55:aa:ad:01:84:43:
    6c:46:42:42:ab:5b:ec:4e:4b:e2:e6:b6:e7:3d:45:
    b7:96:70:fe:16:95:aa:09:f1:90:82:40:e4:30:2b:
    9e:2a:03:e9:74:63:55:66:f0:db:8c:b9:5b:f8:45:
    5f:ad:4e:7a:14:da:02:83:c2:36:a0:84:74:a0:bb:
    f9:3f:03:c8:fe:80:6a:95:0c:17:22:55:40:30:18:
    51:d9:30:db:7c:1b:d0:06:4e:a9:51:1a:31:0e:33:
    f0:6e:ad:53:98:31:a5:ac:a3:a1:44:83:72:a1:ca:
    78:e3:24:70:ab:7a:0e:66:32:3b:f6:c9:90:16:dc:
    89:d0:52:7a:50:a8:f8:59:0a:34:12:2e:85:11:f5:
    80:0d:d4:7d:a7:7b:3b:d7:d9:1e:28:ed:bb:f7:08:
    2e:9f:73:a5:23:d8:53:b4:7e:21:dd:ae:92:4a:d0:
    5b:86:21:9c:82:05:21:29:eb:c1:ab:91:cd:1a:7b:
    95:6d:43:d3:1a:a9:62:2b:b0:95:9e:cf:18:82:64:
    02:f9:38:7e:7f:47:9f:d9:f3:ac:fd:2c:30:ff:75:
    b1:11:27:1c:7a:d6:ca:04:19:f8:31:80:42:e9:4a:
    0d:ab:d5:b8:ad:f2:35:31:a5:3f:98:19:99:fc:29:
    e8:4f
Exponent: 65537 (0x10001)
```

Install and configure the Signal Sciences Agent package

Now that you've downloaded the Alpine package repository and verified that you have the proper key, you can install the Signal Sciences Agent package.

Run the following command to install the Signal Sciences Agent package:

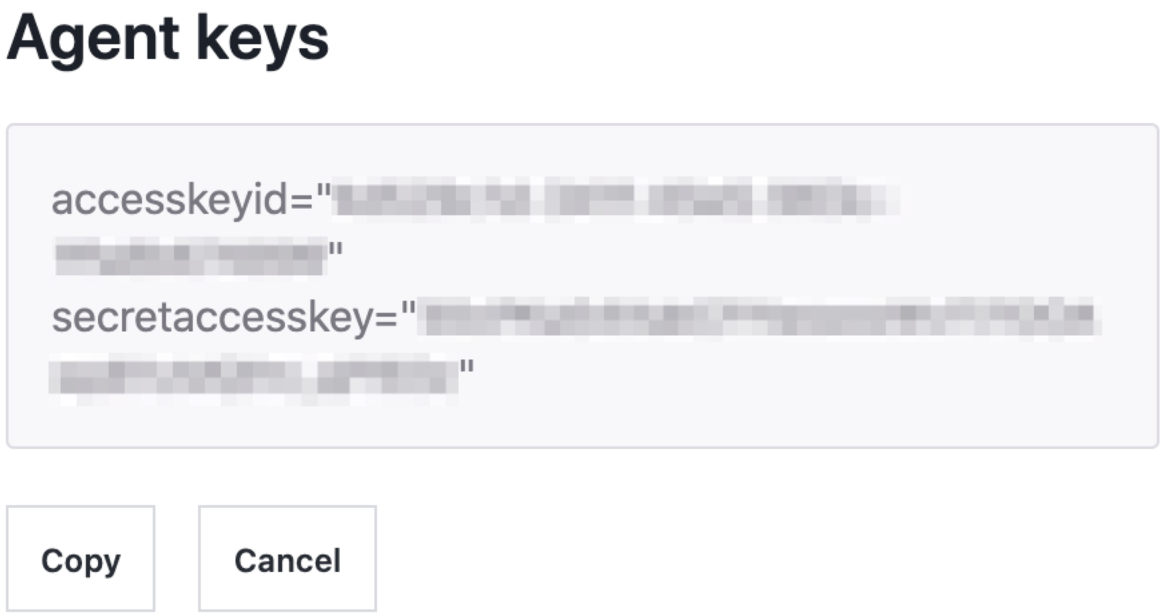
```
sudo apk add sigsci-agent
```

Once the agent package is installed, you must create an agent configuration file and add the **Agent Access Key** and **Agent Secret Key**:

1. Create an empty agent configuration file in the following directory: `/etc/sigsci/agent.conf`.
2. Log in to the Signal Sciences console.

5. Click the **View agent keys** button. The agent keys window appears.

6. Click the **Copy** button to copy the **Agent Access Key** and **Agent Secret Key** to your clipboard.



Agent keys

```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

7. Navigate to the agent configuration file and paste the **Agent Access Key** and **Agent Secret Key** into the file.

```
accesskeyid = "AGENTACCESSKEYHERE"
secretaccesskey = "AGENTSECRETACCESSKEYHERE"
```

8. Save the agent configuration file.

Start the Signal Sciences Agent

Now that you've installed and configured the agent package, you can start the Signal Sciences agent.

If you are running Alpine in a Docker container, run the following command to start the Signal Sciences agent:

```
/usr/sbin/sigsci-agent
```

If you are running Alpine in a VM or on a bare-metal server, run the following command to allow the agent to start on reboot:

```
sudo rc-update add sigsci-agent default
```

Then, start the agent by running any of the following commands:

```
sudo service sigsci-agent start
sudo rc-service sigsci-agent start
sudo /etc/init.d/sigsci-agent start
```

Next Steps

Explore our module options and install the Signal Sciences module.

OpenShift Install

The Signal Sciences agent can be deployed on the Red Hat OpenShift Container Platform.

Installation

Installing the Signal Sciences module and agent in an OpenShift container is similar to a typical Red Hat installation. However, the primary difference for an OpenShift container installation is all processes must run under a *non root* account. To meet this requirement, the only extra step is configuring the module and agent to use a socket file that the non root account has read/write access to.

For more information on running processes as *non root*, see OpenShift guidance here.

Configuring the agent

There are three options for configuring the socket file location. Use the option that works best for your container build process. The examples below use a directory that a non root user would have access to. You can specify a different location, but ensure your non root user account has the read/write permissions to that location.

- You can set the `SIGSCI_RPC_ADDRESS` environment variable in your Dockerfile:

  ```
  ENV SIGSCI_RPC_ADDRESS unix:/tmp/sigsci.sock
  ```

- You can export the `SIGSCI_RPC_ADDRESS` environment variable in a script when your container starts:

  ```
  export SIGSCI_RPC_ADDRESS=unix:/tmp/sigsci.sock
  ```

- You can set the `rpc-address` configuration option in your agent configuration file (by default at `/etc/sigsci/agent.conf`):

  ```
  rpc-address="unix:/tmp/sigsci.sock"
  ```

Additional agent configuration options are listed on the agent configuration page.

Installing and configuring the module

Install and configure your module following one of these sets of instructions.

Apache module install

Follow the Apache module installation instructions for Red Hat.

In your Apache configuration file (`httpd.conf`), add the `AgentHost` directive after the Signal Sciences module is called:

```
AgentHost "/tmp/sigsci.sock"
```

NGINX module install

Follow the NGINX module installation instructions for Red Hat.

Update the `sigsci.agenthost` directive in the module's configuration file located at `/opt/sigsci/nginx/sigsci.conf`. You will need to remove `--` to uncomment the line:

```
sigsci.agenthost = "unix:/tmp/sigsci.sock"
```

Example Dockerfile

Below is an example section of a Dockerfile that installs the Signal Sciences agent and module (for Apache HTTPD Server) and configures them to use a socket file location accessible to a non root account.

```
...

# Add the Signal Sciences package repository
RUN echo "[sigsci_release]" > /etc/yum.repos.d/sigsci.repo && \
    echo "name=sigsci_release" >> /etc/yum.repos.d/sigsci.repo && \
    echo "baseurl=https://yum.signalsciences.net/release/el/7/\$basearch" >> /etc/yum.repos.d/sigsci.repo && \
    echo "repo_gpgcheck=1" >> /etc/yum.repos.d/sigsci.repo && \
    echo "gpgcheck=0" >> /etc/yum.repos.d/sigsci.repo && \
    echo "enabled=1" >> /etc/yum.repos.d/sigsci.repo && \
    echo "gpgkey=https://yum.signalsciences.net/release/gpgkey" >> /etc/yum.repos.d/sigsci.repo && \
    echo "sslverify=1" >> /etc/yum.repos.d/sigsci.repo && \
    echo "sslcacert=/etc/pki/tls/certs/ca-bundle.crt" >> /etc/yum.repos.d/sigsci.repo

# Install the Signal Sciences agent
RUN yum -y install sigsci-agent

# Configure the Signal Sciences agent
ENV SIGSCI_RPC_ADDRESS=unix:/tmp/sigsci.sock

# Install the Signal Sciences module
RUN yum install -y sigsci-module-apache
```

```
RUN echo "LoadModule signalsciences_module /etc/httpd/modules/mod_signalsciences.so" >> /etc/httpd/conf/httpd.con
    echo 'AgentHost "/tmp/sigsci.sock"' >> /etc/httpd/conf/httpd.conf

...
```

IPv6 support

Signal Sciences provides full support for IPv6 in the product, including:

- Detection and decisioning: Requests are appropriately tagged and IPv6 addresses can be automatically flagged within the product.
- Blocklist and allowlist support: IPv6 addresses can be blocklisted and allowlisted within the UI.
- Search: IPv6 addresses can be filtered within search.
- Country/DNS lookups: IPv6 addressed are resolved and mapped to countries, where possible.

Azure App Service Site Extension

> **Note:** The Signal Sciences site extension for Azure App Service does not currently support Azure Functions.

The Azure site extension for Signal Sciences adds the Signal Sciences Next-Gen Web Application Firewall (WAF) to any IIS web application hosted on Azure App Service.

The Signal Sciences Azure site extension downloads and installs the Signal Sciences agent and IIS module. The extension also registers the IIS module to the IIS web server in Azure App Service by generating the XML transformation file, `applicationHost.xdt`. XML transformations are currently the only way to edit the IIS configuration file, `applicationHost.config`.

The Signal Sciences IIS module and agent are configured by using environment variables. Environment variables are set in the web app configuration in the Azure Portal.

Module and agent binaries are extracted into a directory in the App Service environment with the name derived from the downloaded zip file. Agent and module binaries may not be deleted if the site is running.

Signal Sciences Agent Access Keys configuration

Before adding the Signal Sciences site extension, you must first set the Signal Sciences Agent Access Key and Secret Key by setting environment variables in the application settings on https://portal.azure.com/.

1. Log in to the Azure Portal.

2. Click **App Services**. The App Services menu page appears.

3. Select your web app.

4. Click **Configuration**. The Configuration menu page appears.

5. Click **Application settings**. The Application Settings menu page appears.

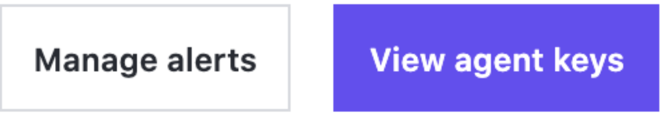
6. Click **New application setting**. The New Application Setting menu page appears.

7. Locate the **Agent Keys** for your Signal Sciences site:

    1. Log in to the Signal Sciences console.

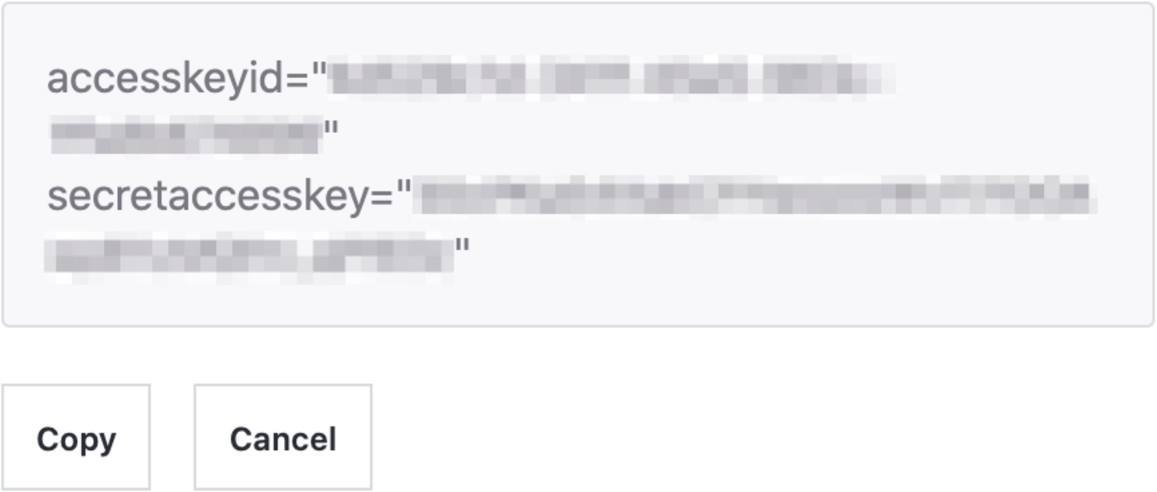
    2. From the **Sites** menu, select a site if you have more than one site.

    3. Click **Agents** in the navigation bar. The agents page appears.

    4. Click **View agent keys**. The agent keys window appears.

    5. Copy the **Agent Access Key** and **Agent Secret Key**.



accesskeyid="[redacted]"
secretaccesskey="[redacted]"

8. In the New Application Setting menu page of the Azure Portal, add the following variables as two name/value pairs:

```
Name: SIGSCI_ACCESSKEYID
Value: <accesskeyid from Signal Sciences console>

Name: SIGSCI_SECRETACCESSKEY
Value:<secretaccesskey from Signal Sciences console>
```

9. Click **Save**.

10. Click on **Overview** in the side bar. The Overview menu page appears.

11. Click the **Stop** button and then the **Start** button to restart the web app.

Install the Signal Sciences WAF site extension

> **Note:** The site extension will take a few minutes to download and install. During this time, the web application may be unavailable or display a `502` error until the site extension is installed.

1. Log in to the Azure Portal.
2. Click **App Services**. The App Services menu page appears.
3. Select your web app.
4. Click on **Overview** in the side bar. The Overview menu page appears.
5. Click the **Stop** button to stop the web app.
6. Click **Extensions** in the sidebar. The Extensions menu page appears.
7. Click **Add**. The Add Extension menu page appears.
8. Click **Choose Extension**. The Choose Extension menu page appears.
9. Select the **Signal Sciences WAF**. The Signal Sciences WAF extension page appears.
10. Click **OK**.
11. Click on **Overview** in the side bar. The Overview menu page appears.
12. Click the **Start** button to start your web app.

Managing the Signal Sciences WAF site extension

Follow these steps when managing the Signal Sciences WAF site extension.

Uninstalling the Signal Sciences WAF site extension

1. Log in to the Azure Portal.
2. Click **App Services**. The App Services menu page appears.
3. Select your web app.
4. Click on **Overview** in the side bar. The Overview menu page appears.
5. Click the **Stop** button to stop the web app.
6. Click **Extensions** in the sidebar. The Extensions menu page appears.
7. Select the **Signal Sciences WAF**. The Signal Sciences WAF extension menu page appears.
8. Click **Delete**.

- reinstalling the extension
- using the Azure CLI

Reinstalling the extension

In the Azure Portal, uninstall and reinstall the Signal Sciences WAF site extension. When the extension is reinstalled, the latest version of the Signal Sciences agent and IIS module will be downloaded and installed.

Using the Azure CLI

Open the Azure CLI and run the `install.cmd` script in the site extension directory. This method can also be used in a PowerShell script for automating the upgrade of multiple agents.

1. Log in to the Azure Portal.
2. Click **App Services**. The App Services menu page appears.
3. Select your web app.
4. Click on **Console** in the sidebar. The Console page appears.
5. In the Windows `cmd` shell run the install script:

```
cd D:\home\SiteExtensions\SignalSciences.Azure.Site.Extension
install.cmd
```

Troubleshooting

- All private site extensions can be disabled by setting **WEBSITE_PRIVATE_EXTENSIONS** to `0` in Application Settings.

  Note: Restart the web app after saving the setting to reflect the changes.

- Windows event log can be viewed at https://APP.scm.azurewebsites.net/DebugConsole/?shell=powershell, replacing `APP` with the name of your web app.

  Click on **LogFiles** and select **eventlog.xml**.

Kubernetes Istio

Introduction

In this example, the Signal Sciences agent runs in a Docker sidecar and integrates directly with an Istio service mesh deployed on the application. In this configuration, you can configure Signal Sciences to inspect east/west (service-to-service) web requests along with the traditional north/south (client to server) requests.

Integrating the Signal Sciences Agent

The Signal Sciences Agent can be installed as a sidecar into each pod or as a service for some specialized needs.

The recommended way of installing the Signal Sciences Agent in Kubernetes is by integrating the `sigsci-agent` into a pod as a sidecar. This means adding the `sigsci-agent` as an additional container to the Kubernetes pod. As a sidecar, the agent will scale with the app/service in the pod instead of having to do this separately. However, in some situations, it may make more sense to install the `sigsci-agent` container as a service and scale it separately from the application.

The `sigsci-agent` container can be configured in various ways depending on the installation type and module being used.

You can use the `preStop` container hook to slow the pod’s shutdown and ensure drain timeouts are met.

```
preStop:
  exec:
    command:
      - sleep
      - "30"
```

Getting and Updating the Signal Sciences Agent Container Image

An official `signalsciences/sigsci-agent` container image is available from the Signal Sciences account on Docker Hub.

even if one already exist locally. This is so the documentation does not fall out of date and anyone using this will not have an agent that stays stagnant. However, this may not be what if you need to keep installations consistent or on a specific version of the agent. In these cases, you should specify an agent version. Images on Docker Hub are tagged with their versions and a list of versions is available on Docker Hub.

Whether you choose to use the `latest` image or a specific version, there are a few items to consider to keep the agent up-to-date.

Using the `latest` Signal Sciences Container Image

If you do choose to use the `latest` image, then you will want to consider how you will keep the agent up to date.

- If you have used the `imagePullPolicy: Always` option, then the latest image will be pulled on each startup and your agent will continue to get updates.
- Alternatively, you may instead choose to manually update the local cache by periodically forcing a pull instead of always pulling on startup:

  ```
  docker pull signalsciences/sigsci-agent:latest
  ```

  Then, use `latest` with `imagePullPolicy: Never` set in the configuration so that pulls are never done on startup (only manually as above):

  ```
  - name: sigsci-agent
      image: signalsciences/sigsci-agent:latest
      imagePullPolicy: Never
      ...
  ```

Using a Versioned Signal Sciences Container Image

To use a specific version of the agent, replace `latest` with the agent version (represented here by `x.xx.x`). You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:x.xx.x
    imagePullPolicy: IfNotPresent
    ...
```

This will pull the specified agent version and cache it locally. If you use this method, then it is recommended that you parameterize the agent image, using Helm or similar, so that it is easier to update the agent images later on.

Using a Custom Tag for the Signal Sciences Container Image

It is also possible to apply a custom tag to a local agent image. To do this, pull the agent image (by version or use `latest`), apply a custom tag, then use that custom tag in the configuration. You will need to specify `imagePullPolicy: Never` so local images are only updated manually. After doing so, you will need to periodically update the local image to keep the agent up-to-date.

For example:

```
docker pull signalsciences/sigsci-agent:latest
docker tag signalsciences/sigsci-agent:latest signalsciences/sigsci-agent:testing
```

Then use this image tag in the configuration:

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:testing
    imagePullPolicy: Never
...
```

Configuring the Signal Sciences Agent Container

Agent configuration is normally done via the environment. Most configuration options are available as environment variables. Environment variables names have the configuration option name all capitalized, prefixed with `SIGSCI_` and any dashes (-) changed to underscores (_). For example, the max-procs option would become the `SIGSCI_MAX_PROCS` environment variable. For more details on what options are available, see the Agent Configuration documentation.

The `sigsci-agent` container has a few required options that need to be configured:

Agent Credentials

The `sigsci-agent` credentials are configured with two environment variables. These variables must be set or the agent will not start.

- **SIGSCI_ACCESSKEYID**: The **Agent Access Key** identifies which site in the Signal Sciences console that the agent is configured for.
- **SIGSCI_SECRETACCESSKEY**: The **Agent Secret Key** is the shared secret key to authenticate and authorize the agent.

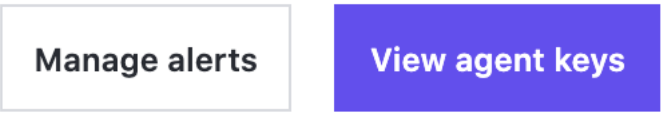
The credentials can be found by following these steps:

1. Log in to the Signal Sciences console.

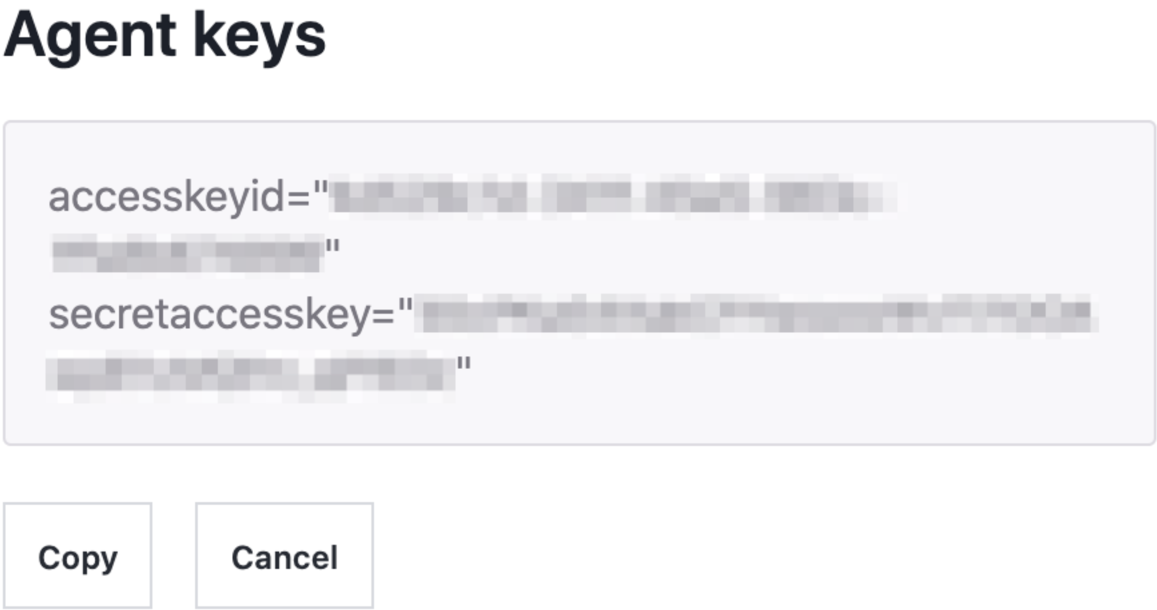
2. From the **Sites** menu, select a site if you have more than one site.

3. Click **Agents** in the navigation bar. The agents page appears.

4. Click **View agent keys**. The agent keys window appears.



5. Copy the **Agent Access Key** and **Agent Secret Key**.



Agent keys

```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

Because of the sensitive nature of these values, we recommend you use the built in `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded values into the deployment configuration. This also makes any desired agent credential rotation easier to manage by having to change them in only one place.

Use the `valueFrom` option instead of the `value` option to use the `secrets` functionality. For example:

```
env:
  - name: SIGSCI_ACCESSKEYID
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: accesskeyid
  - name: SIGSCI_SECRETACCESSKEY
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: secretaccesskey
```

```
apiVersion: v1
kind: Secret
metadata:
  name: sigsci.my-site-name-here
stringData:
  accesskeyid: 12345678-abcd-1234-abcd-1234567890ab
  secretaccesskey: abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

This can also be created from the command line with `kubectl` such as with the following example:

```
kubectl create secret generic sigsci.my-site-name-here \
  --from-literal=accesskeyid=12345678-abcd-1234-abcd-1234567890ab \
  --from-literal=secretaccesskey=abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

Additional information about Kubernetes `secrets` functionality can be found here.

Agent Temporary Volume

For added security, we recommended the `sigsci-agent` container be executed with the root filesystem mounted as read only. However, the agent still needs to write some temporary files such as the socket file for RPC communication and some periodically updated files such as GeoIP data.

To accomplish this with a read only root filesystem, there needs to be a writeable volume mounted. This writeable volume can also be shared to expose the RPC socket file to other containers in the same pod.

The recommended way of creating a writeable volume is to use the builtin `emptyDir` volume type. This is typically configured in the `volumes` section of a deployment, as shown in the following example:

```
volumes:
  - name: sigsci-tmp
    emptyDir: {}
```

Containers will then mount this volume at `/sigsci/tmp`:

```
volumeMounts:
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
```

The default in the official agent container image is to have the temporary volume mounted at `/sigsci/tmp`. If this needs to be moved for the agent container, then the following agent configuration options should also be changed from their defaults to match the new mount location:

- `rpc-address` defaults to `/sigsci/tmp/sigsci.sock`
- `shared-cache-dir` defaults to `/sigsci/tmp/cache`

Integrating the Signal Sciences agent using External Authorization

As of Istio v1.9, support has been added to setup an authorization policy that delegates access control to an external authorization system.

The snippets below follow Istio's example and enhance the process to replace the example `ext-authz` service with the Signal Sciences Agent. Refer to the Istio documentation for initial namespace and test workloads, as those are referenced in the snippets below. All files are applied to the 'foo' namespace unless otherwise indicated.

Deploy the external authorizer

Assumes the secrets have been applied.

```
apiVersion: v1
kind: Service
metadata:
  name: sigsci-agent
  labels:
    app: sigsci-agent
spec:
  ports:
  - name: grpc
```

```
    app: sigsci-agent
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sigsci-agent
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sigsci-agent
  template:
    metadata:
      labels:
        app: sigsci-agent
    spec:
      containers:
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: IfNotPresent
        # Configure the agent to use Envoy gRPC on port 9999
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci-agent-accesskey
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci-agent-accesskey
              key: secretaccesskey
        # Configure the Envoy to expect response data (if using a gRPC access log config for Envoy)
        - name: SIGSCI_ENVOY_EXPECT_RESPONSE_DATA
          value: "1"
        - name: SIGSCI_ENVOY_GRPC_ADDRESS
          value: :9999
        ports:
        - containerPort: 9999
        securityContext:
          # The sigsci-agent container should run with its root filesystem read only
          readOnlyRootFilesystem: true
---
```

Verify the Agent is running.

```
kubectl logs "$(kubectl get pod -l app=sigsci-agent -n foo -o jsonpath={.items..metadata.name})" -n foo -c sigsci-
```

Define the external authorizer

Edit the mesh config with the following command and add the extension provide definitions.

```
kubectl edit configmap istio -n istio-system
```

```
data:
  mesh: |-
    # Add the following content to define the external authorizers.
    extensionProviders:
      - name: "sigsci-agent-ext-authz"
```

```
          timeout: 0.2s
          failOpen: true
      - name: "sigsci-agent-access-log"
        envoyHttpAls:
          service: "sigsci-agent.foo.svc.cluster.local"
          port: "9999"
          additionalRequestHeadersToLog:
          - "x-sigsci-request-id"
          - "x-sigsci-waf-response"
          - "accept"
          - "content-type"
          - "content-length"
          additionalResponseHeadersToLog:
          - "date"
          - "server"
          - "content-type"
          - "content-length"
```

Enable with external authorization

Enable the external authorization and apply logging.

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: ext-authz
spec:
  selector:
    matchLabels:
      app: httpbin
  action: CUSTOM
  provider:
    # The provider name must match the extension provider defined in the mesh config.
    name: sigsci-agent-ext-authz
  rules:
    # The rules specify when to trigger the external authorizer.
    - to:
      - operation:
          paths: ["/headers"]
```

```
# kubectl apply -f logging.yaml
apiVersion: telemetry.istio.io/v1alpha1
kind: Telemetry
metadata:
  name: mesh-default
  namespace: istio-system
spec:
  accessLogging:
    - providers:
      - name: sigsci-agent-access-log
```

```
# In another terminal curl the httpbin app:
kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.items..metadata.name})" -c sleep -n foo -- curl

# tail the logs
kubectl logs -f "$(kubectl get pod -l app=sigsci-agent -n foo -o jsonpath={.items..metadata.name})" -n foo -c sig
```

Integrating the Signal Sciences agent using EnvoyFilter

Istio uses Envoy proxy under its hood. Because of this, Istio can use the Signal Sciences agent in gRPC mode in the same way as with a generic Envoy install. The method of installing and configuring the Signal Sciences agent is similar to a generic Envoy install except the Envoy

To add Signal Sciences support to an Istio based application deployment, you will need to:

- Add the `sigsci-agent` container to the pod, configured in Envoy gRPC listener mode.
- Add an `emptyDir{}` volume as a place for the `sigsci-agent` to write temporary data.
- Add an Istio `EnvoyFilter` for the app to allow the required Envoy configuration to be injected into the generated `istio-proxy` config.

Add the Signal Sciences agent as an Envoy gRPC service

```
...
      containers:
      # Example helloworld app running on port 8000 without sigsci configured
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
        # Address for the app to listen on
        - localhost:8080
        ports:
        - containerPort: 8080
      # Signal Sciences Agent running in Envoy gRPC mode (SIGSCI_ENVOY_GRPC_ADDRESS configured)
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: IfNotPresent
        # Configure the agent to use Envoy gRPC on port 9999
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
              # This secret needs added (see docs on sigsci secrets)
              name: sigsci.my-site-name-here
              key: secretaccesskey
        # Configure the Envoy to expect response data (if using a gRPC access log config for Envoy)
        - name: SIGSCI_ENVOY_EXPECT_RESPONSE_DATA
          value: "1"
        # Configure the Envoy gRPC listener address on any unused port
        - name: SIGSCI_ENVOY_GRPC_ADDRESS
          value: localhost:9999
        ports:
        - containerPort: 9999
        securityContext:
          # The sigsci-agent container should run with its root filesystem read only
          readOnlyRootFilesystem: true
```

Adding the Signal Sciences agent temp volume definition to the deployment

The agent temp volume needs to be defined for use by the other containers in the pod using the builtin `emptyDir: {}` volume type:

```
...
      volumes:
      # Define a volume where sigsci-agent will write temp data and share the socket file,
      # which is required with the root filesystem is mounted read only
      - name: sigsci-tmp
        emptyDir: {}
```

Adding the Istio EnvoyFilter object to inject the required Envoy config into the Istio proxy

name below. Additional Envoy configuration options are outlined in the Envoy install guide. These sections are highlighted with comments in the example YAML.

Example `example-helloworld_sigsci-envoyfilter.yaml`:

```
# The following adds the required Envoy configuration into the istio-proxy configuration
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  # This needs adjusted to be the app name protected by sigsci
  name: helloworld
spec:
  workloadSelector:
    labels:
      # This needs adjusted to be the app name protected by sigsci
      app: helloworld

  # Patch the Envoy configuration, adding in the required sigsci config
  configPatches:

  # Adds the ext_authz HTTP filter for the sigsci-agent ext_authz API
  - applyTo: HTTP_FILTER
    match:
      context: SIDECAR_INBOUND
      listener:
        name: virtualInbound
        filterChain:
          filter:
            name: "envoy.http_connection_manager"
    patch:
      operation: INSERT_BEFORE
      value:
        # Configure the envoy.ext_authz here:
        name: envoy.filters.http.ext_authz
        typed_config:
          "@type": "type.googleapis.com/envoy.extensions.filters.http.ext_authz.v3.ExtAuthz"
          transport_api_version: "V3"
          grpc_service:
            # NOTE: *SHOULD* use envoy_grpc as ext_authz can use dynamic clusters and has connection pooling
            envoy_grpc:
              cluster_name: sigsci-agent-grpc
            timeout: 0.2s
          failure_mode_allow: true
          with_request_body:
            max_request_bytes: 8192
            allow_partial_message: true

  # Adds the access_log entry for the sigsci-agent http_grpc_access_log API
  - applyTo: NETWORK_FILTER
    match:
      context: SIDECAR_INBOUND
      listener:
        name: virtualInbound
        filterChain:
          filter:
            name: "envoy.http_connection_manager"
    patch:
      operation: MERGE
      value:
          access_log:
          # Configure the envoy.http_grpc_access_log here:
          - name: "envoy.http_grpc_access_log"
            typed_config:
              "@type": "type.googleapis.com/envoy.extensions.access_loggers.grpc.v3.HttpGrpcAccessLogConfig"
              common_config:
                log_name: "sigsci-agent-grpc"
                transport_api_version: "V3"
                grpc_service:
                  # NOTE: *MUST* use google_grpc as envoy_grpc cannot handle a dynamic cluster for ALS (yet)
                  google_grpc:
                    # The address *MUST* be 127.0.0.1 so that communication is intra-pod
                    # Configure the sigsci-agent port number here:
                    target_uri: 127.0.0.1:9999
                    stat_prefix: "sigsci-agent"
                  timeout: 0.2s
              additional_request_headers_to_log:
              # These are required:
              - "x-sigsci-request-id"
              - "x-sigsci-waf-response"
              # These are additional you want recorded:
              - "accept"
              - "content-type"
              - "content-length"
              additional_response_headers_to_log:
              # These are additional you want recorded:
              - "date"
              - "server"
              - "content-type"
              - "content-length"

  # Adds a dynamic cluster for the sigsci-agent via CDS for sigsci-agent ext_authz API
  - applyTo: CLUSTER
    patch:
      operation: ADD
      value:
        name: sigsci-agent-grpc
        type: STRICT_DNS
        connect_timeout: 0.5s
        http2_protocol_options: {}
        load_assignment:
          cluster_name: sigsci-agent-grpc
          endpoints:
          - lb_endpoints:
            - endpoint:
                address:
                  socket_address:
                    # The address *MUST* be 127.0.0.1 so that communication is intra-pod
                    address: 127.0.0.1
                    # Configure the agent port here:
                    port_value: 9999
```

The application can then be deployed as you normally would with Istio. For example:

```
$ istioctl kube-inject -f example-helloworld-sigsci.yaml | kubectl apply -f -
service/helloworld created
deployment.apps/helloworld created
$ kubectl apply -f example-helloworld-sigsci_envoyfilter.yaml
envoyfilter.networking.istio.io/helloworld created
```

```
$ kubectl get pod helloworld-7954bb57bc-pfr22 -o jsonpath='{.spec.containers[*].name}'
helloworld sigsci-agent istio-proxy
$ kubectl logs helloworld-7954bb57bc-pfr22 sigsci-agent | head
2019/10/01 21:04:57.540047 Signal Sciences Agent 4.39.0 starting as user sigsci with PID 1, Max open files=104857
2019/10/01 21:04:57.541987 =====================================================
2019/10/01 21:04:57.542028 Agent:    helloworld-7954bb57bc-pfr22
2019/10/01 21:04:57.542034 System:   alpine 3.9.4 (linux 4.9.184-linuxkit)
2019/10/01 21:04:57.542173 Memory:   1.672G / 3.854G RAM available
2019/10/01 21:04:57.542187 CPU:      6 MaxProcs / 12 CPU cores available
2019/10/01 21:04:57.542257 =====================================================
2019/10/01 21:04:57.630755 Envoy gRPC server on 127.0.0.1:9999 starting
```

Note that there are three containers running in the pod: `app=helloworld`, `sigsci-agent`, and the `istio-proxy`.

Red Hat NGINX 1.10-1.14

Add the package repositories

Add the version of the Red Hat CentOS package repository that you want to use.

Red Hat CentOS 8

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/8/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 7

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 6

> **Note:** After Q2 2017, RHEL6 and CentOS 6 will exit Production Phase 2 according to the Red Hat Enterprise Linux Life Cycle. Only limited critical security fixes will be issued. You will need to review the lifecycle document for details and plan appropriately.

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
```

Enable Lua for NGINX

Some older versions of NGINX don’t support native loading of Lua modules. Therefore, we require NGINX to be built with Lua and LuaJIT support. You must first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

Install the Lua NGINX Module

Install the dynamic Lua NGINX Module appropriate for your NGINX distribution:

NGINX.org distribution

- NGINX 1.12.1 or higher

  ```
  sudo yum install nginx-module-lua-`rpm -q --qf "%{VERSION}" nginx`
  ```

- NGINX 1.11

  ```
  sudo yum install nginx111-lua-module
  ```

- NGINX 1.10

  ```
  sudo yum install nginx110-lua-module
  ```

Red Hat distribution

- NGINX 1.12.2 or higher

  ```
  sudo yum install nginx-module-lua-epel
  ```

- NGINX 1.11

  ```
  sudo yum install nginx111-lua-module
  ```

- NGINX 1.10

  ```
  sudo yum install nginx110-lua-module-epel
  ```

Enable the Lua NGINX Module

1. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the line that starts with `pid`:

   ```
   load_module /usr/lib64/nginx/modules/ndk_http_module.so;
   load_module /usr/lib64/nginx/modules/ngx_http_lua_module.so;
   ```

   Alternatively, you can create a `mod-lua.conf` file with the above lines in the NGINX dynamic module configuration directory.

2. Restart the NGINX service to initialize the new module:

   - CentOS 7/RHEL 7 or higher

     ```
     systemctl restart nginx
     ```

   - CentOS 6/RHEL 6

     ```
     restart nginx
     ```

Check that Lua is loaded correctly

Load the following config (e.g., `sigsci_check_lua.conf`) with NGINX to verify that Lua has been loaded properly:

```
# Config just to test for lua jit support
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
```

```
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768;
    # multi_accept on;
}
http {
init_by_lua '

local m = {}
local ngx_lua_version = "dev"

if ngx then
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
if not r then
  error("ERROR: No lua jit support: No support for SigSci Lua module")
else

  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
  end

end

';

}
```

Example of a successfully loaded config and its output

```
$ nginx -t -c <your explicit path>/sigsci_check_lua.conf

nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
nginx: the configuration file <your explicit path>/sigsci_check_lua.conf syntax is ok
nginx: configuration file <your explicit path>/sigsci_check_lua.conf test is successful
```

Install the NGINX module

1. Install the module.

   ```
   yum install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX service to initialize the new module.

- CentOS 6/RHEL 6

  ```
  restart nginx
  ```

.Net Module Install

Requirements

- .NET Framework 4.5 or higher.
- Verify you have installed the Signal Sciences Windows Agent. This will ensure the appropriate folder structure is in place on your file system.
- Download the latest .NET Module, or get it via Nuget.

Install

1. Download the latest Signal Sciences .Net module via one of these methods:
   - Directly from https://dl.signalsciences.net/sigsci-module-dotnet/sigsci-module-dotnet_latest.zip
   - Via Nuget
2. Extract the contents of `sigsci-module-dotnet-x.x.x.zip` to your application's `bin` directory.
3. Add the following sections to your application's `web.config` file:

```
<configuration>
    ...
    <configSections>
        <section name="SignalSciencesModule" type="SignalSciences.ModuleConfiguration"/>
    </configSections>

    ...
    <system.webServer>
        <modules>
            <add name="SignalSciencesModule" type="SignalSciences.HttpModule"/>
        </modules>
    </system.webServer>
    ...

    <SignalSciencesModule agentEndPoint="127.0.0.1:737" />
    ...
</configuration>
```

4. Restart the web site service.

   Note: Ensure the `AgentEndPoint` value is set to the same IP and port configured with the Signal Sciences agent's `rpc-address` value. See the Windows agent installation documentation for additional information about Windows agent configuration options.

.NET module configuration

Option	Default	Description
`agentEndPoint`	required, no default	The TCP endpoint (`host:port`) that the Agent is listening on. `host` can be either a hostname or an IPv4 or IPv6 address.
`filterHeaders`	optional, no default	Comma-separated list of request and response headers that should not be sent to the Agent. Case insensitive. Regardless of configuration, it always includes `Cookie`, `Set-Cookie`, `Authorization` and `X-Auth-Token`.
`agentRpcTimeoutMillis`	optional, default: 200	Maximum number of milliseconds allowed for each RPC call to the Agent.
`agentConnectionPoolSize`	optional, default: 10	Number of connections that, once opened, will be retained in a pool.
`maxPostSize`	optional, default:	A request body above this size will not be sent to the Agent.

`anomalySize`	optional, default: 524288	If the HTTP response is this size or larger, log it with the Agent.
`anomalyDurationMillis`	optional, default: 1000	If the response took longer than this number of milliseconds, log it with the Agent.

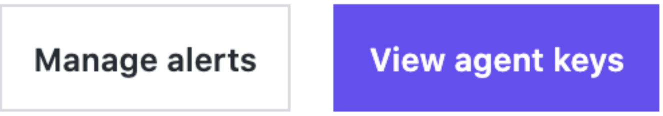
Sample advanced .NET module configuration

```
<SignalSciencesModule
            agentEndPoint="127.0.0.1:737"
            filterHeaders="X-My-Private-Header, X-My-Other-Header"
            agentRpcTimeoutMillis="200"
            agentConnectionPoolSize="10"
            maxPostSize="100000"
            anomalySize="524288"
            anomalyDurationMillis="1000"
            />
```

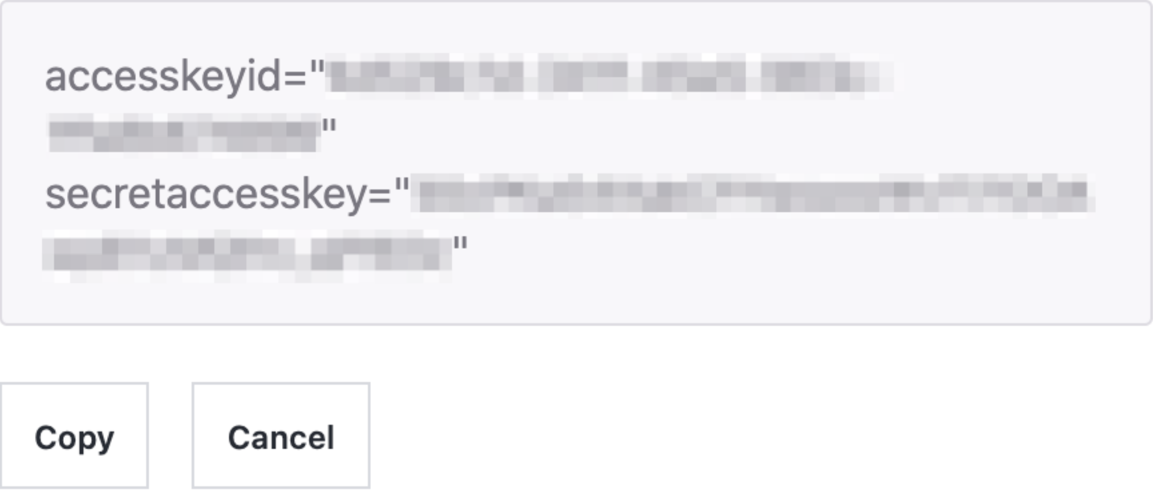
Windows Agent Installation

The Signal Sciences Agent is a small daemon process which provides the interface between your web server and our analysis platform. An inbound web request is passed to the agent, the agent then decides whether the requests should be permitted to continue or whether we should take action.

1. Create an empty agent configuration file at `C:\Program Files\Signal Sciences\Agent\agent.conf`.
    - If you need to specify a custom location for the `agent.conf` file, set the absolute file path with the system environment variable `SIGSCI_CONFIG`.
    - If you are deploying the agent in reverse proxy mode, see the Reverse Proxy Mode configuration page for details on required configuration options.
2. Configure the agent by inputting the **Agent Access Key** and **Agent Secret Key** into the agent configuration file at `C:\Program Files\Signal Sciences\Agent\agent.conf`.
    1. Log in to the Signal Sciences console.
    2. From the **Sites** menu, select a site if you have more than one site.
    3. Click **Agents** in the navigation bar. The agents page appears.
    4. Click **View agent keys**. The agent keys window appears.

    5. Copy the **Agent Access Key** and **Agent Secret Key**.



accesskeyid="[redacted]"
secretaccesskey="[redacted]"

Copy Cancel

6. Enter the **Agent Access Key** and **Agent Secret Key** into `C:\Program Files\Signal Sciences\Agent\agent.conf`.

```
accesskeyid = "AGENTACCESSKEYHERE"
secretaccesskey = "AGENTSECRETACCESSKEYHERE"
```

3. Download the latest Signal Sciences Windows Agent `.msi` from https://dl.signalsciences.net/?prefix=sigsci-agent/.

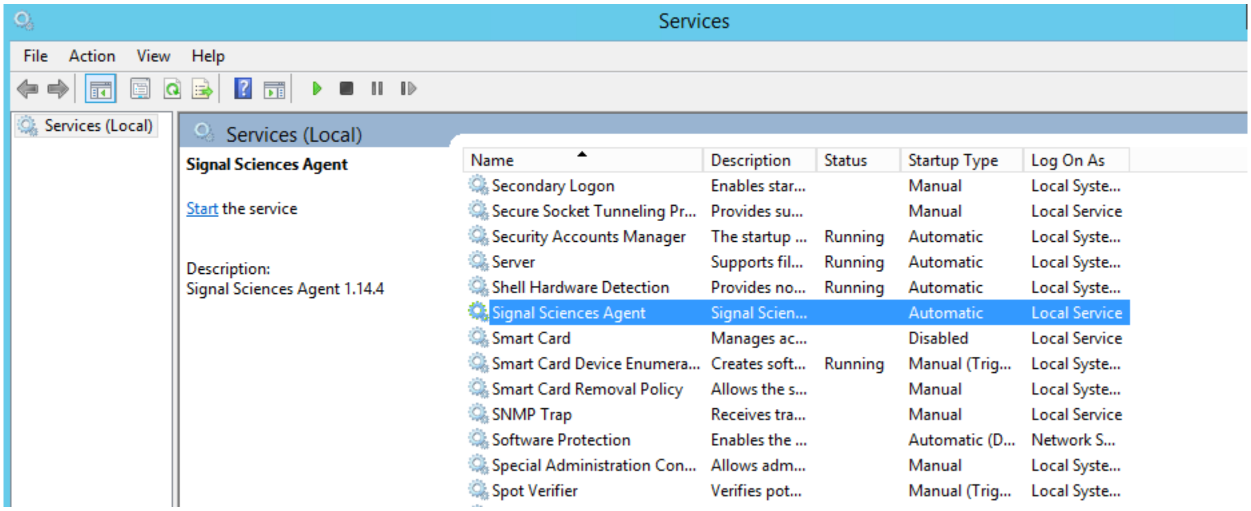
4. Run the `.msi` to install the Agent automatically with no prompts. It will install the executable in `C:\Program Files\Signal Sciences\Agent`, add a service entry for the Agent, and start the service if the agent configuration file is present with valid `accesskeyid` and `secretaccesskey` settings.

   The installed service name is `sigsci-agent` and can be controlled with PowerShell cmdlets:

   ```
   Start-Service sigsci-agent
   Restart-Service sigsci-agent
   Stop-Service sigsci-agent
   ```

   Alternatively, you can download the latest Signal Sciences Windows Agent as a `.zip` file, which contains the agent binary. You can run this from any location you prefer. However, to install the agent in this way, you will need to configure the Service entry and start the service manually.

Example `services.msc` screenshot:



Signal Sciences Agent

Start the service

Description:
Signal Sciences Agent 1.14.4

Name	Description	Status	Startup Type	Log On As
Secondary Logon	Enables star...		Manual	Local Syste...
Secure Socket Tunneling Pr...	Provides su...		Manual	Local Service
Security Accounts Manager	The startup ...	Running	Automatic	Local Syste...
Server	Supports fil...	Running	Automatic	Local Syste...
Shell Hardware Detection	Provides no...	Running	Automatic	Local Syste...
Signal Sciences Agent	Signal Scien...		Automatic	Local Service
Smart Card	Manages ac...		Disabled	Local Service
Smart Card Device Enumera...	Creates soft...	Running	Manual (Trig...	Local Syste...
Smart Card Removal Policy	Allows the s...		Manual	Local Syste...
SNMP Trap	Receives tra...		Manual	Local Service
Software Protection	Enables the ...		Automatic (D...	Network S...
Special Administration Con...	Allows adm...		Manual	Local Syste...
Spot Verifier	Verifies pot...		Manual (Trig...	Local Syste...

Next Steps

Explore our module options and install the Signal Sciences module.

Real Remote (Client) IP Addresses

remote IP address in an HTTP header that will be added to the request for other devices to use. The most common HTTP headers used for this are the `X-Forwarded-For` and `X-Real-Ip` headers. By default, the agent will take the real remote address from the `X-Forwarded-For` HTTP header when it is present, but the agent may need to be configured to use a different header (or none at all) in your environment. This (or another) HTTP header must be added by configuring the load balancer or proxy with access to the real remote address. In most cases this has already been done as it is generally required by other services as well.

To be the most compatible out of the box, the default for the agent is to take the real remote address from the `X-Forwarded-For` HTTP header. Without any additional configuration, the agent will use the remote address specified by this HTTP header. While this normally gives correct results, this method may not work in some environments that use a different header or another means of obtaining the real remote address.

Setting alternative headers in the console

You can set alternative client IP headers for the agent to source the real remote IP address directly from the console:

1. From the **Manage** menu, select **Site Settings**. The Site Settings menu page appears.
2. Click the **Agent Configurations** link. The Agent Configurations menu appears.
3. Under **Client IP Headers**, click the **Add header** button. A **Header** text box appears.
4. In the **Header** text box, enter the header name. Headers are not case sensitive.
5. If you want to add another header, click the **Add header** button again and enter another header name in the new **Header** text box.
6. Click the **Update** button.

You can specify up to 10 different headers. Headers will be used in order from top to bottom, meaning if the first header is not present in the request, the agent will proceed to check for the second header, and so on, until one of the listed headers is found. If none of the defined headers exist, or the value is not an IP address, then the agent will use the socket address.

> **Note:** Alternative client IP headers set in the console take priority and will override any alternative client IP headers set directly in the agent. Client IP headers set in the console do not currently apply to WebSocket inspection or agents deployed at the edge. The client IP header must be set directly in the agent.

Removing alternate headers in the console

1. From the **Manage** menu, select **Site Settings**. The Site Settings menu page appears.
2. Click the **Agent Configurations** link. The Agent Configurations menu appears.
3. Under **Client IP Headers**, click the **Delete header** button to the right of the header you want to delete.

Setting alternative headers directly in the agent

You can set alternative headers directly in the agent following the details below.

Alternative HTTP header

If your environment uses a different HTTP header to pass the real remote address, you will need to configure the agent to use that header. You can set an alternative header using the `client-ip-header` agent configuration option. For example, you can specify the agent use the `X-Real-IP` header by adding the following line to the `/etc/sigsci/agent.conf` file:

```
client-ip-header = "X-Real-Ip"
```

As this is such a common issue, most web servers offer an alternative module for interpreting the real remote address. If one of these is used, the remote address will be correctly passed to the agent and you will want to disable the agent from interpreting the default `X-Forwarded-For` header. If this is not done, then the agent may misinterpret the remote address. To do this, you will need to set the `client-ip-header` option to an empty value:

```
client-ip-header = " "
```

If the agent configuration is updated, the agent will then need to be restarted.

X-Forwarded-For header configuration

When a request is received, the agent will read the left-most IP address from the X-Forwarded-For (XFF) header.

For example, if a received request contains:

```
X_FORWARDED_FOR="127.0.0.1, 203.0.113.63"
```

The agent will report:

left instead. You can set the agent to read XFF IP addresses from right to left by setting the `local-networks` agent configuration option to `private`. Add the following line to your agent configuration file (by default at `/etc/sigsci/agent.conf`):

```
local-networks = "private"
```

By setting the `local-networks` option to `private`, the agent will instead read the IP addresses in the XFF header from right to left and choose the first non-local IP address. In the example above, the agent would then report:

```
203.0.113.63
```

Additional information about agent configuration options can be found here.

Alternatives with various web servers

There are a number of alternative modules for interpreting the real remote address. If one of these is used, be sure to disable the agent from interpreting the headers as outlined above.

NGINX - http_realip_module

The `http_realip_module` that is included with NGINX will allow you to extract the real IP from an HTTP header and use it internally. This performs some configurable validation and is far less prone to spoofing. In addition, the module seamlessly replaces the remote address so that NGINX will just do the right thing.

To use the `http_realip_module` in NGINX, you will need that module built into the binary. For Signal Sciences supplied binaries, this is already included (as is most vendor supplied NGINX binaries). However, if you are building NGINX from source, then you will need to configure NGINX to enable this module.

The NGINX documentation on this module provides more details.

The recommended configuration for this module is to set the `set_real_ip_from` directive to all trusted (internal) addresses or networks and enable recursion via the `real_ip_recursive` directive. For example, if your load balancer IP is `192.0.2.54` and is adding the `X-Forwarded-For` header, then you might use the following configuration in NGINX in either the `http` or `server` blocks:

```
set_real_ip_from  192.0.2.54;
real_ip_header    X-Forwarded-For;
real_ip_recursive on;
```

NGINX http_realip_module - Proxy Protocol

If your NGINX deployment is configured behind a load balancer or similar that communicates to NGINX over the proxy protocol, then the `real_ip_header` needs to be sourced from the `proxy_protocol` parameter. This can be configured in either the `http` or `server` blocks.

```
set_real_ip_from  192.0.2.54;
real_ip_header    proxy_protocol;
```

For more configuration guidance around this type of deployment, check out the NGINX documentation.

Apache Web Server 2.4+ - mod_remoteip

The `mod_remoteip` module that is included with Apache Web Server 2.4+ will allow you to extract the real IP from an HTTP header and use it internally. This performs some configurable validation and is far less prone to spoofing. In addition, the module seamlessly replaces the remote address so that the web server will just do the right thing.

To use the `mod_remoteip`, you will need to load the module and configure it.

The Apache documentation on this module provides more details.

The recommended configuration for this module is to set the `set_real_ip_from` directive to all trusted (internal) addresses or networks and enable recursion via the `real_ip_recursive` directive. For example, if your load balancer IP is `192.0.2.54` and is adding the `X-Forwarded-For` header, then you might use the following config:

```
# Load the module (see also a2enmod command)
LoadModule remoteip_module mod_remoteip.so

# Configure
```

Note: On Debian/Ubuntu, you will typically use the `a2enmod` command to enable the module vs. adding the LoadModule directive directly. For example:

```
sudo a2enmod remoteip
```

Apache Web Server 2.2 or less - various solutions

The Apache Web Server prior to 2.4 does not supply a module to interpret an HTTP header to get the real remote address. However, there are a number of third party modules that can be used similar to Apache Web Server 2.4+ above.

Take a look at one of these popular third party modules:

- `mod_realip2`
- `mod_extract_forwarded`
- `mod_rpaf`

Known issues

When managing real client IP addresses, keep the following in mind.

Google Container Engine

If you have downgraded or not upgraded Kubernetes in Google Container Engine (GKE) to at least Kubernetes v1.1, then you may not be able to get the real client IP address. The solution is to upgrade Kubernetes. See further notes on this below.

Kubernetes prior to v1.1

If you are using Kubernetes prior to v1.1, then currently the only non-beta load balancer option is their network load balancer. The network load balancer does not add the extra `X-Forwarded-For` header as the HTTPS load balancer. Because of this, the real remote address cannot be obtained. The HTTPS load balancer that does add in this support is currently in beta and should be available with Kubernetes v1.1.

- **Google Container Network Load Balancer:** https://cloud.google.com/container-engine/docs/load-balancer
- **Google Container HTTP Load Balancer (beta):** https://cloud.google.com/container-engine/docs/tutorials/http-balancer
- **Kubernetes Ingress Load Balancing:** https://kubernetes.io/docs/concepts/services-networking/ingress/#load-balancing

AWS Lambda

Fastly's Next-Gen WAF (powered by Signal Sciences) supports any Lambda function on Amazon Web Services (AWS). Our Lambda extension acts as an HTTP proxy between the AWS Lambda service and runtime and will allow or block traffic after inspecting the JSON payload of the web API event used by the Lambda runtime.

The Fastly WAF Lambda extension is configured by using the AWS Secrets Manager. You can download Fastly's WAF binaries to create a layer that a Lambda function can use.

Recommendations

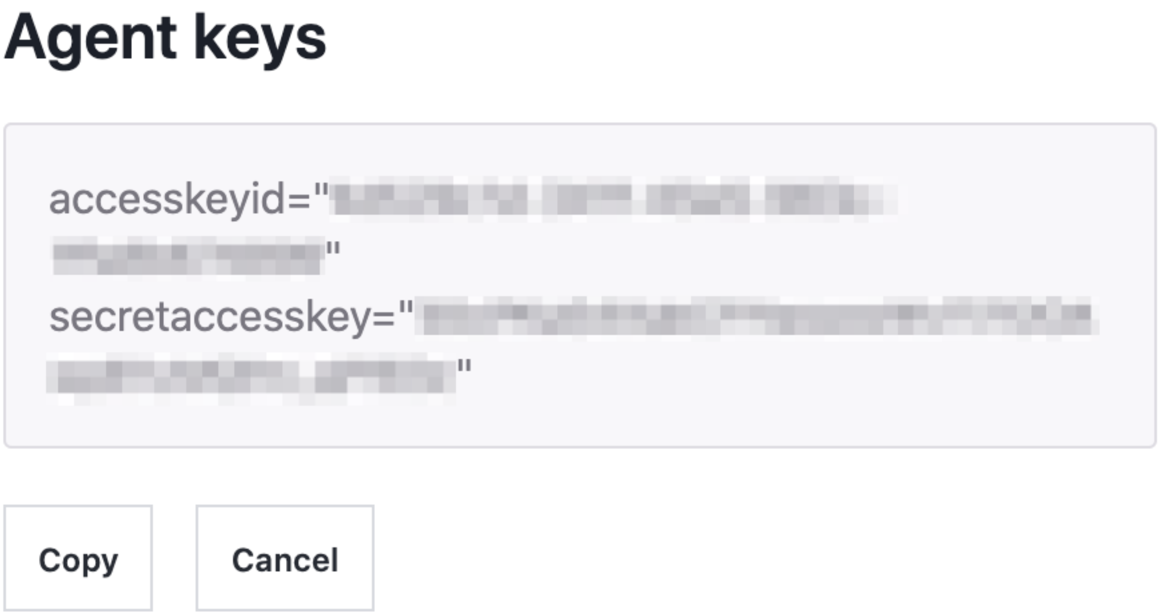
For reduced latency and improved performance, we recommend setting the memory for your Lambda function to at least 512 MB.

Configure the AWS Secrets Manager

1. Log in to the AWS Management Console.
2. From the **Services** menu, select **Security, Identify, & Compliance**, and then select **Secrets Manager**.
3. Click the **Store a new secret** button. The Choose secret type window appears.
4. For the Secret type, select **Other type of secret**. This option allows you to create a secret that can store credentials or other information by defining key-value strings.
5. Locate the **Agent Keys** for your Signal Sciences site:
    1. Log in to the Signal Sciences console.
    2. From the **Sites** menu, select a site if you have more than one site.
    3. Click **Agents** in the navigation bar. The agents page appears.

4. Click **View agent keys**. The agent keys window appears.

5. Copy the **Agent Access Key** and **Agent Secret Key**.



Agent keys

accesskeyid="[redacted]"
secretaccesskey="[redacted]"

6. In the AWS Management Console, enter the follow variables in the **Key/value pairs** fields:

Key	**Value**
`SIGSCI_ACCESSKEYID`	`accesskeyid` from Signal Sciences console
`SIGSCI_SECRETACCESSKEY`	`secretaccesskey` from Signal Sciences console

7. Click the **Next** button. The Configure secret window appears.

8. In the **Secret name** and **Description** fields, enter a human-readable name and description for the secret (e.g., `Fastly secret for Lambda extension`).

9. Locate the **Execute role** of your Lambda function:
   - In another tab, log in to the AWS Management Console.
   - From the **Services** menu, select **Compute**, and then select **Lambda**.
   - Select your Lambda function.
   - Click **Configuration**. The Configuration page appears.
   - From the sidebar, click **Permissions**, and then click the role name link for your Lambda function in the Execution role area.
   - From the Identity and Access Management (IAM) page that appears, copy the ARN displayed on the page.

10. Back on the Configure secret page in the AWS Management Console, click the **Edit permissions** button.

11. Modify the configuration shown below to allow your Lambda function role to access this secret.

```
{
    "Version" : "2012-10-17",
    "Statement" : [ {
        "Effect" : "Allow",
        "Principal" : {
            "AWS" : "arn:aws:iam::role/service-role/YOUR_LAMBDA_FUNCTION_ROLE"
        },
        "Action" : "secretsmanager:GetSecretValue",
        "Resource" : "*"
    } ]
}
```

12. Click the **Save** button, and then click the **Next** button. The Configure rotation page appears.

Configure the Fastly WAF Lambda extension

1. Log in to the AWS Management Console.
2. Click **Services**. Select **Compute**, then select **Lambda**.
3. Select your Lambda function.
4. Click **Configuration**. The Configuration menu pane appears.
5. Click **Environment variables**.
6. Click **Edit**. The Edit environment variables menu page appears.
7. Add the following variables in the **Key/value pairs** fields:

Key	Value
`SECRET_ARN`	Secret ARN of the newly created secret Example: arn:aws:secretsmanager:us-west-2:secret:lambda_secrets-kMxqBg
`SECRET_REGION`	Region where the newly created secret resides Example: us-west-2
`AWS_LAMBDA_EXEC_WRAPPER`	`/opt/sigsci-wrapper`
`SIGSCI_KEYSTORE_WRAPPER`	`/opt/fetch-aws-secrets` Only needed if using AWS Secrets Manager

8. Click **Save**.

Install the Fastly WAF Lambda extension

1. Download the latest version of the Agent for your particular architecture or use the public regional layer.

   x86_64

   ```
   AGENT_VER=`curl --fail  -Ss https://dl.signalsciences.net/sigsci-agent/VERSION`
   curl --fail -O -Ss https://dl.signalsciences.net/sigsci-agent/${AGENT_VER}/linux/sigsci-agent_${AGENT_VER}_la
   ```

   arm64

   ```
   AGENT_VER=`curl --fail  -Ss https://dl.signalsciences.net/sigsci-agent/VERSION`
   curl --fail -O -Ss https://dl.signalsciences.net/sigsci-agent/${AGENT_VER}/linux/sigsci-agent_${AGENT_VER}_la
   ```

   Lambda Layers

   `arn:aws:lambda:us-east-1:303561444828:layer:sigsci-agent-lambda_amd64:8`

   `arn:aws:lambda:us-east-1:303561444828:layer:sigsci-agent-lambda_arm64:11`

   `arn:aws:lambda:us-east-2:303561444828:layer:sigsci-agent-lambda_amd64:8`

   `arn:aws:lambda:us-east-2:303561444828:layer:sigsci-agent-lambda_arm64:8`

   `arn:aws:lambda:us-west-1:303561444828:layer:sigsci-agent-lambda_amd64:8`

   `arn:aws:lambda:us-west-1:303561444828:layer:sigsci-agent-lambda_arm64:8`

   `arn:aws:lambda:us-west-2:303561444828:layer:sigsci-agent-lambda_amd64:8`

   `arn:aws:lambda:us-west-2:303561444828:layer:sigsci-agent-lambda_arm64:8`

2. If the Lambda Agent is configured to retrieve secrets from the AWS Secrets Manager, add the appropriate regional layer, making sure this layer is ordered before the lambda extension.

   `arn:aws:lambda:us-east-1:303561444828:layer:sigsci-get-aws-secrets_amd64:1`

```
arn:aws:lambda:us-east-2:303561444828:layer:sigsci-get-aws-secrets_arm64:1

arn:aws:lambda:us-west-1:303561444828:layer:sigsci-get-aws-secrets_amd64:1

arn:aws:lambda:us-west-1:303561444828:layer:sigsci-get-aws-secrets_arm64:1

arn:aws:lambda:us-west-2:303561444828:layer:sigsci-get-aws-secrets_amd64:1

arn:aws:lambda:us-west-2:303561444828:layer:sigsci-get-aws-secrets_arm64:1
```

3. Publish the Lambda agent zip file as a layer *if downloaded*.

   > **Note:** An example is shown below using the AWS Command Line Interface. The layer name and compatible-runtimes are at your discretion.

   ```
   aws lambda publish-layer-version --layer-name "my-sigsci-lambda-layer" --zip-file "fileb://sigsci-agent_lates
   ```

4. Once the layer is successfully published, return to your Lambda function page within AWS.

5. Click **Add a layer** towards the bottom of the page in the **Layers** pane.

6. Add the layer that matches the published layer-name in the previous steps.

7. Click **Save**.

Troubleshooting

Take note of the ordering of the layers. If using the `sigsci-get-aws-secrets` layer, make sure it's ordered before the Lambda extension.

All of our agent logging can be found in the Lambda logs in AWS' CloudWatch. On the Lambda function page, select **Monitor**, then **View logs in CloudWatch**. Logs can be viewed and captured here.

In development environments, the Fastly WAF Lambda extension can use the `SIGSCI_ACCESSKEYID` and `SIGSCI_SECRETACCESSKEY` key/value pairs as environment variables in the Lambda function configuration to avoid using the AWS Secrets Manager. However, this is not recommended for production environments.

Kubernetes Ambassador

Installing with Ambassador Edge Stack (AES)

In this example, Signal Sciences is integrated with Ambassador Edge Stack, a cloud native API gateway and ingress controller for Kubernetes, built upon Envoy proxy.

Integrating the Signal Sciences Agent

The Signal Sciences Agent can be installed as a sidecar into each pod or as a service for some specialized needs.

The recommended way of installing the Signal Sciences Agent in Kubernetes is by integrating the `sigsci-agent` into a pod as a sidecar. This means adding the `sigsci-agent` as an additional container to the Kubernetes pod. As a sidecar, the agent will scale with the app/service in the pod instead of having to do this separately. However, in some situations, it may make more sense to install the `sigsci-agent` container as a service and scale it separately from the application.

The `sigsci-agent` container can be configured in various ways depending on the installation type and module being used.

You can use the `preStop` container hook to slow the pod's shutdown and ensure drain timeouts are met.

```
preStop:
  exec:
    command:
      - sleep
      - "30"
```

Getting and Updating the Signal Sciences Agent Container Image

An official `signalsciences/sigsci-agent` container image is available from the Signal Sciences account on Docker Hub.

even if one already exist locally. This is so the documentation does not fall out of date and anyone using this will not have an agent that stays stagnant. However, this may not be what if you need to keep installations consistent or on a specific version of the agent. In these cases, you should specify an agent version. Images on Docker Hub are tagged with their versions and a list of versions is available on Docker Hub.

Whether you choose to use the `latest` image or a specific version, there are a few items to consider to keep the agent up-to-date.

Using the `latest` Signal Sciences Container Image

If you do choose to use the `latest` image, then you will want to consider how you will keep the agent up to date.

- If you have used the `imagePullPolicy: Always` option, then the latest image will be pulled on each startup and your agent will continue to get updates.
- Alternatively, you may instead choose to manually update the local cache by periodically forcing a pull instead of always pulling on startup:

  ```
  docker pull signalsciences/sigsci-agent:latest
  ```

  Then, use `latest` with `imagePullPolicy: Never` set in the configuration so that pulls are never done on startup (only manually as above):

  ```
  - name: sigsci-agent
      image: signalsciences/sigsci-agent:latest
      imagePullPolicy: Never
      ...
  ```

Using a Versioned Signal Sciences Container Image

To use a specific version of the agent, replace `latest` with the agent version (represented here by `x.xx.x`). You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:x.xx.x
    imagePullPolicy: IfNotPresent
    ...
```

This will pull the specified agent version and cache it locally. If you use this method, then it is recommended that you parameterize the agent image, using Helm or similar, so that it is easier to update the agent images later on.

Using a Custom Tag for the Signal Sciences Container Image

It is also possible to apply a custom tag to a local agent image. To do this, pull the agent image (by version or use `latest`), apply a custom tag, then use that custom tag in the configuration. You will need to specify `imagePullPolicy: Never` so local images are only updated manually. After doing so, you will need to periodically update the local image to keep the agent up-to-date.

For example:

```
docker pull signalsciences/sigsci-agent:latest
docker tag signalsciences/sigsci-agent:latest signalsciences/sigsci-agent:testing
```

Then use this image tag in the configuration:

```
- name: sigsci-agent
    image: signalsciences/sigsci-agent:testing
    imagePullPolicy: Never
...
```

Configuring the Signal Sciences Agent Container

Agent configuration is normally done via the environment. Most configuration options are available as environment variables. Environment variables names have the configuration option name all capitalized, prefixed with `SIGSCI_` and any dashes (-) changed to underscores (_). For example, the max-procs option would become the `SIGSCI_MAX_PROCS` environment variable. For more details on what options are available, see the Agent Configuration documentation.

The `sigsci-agent` container has a few required options that need to be configured:

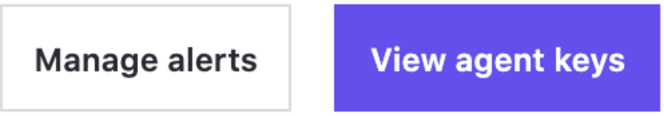
Agent Credentials

The `sigsci-agent` credentials are configured with two environment variables. These variables must be set or the agent will not start.

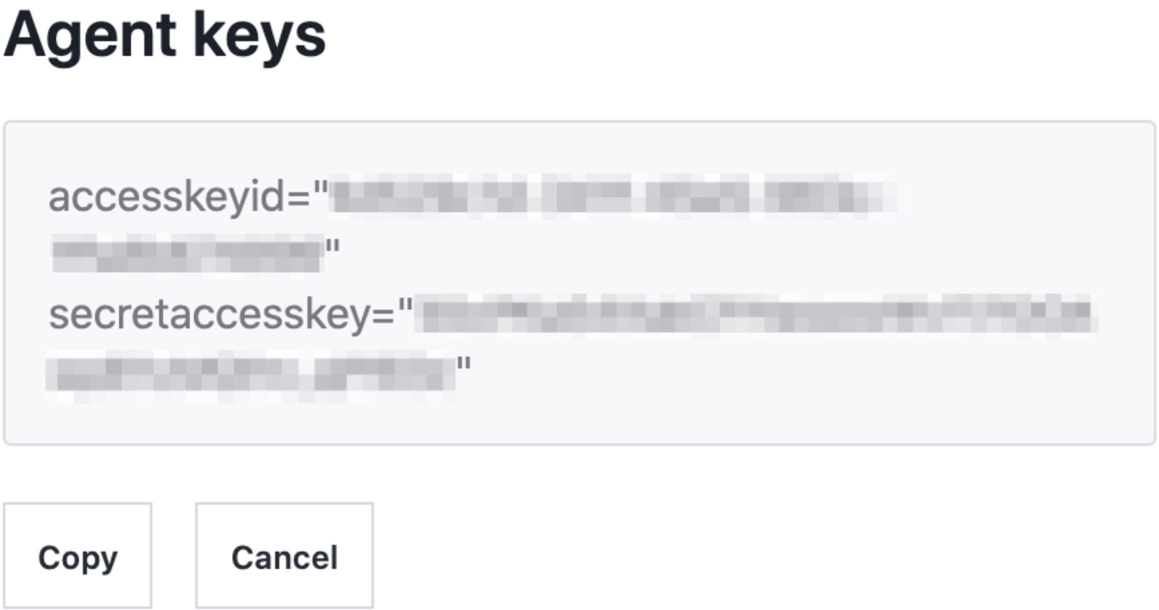
- **SIGSCI_ACCESSKEYID**: The **Agent Access Key** identifies which site in the Signal Sciences console that the agent is configured for.
- **SIGSCI_SECRETACCESSKEY**: The **Agent Secret Key** is the shared secret key to authenticate and authorize the agent.

The credentials can be found by following these steps:

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. Click **Agents** in the navigation bar. The agents page appears.
4. Click **View agent keys**. The agent keys window appears.



5. Copy the **Agent Access Key** and **Agent Secret Key**.


Agent keys

accesskeyid="[redacted]"
secretaccesskey="[redacted]"

Because of the sensitive nature of these values, we recommend you use the built in `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded values into the deployment configuration. This also makes any desired agent credential rotation easier to manage by having to change them in only one place.

Use the `valueFrom` option instead of the `value` option to use the `secrets` functionality. For example:

```
env:
  - name: SIGSCI_ACCESSKEYID
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: accesskeyid
  - name: SIGSCI_SECRETACCESSKEY
    valueFrom:
      secretKeyRef:
        # Update my-site-name-here to the correct site name or similar identifier
        name: sigsci.my-site-name-here
        key: secretaccesskey
```

```
apiVersion: v1
kind: Secret
metadata:
  name: sigsci.my-site-name-here
stringData:
  accesskeyid: 12345678-abcd-1234-abcd-1234567890ab
  secretaccesskey: abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

This can also be created from the command line with `kubectl` such as with the following example:

```
kubectl create secret generic sigsci.my-site-name-here \
  --from-literal=accesskeyid=12345678-abcd-1234-abcd-1234567890ab \
  --from-literal=secretaccesskey=abcdefg_hijklmn_opqrstuvwxy_z0123456789ABCD
```

Additional information about Kubernetes `secrets` functionality can be found here.

Agent Temporary Volume

For added security, we recommended the `sigsci-agent` container be executed with the root filesystem mounted as read only. However, the agent still needs to write some temporary files such as the socket file for RPC communication and some periodically updated files such as GeoIP data.

To accomplish this with a read only root filesystem, there needs to be a writeable volume mounted. This writeable volume can also be shared to expose the RPC socket file to other containers in the same pod.

The recommended way of creating a writeable volume is to use the builtin `emptyDir` volume type. This is typically configured in the `volumes` section of a deployment, as shown in the following example:

```
volumes:
  - name: sigsci-tmp
    emptyDir: {}
```

Containers will then mount this volume at `/sigsci/tmp`:

```
volumeMounts:
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
```

The default in the official agent container image is to have the temporary volume mounted at `/sigsci/tmp`. If this needs to be moved for the agent container, then the following agent configuration options should also be changed from their defaults to match the new mount location:

- `rpc-address` defaults to `/sigsci/tmp/sigsci.sock`
- `shared-cache-dir` defaults to `/sigsci/tmp/cache`

Integrating the Signal Sciences agent into Ambassador Edge Stack (AES)

The Signal Sciences Agent can be integrated with Datawire's Ambassador Edge Stack (AES). This integration uses the underlying Envoy integration built into the agent. The agent is configured with an Envoy gRPC Listener and through AES's Filter, FilterPolicy, and LogService Kubernetes resources. Deployment and configuration is flexible. As such, this guide is designed to provide information that can be applied to your own methods of deployment.

Note that the examples in the documentation will refer to installing the latest agent version, but this is only so that the documentation examples do not fall behind. Refer to the documentation on getting and updating the agent for more details on agent versioning and how to keep the agent up-to-date.

Namespaces

By default, AES is installed into the ambassador Kubernetes namespace. The agent and any applications running behind AES do not have to run in this namespace, but you must take care during configuration to use the correct namespaces as this documentation may differ from your configuration. The following namespaces are used in this documentation:

Ambassador

- Used for the ambassador install.
- Used for all ambassador resources (e.g., Filter, FilterPolicy, LogService, Mapping).

- Used for all applications and services running behind AES.
- Used for the agent when run in standalone mode.

Ambassador ID

Ambassador Edge Stack supports running multiple Ambassador Edge Stacks in the same cluster, without restricting a given Ambassador Edge Stack to a single namespace. This is done with the `ambassador_id` setting. When running multiple Ambassador Edge Stacks, care must be taken to include the correct `ambassador_id` value within all ambassador resources (Filter, LogService, Mapping, etc), otherwise the configuration will not be used. AES deployments only running one stack under the `default` ID don't need to set this value. Refer to the Ambassador ID docs for more information.

Running the agent as standalone or sidecar

The agent can run as a standalone deployment service or as a sidecar container within the AES pod. Either is fine, but running as a sidecar is easier if you are using Helm, as this is directly supported in the Helm values file. Running as a sidecar also has the advantage of scaling with AES, so this is the recommended route if you are using scaling via replica counts or autoscaling.

Installation

Installation involves two tasks: Deploying the agent configured in gRPC mode and Configuring AES to send traffic to the agent.

Deploying the agent

Deploying the agent is done by deploying the `signalsciences/sigsci-agent` container as a sidecar to AES or as a standalone service. The agent must be configured with its Agent Access Key and Agent Secret Key. This is typically done via a Kubernetes secret. One important point about secrets is that the secret must be in the same namespace as the pod using the secret. So, if you are running as a sidecar in the ambassador namespace, then the secret must also reside in that namespace. Refer to the agent credentials documentation for more details.

Example Secret in the ambassador namespace:

```
apiVersion: v1
kind: Secret
metadata:
  # Edit `my-site-name-here`
  # and change the namespace to match that which
  # the agent is to be deployed
  name: sigsci.my-site-name-here
  namespace: ambassador
stringData:
  # Edit these `my-agent-*-here` values:
  accesskeyid: my-agent-access-key-id-here
  secretaccesskey: my-agent-secret-access-key-here
```

Sidecar with Helm

Configuring AES with Helm is the easiest way to deploy, as the Ambassador values file already has direct support for this without having to modify an existing deployment YAML file. Refer to the AES documentation for installing with helm.

To install the agent as a sidecar, you will need to add new configuration lines to your custom values file, then install or upgrade AES with this values file. Refer to the Ambassador helm chart documentation for a reference on the values file. This will add the container with the correct configuration to the AES pod as a sidecar.

Add the following to the values YAML file:

```
sidecarContainers:
- name: sigsci-agent
  image: signalsciences/sigsci-agent:latest
  imagePullPolicy: IfNotPresent
  # Configure the agent to use Envoy gRPC on port 9999
  env:
  - name: SIGSCI_ACCESSKEYID
    valueFrom:
      secretKeyRef:
        # This secret needs added (see documentation on sigsci secrets)
        name: sigsci.my-site-name-here
      secretKeyRef:
        # This secret needs added (see documentation on sigsci secrets)
        name: sigsci.my-site-name-here
        key: secretaccesskey
  # Configure the Envoy to expect response data
  - name: SIGSCI_ENVOY_EXPECT_RESPONSE_DATA
    value: "1"
  # Configure the Envoy gRPC listener address on any unused port
  - name: SIGSCI_ENVOY_GRPC_ADDRESS
    value: localhost:9999
  ports:
  - containerPort: 9999
    name: grpc
  securityContext:
    # The sigsci-agent container should run with its root filesystem read only
    readOnlyRootFilesystem: true
    # Ambassador uses user 8888 by default, but the sigsci-agent container
    # needs to run as sigsci(100)
    runAsUser: 100
  volumeMounts:
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
volumes:
- name: sigsci-tmp
  emptyDir: {}
```

Example of upgrading AES with helm:

```
helm upgrade ambassador \
  --values /path/to/ambassador-sigsci_values.yaml \
  --namespace ambassador \
  datawire/ambassador
```

Alternatively, you can use Helm to render the manifest files. This makes adding the agent sidecar much easier than manually editing the YAML files. The modified deployment YAML will be in:

```
<output-dir>/ambassador/templates/deployment.yaml
```

Example of rendering the manifests with helm and applying the results:

```
helm template \
  --output-dir ./manifests \
  --values ./ambassador-sigsci_values.yaml \
  --namespace ambassador \
  datawire/ambassador
kubectl apply \
  --recursive
  --filename ./manifests/ambassador
```

Sidecar manually

Deploying the agent as a sidecar into the AES pod manually requires significantly more work than using Helm to render the manifests and is therefore not recommended.

You will need to modify the `aes.yaml` file, available at https://www.getambassador.io/yaml/aes.yaml. Append the container and volumes as described in the using Helm instructions. Refer to the AES installation guide and the Kubernetes and Envoy documentation for more details.

You will need to modify the following resource:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
```

```
…
  containers:
  …
  volumes:
  …
```

The container will need to be added to the `containers` section and the volume to the `volumes` section.

Standalone

To deploy a standalone agent, you only need to add a `Deployment` and `Service` resource for the agent, as shown in the following example:

```
apiVersion: v1
kind: Service
metadata:
  name: sigsci-agent
  # You may want it running in the ambassador namespace
  #namespace: ambassador
  labels:
    service: sigsci-agent
spec:
  type: ClusterIP
  ports:
  - name: sigsci-agent
    port: 9999
    targetPort: grpc
  selector:
    service: sigsci-agent
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sigsci-agent
  # You may want it running in the ambassador namespace
  #namespace: ambassador
spec:
  replicas: 1
  selector:
    matchLabels:
      service: sigsci-agent
  template:
    metadata:
      labels:
        service: sigsci-agent
    spec:
      containers:
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: IfNotPresent
        # Configure the agent to use Envoy gRPC on port 9999
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              # This secret needs added (see documentation on sigsci secrets)
              name: sigsci.my-site-name-here
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
```

```
        # Configure the Envoy to expect response data
        - name: SIGSCI_ENVOY_EXPECT_RESPONSE_DATA
          value: "1"
        # Configure the Envoy gRPC listener address on any unused port
        - name: SIGSCI_ENVOY_GRPC_ADDRESS
          value: 0.0.0.0:9999
        ports:
        - containerPort: 9999
          name: grpc
        securityContext:
          # The sigsci-agent should run with its root filesystem read only
          readOnlyRootFilesystem: true
        volumeMounts:
        - name: sigsci-tmp
          mountPath: /sigsci/tmp
      volumes:
      - name: sigsci-tmp
        emptyDir: {}
```

For more information, refer to the Kubernetes and Envoy documentation.

Sending traffic to the agent

You will need to configure three Ambassador resources for AES to send data to the agent. Refer to the Envoy configuration documentation for more detailed information on what each of these configures in the underlying Envoy install. The following guide uses the example `quote` service included with Ambassador.

Filter

The Filter resource is used to add the external authorization (`ext_authz`) filter to Envoy. This will inspect incoming requests that match the FilterPolicy.

The Signal Sciences agent requires AuthService to be defined in the Ambassador configuration, otherwise the agent will not receive request data. AuthService should be enabled by default. If requests are not being received by the agent, check that AuthService is enabled by running `kubectl get authservice`.

The namespace used for the `auth_service` configuration is the namespace the agent is deployed to. This guide uses the `ambassador` namespace for sidecar agents and `default` namespace for standalone agents. The format for the `auth_service` URL must be:

```
agent-hostname[.namespace]:agent-port
```

Examples:

- Sidecar: `auth_service: localhost:9999`
- Standalone: `auth_service: sigsci-agent.default:9999`

Example Filter YAML:

```
# Filter defines an external auth filter to send to the agent
kind: Filter
apiVersion: getambassador.io/v2
metadata:
  name: sigsci
  namespace: ambassador
  annotations:
    getambassador.io/resource-changed: "true"
spec:
  External:
    # Sidecar agent:
    auth_service: localhost:9999
    # Standalone sigsci-agent service in default namespace:
    #auth_service: sigsci-agent.default:9999
    path_prefix: ""
```

```
      max_bytes: 8192
      allow_partial: true
    failure_mode_allow: true
    timeout_ms: 100000
```

FilterPolicy

The FilterPolicy resource maps what paths will be inspected by the agent. You can map this to all traffic (`path: /*`) or subsets (`path: /app1/*`). However, there is a limitation that each subset must map to the same agent. This is due to a limitation on the LogService not having a path based filter like the FilterPolicy. The LogService must route all matching response data to the same agent that handled the request.

Example routing all traffic to the agent:

```
# FilterPolicy defines which requests go to sigsci
kind: FilterPolicy
apiVersion: getambassador.io/v2
metadata:
  namespace: ambassador
  name: sigsci-policy
  annotations:
    getambassador.io/resource-changed: "true"
spec:
  rules:
    - host: "*"
      # All traffic to the sigsci-agent
      path: "/*"
      filters:
        # Use the same name as the Filter above
        - name: sigsci
          namespace: ambassador
          onDeny: break
          onAllow: continue
          ifRequestHeader: null
          arguments: {}
```

You can route subsets of traffic to the agent with multiple rules. However every rule must go to the same agent due to the limitations described above.

Example routing subsets of traffic to the agent:

```
# FilterPolicy defines which requests go to the sigsci-agent
kind: FilterPolicy
apiVersion: getambassador.io/v2
metadata:
  namespace: ambassador
  name: sigsci-policy
  annotations:
    getambassador.io/resource-changed: "true"
spec:
  rules:
    # /app1/* and /app2/* to the sigsci-agent
    - host: "*"
      path: "/app1/*"
      filters:
        # Use the same name as the Filter above
        - name: sigsci
          namespace: ambassador
          onDeny: break
          onAllow: continue
          ifRequestHeader: null
      filters:
        # Use the same name as the Filter above
        - name: sigsci
          namespace: ambassador
          onDeny: break
          onAllow: continue
          ifRequestHeader: null
          arguments: {}
```

LogService

The LogService resource is used to add the gRPC Access Log Service to Envoy. This will inspect the outgoing response data and record this data if a signal was detected. It is also used for anomaly signals such as `HTTP_4XX` and `HTTP_5XX`.

The namespace used for the `service` configuration is the namespace the agent is deployed to. This guide uses the `ambassador` namespace for sidecar agents and `default` namespace for standalone agents. The format for the `service` URL must be:

```
agent-hostname[.namespace]:agent-port
```

Examples:

- Sidecar: `service: localhost:9999`
- Standalone: `service: sigsci-agent.default:9999`

Example:

```
# Configure the access log gRPC service for the response
# NOTE: There is no policy equiv here, so all requests are sent
apiVersion: getambassador.io/v2
kind: LogService
metadata:
  namespace: ambassador
  name: sigsci-agent
spec:
  # Sidecar agent
  service: localhost:9999
  # Standalone sigsci-agent service in default namespace:
  #service: sigsci-agent.default:9999
  driver: http
  driver_config:
    additional_log_headers:
    ### Request headers:
    # Required:
    - header_name: "x-sigsci-request-id"
      during_request: true
      during_response: false
      during_trailer: false
    - header_name: "x-sigsci-waf-response"
      during_request: true
      during_response: false
      during_trailer: false
    # Recommended:
    - header_name: "accept"
      during_request: true
      during_response: false
      during_trailer: false
    - header_name: "date"
      during_request: false
      during_response: true
      during_trailer: true
    - header_name: "server"
    ### Both request/response headers:
    # Recommended
    - header_name: "content-type"
      during_request: true
      during_response: true
      during_trailer: true
    - header_name: "content-length"
      during_request: true
      during_response: true
      during_trailer: true
  grpc: true
```

Red Hat NGINX 1.9 or lower

Add the package repositories

Add the version of the Red Hat CentOS package repository that you want to use.

Red Hat CentOS 8

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/8/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 7

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 6

> **Note:** After Q2 2017, RHEL6 and CentOS 6 will exit Production Phase 2 according to the Red Hat Enterprise Linux Life Cycle. Only limited critical security fixes will be issued. You will need to review the lifecycle document for details and plan appropriately.

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

ngx_lua module. Because most older versions of NGINX do not support dynamically loadable modules, you will likely need to rebuild NGINX from source.

To assist you, we provide pre-built drop-in replacement NGINX packages already built with the ngx_lua module. This is intended for users who prefer not to build from source, or who either use a distribution-provided package or an official NGINX provided package. These pre-built packages are built to support much older distributions and are not gpg signed.

Flavors

We support three flavors of NGINX. These flavors are based on what upstream package we've based our builds on. All our package flavors are built according to the official upstream maintainer's build configuration with the addition of the ngx_lua and ngx_devel_kit modules.

Our provided flavors are:

- **Distribution:** the distribution flavor is based off the official distribution-provided NGINX packages. For Debian-based Linux distributions (Red Hat and Debian) these are the based off the official Debian NGINX packages. For Red Hat based Linux distributions we've based them off the EPEL packages as neither Red Hat or CentOS ship an NGINX package in their default distribution.
- **Stable:** the stable flavor is based off the official NGINX.org stable package releases.
- **Mainline:** the mainline flavor is based off the official NGINX.org mainline package releases.

Flavor version support

The following versions are contained in the various OS and flavor packages:

OS	Distribution	Stable	Mainline
Red Hat/CentOS EL7	1.6.2	1.8.1	1.9.10
Red Hat/CentOS EL6	1.0.15	1.8.1	1.9.10

The versions are dependent on the upstream package maintainer's supported version.

Yum Repository setup for CentOS 7/RHEL 7

1. Create a file /etc/yum.repos.d/sigsci_nginx.repo with the following contents:

   - Distribution (CentOS 7/RHEL 7) flavor

     > **Note:** Our distribution release depends on the EPEL repository, you will need to ensure your system also has it installed.

     ```
     [sigsci_nginx]
     name=sigsci_nginx
     priority=1
     baseurl=https://yum.signalsciences.net/nginx/distro/el7/$basearch
     repo_gpgcheck=1
     gpgcheck=0
     enabled=1
     gpgkey=https://yum.signalsciences.net/nginx/gpg.key
     sslverify=1
     sslcacert=/etc/pki/tls/certs/ca-bundle.crt

     [sigsci-nginx-noarch]
     name=sigsci_nginx_noarch
     priority=1
     baseurl=https://yum.signalsciences.net/nginx/distro/el7/noarch
     repo_gpgcheck=1
     gpgcheck=0
     enabled=1
     gpgkey=https://yum.signalsciences.net/nginx/gpg.key
     sslverify=1
     sslcacert=/etc/pki/tls/certs/ca-bundle.crt
     ```

   - Stable (CentOS 7/RHEL 7) flavor

     ```
     [sigsci_nginx]
     name=sigsci_nginx
     priority=1
     ```

```
enabled=1
gpgkey=https://yum.signalsciences.net/nginx/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

   - Mainline (CentOS 7/RHEL 7) flavor

```
[sigsci_nginx]
name=sigsci_nginx
priority=1
baseurl=https://yum.signalsciences.net/nginx/mainline/el7/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://yum.signalsciences.net/nginx/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

2. Rebuild the `yum` cache for the Signal Sciences repository.

```
yum -q makecache -y --disablerepo=* --enablerepo=sigsci_*
```

3. Install the version of NGINX provided by Signal Sciences.

```
yum install nginx
```

Yum repository setup for CentOS 6/RHEL 6

To configure your yum repository on your Red Hat or CentOS systems:

1. Create a file `/etc/yum.repos.d/sigsci_nginx.repo` with the following contents:

   - Distribution (CentOS 6/RHEL 6) flavor

> **Note:** Our distribution release depends on the EPEL repository, you will need to ensure your system also has it installed.

```
[sigsci_nginx]
name=sigsci_nginx
priority=1
baseurl=https://yum.signalsciences.net/nginx/distro/el6/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://yum.signalsciences.net/nginx/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt

[sigsci-nginx-noarch]
name=sigsci_nginx_noarch
priority=1
baseurl=https://yum.signalsciences.net/nginx/distro/el6/noarch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://yum.signalsciences.net/nginx/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

   - Stable (CentOS 6/RHEL 6) flavor

```
[sigsci_nginx]
name=sigsci_nginx
priority=1
baseurl=https://yum.signalsciences.net/nginx/stable/el6/$basearch
```

```
gpgkey=https://yum.signalsciences.net/nginx/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

- Mainline (CentOS 6/RHEL 6) flavor

  ```
  [sigsci_nginx]
  name=sigsci_nginx
  priority=1
  baseurl=https://yum.signalsciences.net/nginx/mainline/el6/$basearch
  repo_gpgcheck=1
  gpgcheck=0
  enabled=1
  gpgkey=https://yum.signalsciences.net/nginx/gpg.key
  sslverify=1
  sslcacert=/etc/pki/tls/certs/ca-bundle.crt
  ```

2. Rebuild the `yum` cache for the Signal Sciences repository.

   ```
   yum -q makecache -y --disablerepo=* --enablerepo=sigsci_*
   ```

3. Install the version of NGINX provided by Signal Sciences.

   ```
   yum install nginx
   ```

Check Lua is loaded correctly

To verify Lua has been loaded properly load the following config (`sigsci_check_lua.conf`) with NGINX:

```
# Config just to test for lua jit support
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
# Also, for some nginx-1.10.nn installed from nginx-extras package, you may
# need to specify the load directives.
# Given the above uncomment the following:
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768;
    # multi_accept on;
}
http {
init_by_lua '

local m = {}
local ngx_lua_version = "dev"

if ngx then
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
```

```
  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
  end

end

';

}
```

If the config is successfully loaded, the above script will create the following output:

```
$ nginx -t -c <your explicit path>/sigsci_check_lua.conf

nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
nginx: the configuration file <your explicit path>/sigsci_check_lua.conf syntax is ok
nginx: configuration file <your explicit path>/sigsci_check_lua.conf test is successful
```

Install the NGINX module

1. Install the module with `yum`.

   ```
   yum install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX service to initialize the new module.

   - CentOS 7/RHEL 7

     ```
     systemctl restart nginx
     ```

   - CentOS 6/RHEL 6

     ```
     restart nginx
     ```

.Net Core Module Install

Requirements

- .NET Core 2.1 or later.
- Verify you have installed the Signal Sciences agent for your platform (e.g., Linux or Windows). See Agent Installation instructions.

Installation

1. Download the latest SigSci HTTP middleware using one these methods:

   - Directly from https://dl.signalsciences.net/?prefix=sigsci-module-dotnetcore/
   - Via Nuget

2. Add the SigSci HTTP middleware to your project. Replace `<packagePath>` with the path to `SignalSciences.HttpMiddleware.<version>.nupkg` and `<sourcePath>` with the folder-based package source to which the package will be added:

   ```
   nuget add <packagePath> -Source <sourcePath> -Expand
   dotnet add package SignalSciences.HttpMiddleware -s <sourcePath>
   ```

```
    "SigsciOptions": {
        "AgentEndPoint": "127.0.0.1:2345"
    }
}
```

4. Configure the HTTP request pipeline with `Configure`:

```
Configure(IApplicationBuilder app, IHostingEnvironment env) {
    var sigsciOptions = Configuration.GetSection("SigsciOptions").Get<SigSciOptions>();
    app.UseSigSciHandler(sigsciOptions);
}
```

5. Restart the web site service.

> **Note:** Ensure the `AgentEndPoint` value is set to the same IP and port configured with the Signal Sciences agent's `rpc-address` value. See the Windows agent installation documentation for additional information about Windows agent configuration options.

.NET Core module configuration

Option	Default	Description
`AgentEndPoint`	required, no default	The TCP endpoint (`host:port`) that the Agent is listening on. `host` can be either a hostname or an IPv4 or IPv6 address.
`AgentRpcTimeoutMillis`	optional, default: 200	Maximum number of milliseconds allowed for each RPC call to the Agent.
`MaxPostSize`	optional, default: 100000	A request body above this size will not be sent to the Agent.
`AnomalySize`	optional, default: 524288	If the HTTP response is this size or larger, log it with the Agent.
`AnomalyDurationMillis`	optional, default: 1000	If the response took longer than this number of milliseconds, log it with the Agent.
`ExpectedContentTypes`	optional, no default	Adds custom types that allow inspection to the conditional content-type list.

Sample advanced .NET Core module configuration

```
{
        "SigsciOptions": {
                "AnomalySize": 200000,
                "AgentRPCTimeoutMillis": 200,
                "MaxPostSize": 50000,
                "AnomalyDurationMillis": 1000,
                "AgentEndPoint": "127.0.0.1:2345",
                "ExpectedContentTypes": "application/custom-abc application/hal-test"
        }
}
```

Working with Multiple Lua Scripts in NGINX

Currently, NGINX only supports one `init_by_lua` or `init_by_lua_file`, which is used by the Signal Sciences NGINX module. If you have your own Lua scripts embedded within NGINX, you will need to splice the Signal Sciences module into your custom Lua code.

> **Note:** By not using the `sigsci.conf` configuration file, you will not receive configuration file updates when the module is upgraded. You should take care and review your Lua module when a Signal Sciences module release is updated.

Removing the Signal Sciences NGINX Lua Module

Before you add our module into your existing Lua code, you'll need to remove any references to the `sigsci` include file: Look for and remove any lines that look like:

```
include /opt/sigsci/nginx/sigsci.conf;
```

Next, the following should be added to your NGINX configuration:

Within your `init_by_lua` or the file specified by `init_by_lua_file`, include the following snippet:

```
package.path = "/opt/sigsci/nginx/?.lua;" .. package.path
sigsci = require("SignalSciences")
```

Lastly, you will need to add an `access_by_lua` and `log_by_lua` into your NGINX configuration. If you already have these directives defined, copy the `sigsci.prerequest()` and `sigsci.postrequest()` statements to their respective Lua callers.

```
access_by_lua 'sigsci.prerequest()';
log_by_lua    'sigsci.postrequest()';
```

After adding those lines to your custom Lua scripts, restart NGINX.

Fastly Security Labs

Fastly Security Labs is a program that grants your Signal Sciences corp access to in-development beta features. In addition to early access to these upcoming features, you will also have the opportunity to provide regular feedback to help shape them as they develop.

> **Note:** Features included in the Fastly Security Labs program may be part of a Beta release. The status of each feature will be specified in the documentation for that feature. For more information, read our product and feature lifecycle descriptions.

Enrolling

Customers on the Professional or Premier platforms are eligible for participation in Fastly Security Labs. To participate, contact our support team.

Opting out of features

Your corp will be subscribed to all features by default. You can choose to opt out of specific features by following these steps:

1. Log in to the Signal Sciences console.
2. From the **Corp Manage** menu, select **User Authentication**. The User Authentication page appears.
3. In the **Fastly Security Labs** section, deselect the features to opt out of.
4. Click the **Update labs** button.

Limitations

Because Fastly Security Labs features are still in development, issues related to these features may need to be escalated to our development team for troubleshooting. As a result, these features are not covered by our support SLA because issue response and resolution times may take longer than typically expected.

Agent Scaling and Running as a Service

Scaling the agent

If the `sigsci-agent` is installed as a sidecar into a pod, the agent will scale however you have chosen to scale the application in the pod. This is the recommended method of installing the agent as it does not require a different means of scaling your application. However, for some installations the agent may need to be scaled at a different rate than the application. In these cases you can install the agent as a service to be used by the application pods. However, there are limitations when installing the agent as a service.

Limitations

- The `sigsci-agent` can only be configured for a single site. This means that any agent service would only be able to send to a single site. All of the agents in the service will have the same configuration.
- The `sigsci-agent` keeps some request states when processing the responses. This means that the agent that processed the request data needs to be the same agent that processes the response data. Therefore, load balancing agents require affinity, which makes the service more complex to scale.
- Using the `sigsci-agent` as a service means configuring the communication channel as TCP instead of a Unix domain socket and this is slightly less efficient.

Installing the Signal Sciences agent as a service

The `sigsci-agent` can be installed as a service, but care must be taken when configuring the service due the above limitations. The service will be tied to a single site. If you will have multiple sites, then you should name the service based on the Signal Sciences site name. To scale

Client IP:

Below is an example service tied to a site named my-site-name using Client IP affinity:

```
apiVersion: v1
kind: Service
metadata:
  name: sigsci-agent-my-site-name
  labels:
    app: sigsci-agent-my-site-name
spec:
  ports:
  # Port names and numbers are arbitrary
  #  737 is the default RPC port
  #  8000 may be more appropriate for gRPC used with Envoy
  - name: rpc
    port: 737
    targetPort: 737
  selector:
    app: sigsci-agent-my-site-name
  sessionAffinity: ClientIP
  sessionAffinityConfig:
    clientIP:
      timeoutSeconds: 60
```

The service must then be backed by a deployment with any number of replicas. The `sigsci-agent` container must be configured as in a typical sidecar install, but must use TCP instead of a shared Unix domain socket. This is done by setting the `SIGSCI_RPC_ADDRESS` configuration option. Note that if using this with Envoy, you must use `SIGSCI_ENVOY_GRPC_ADDRESS` instead.

Example deployment corresponding with the service above:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sigsci-agent-my-site-name
  labels:
    app: sigsci-agent-my-site-name
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sigsci-agent-my-site-name
  template:
    metadata:
      labels:
        app: sigsci-agent-my-site-name
    spec:
      containers:
      - name: sigsci-agent
        image: signalsciences/sigsci-agent:latest
        imagePullPolicy: IfNotPresent
        env:
        - name: SIGSCI_ACCESSKEYID
          valueFrom:
            secretKeyRef:
              name: sigsci.my-site-name
              key: accesskeyid
        - name: SIGSCI_SECRETACCESSKEY
          valueFrom:
            secretKeyRef:
```

```
        - name: SIGSCI_RPC_ADDRESS
          value: "0.0.0.0:737"
        # Use all available resources.limits.cpu cores
        - name: SIGSCI_MAX_PROCS
          value: "100%"
        securityContext:
          readOnlyRootFilesystem: true
        volumeMounts:
        - name: sigsci-tmp
          mountPath: /sigsci/tmp
        # Set CPU resource limits (required for autoscaling)
        resources:
          limits:
            cpu: 4
          requests:
            cpu: 1
      volumes:
      - name: sigsci-tmp
        emptyDir: {}
```

The above example will deploy two `sigsci-agent` pods for the `sigsci-agent-my-site-name` service to use for the `my-site-name` Signal Sciences site. Each agent will see up to 4 CPU cores, requiring resources for at least one core.

Each application pod must then have its module configured to send to a `sigsci-agent` at the service name and port defined by the service. In this example the module would be configured to sent to host `sigsci-agent-my-site-name` and port `737`. These values are defined by the service as well as the `SIGSCI_RPC_ADDRESS` configuration option (or `SIGSCI_ENVOY_GRPC_ADDRESS` if Envoy is being used).

As for scaling, each pod that connects to this service will be assigned a `sigsci-agent` running in the service and affinity will be locked to this agent. If the agent is then updated or otherwise removed from the service (such as due to an autoscaling down event) the agent will be reassigned to the client application pod. Because of how agents are assigned to pods with affinity, the maximum number of active agents will not be more than the number of pods connecting to the service. This should be considered when determining the number of replicas and autoscaling parameters.

The deployment can be autoscaled. As an example, it is possible to autoscale with a Horizontal Pod Autoscaler via `kubectl autoscale`. In the example below, the deployment will use a minimum of 2 agents and be scaled up to 6 agents whenever the overall CPU usage reaches 60%. Note again, however, that all of these agents will only be handling a single Signal Sciences site.

```
kubectl autoscale deployment sigsci-agent-my-site-name --cpu-percent=60 --min=2 --max=6
```

The status of the Horizontal Pod Autoscaler can be viewed via the `kubectl get hpa` command:

```
$ kubectl get hpa
NAME                      REFERENCE                            TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
sigsci-agent-my-site-name Deployment/sigsci-agent-my-site-name 42%/60%   2         6         2          53m42s
```

There are some limitations to this type of scaling. When scaling (by manually setting the replica number or autoscaling), the `sigsci-agent` pod count will change for the service. When an agent is added, new connections to the service may get assigned affinity to new agent pods, but note that application pods that already have their affinity set to a specific agent pod will not be rebalanced unless the service setting for the affinity timeout (`sessionAffinityConfig.clientIP.timeoutSeconds`) is hit. Because of this, this scaling works best when the application pods are also scaled so that new application pods will get balanced to new agent pods. Similarly, when an agent pod is removed from the service due to scaling down, the application pods that were assigned to this agent will be reassigned to another agent and affinity set. When scaling back up, these *will not get rebalanced*. If this occurs often, then you may consider reducing the affinity timeout (`sessionAffinityConfig.clientIP.timeoutSeconds`) to allow for rebalancing if there is some idle time.

Red Hat NGINX-Plus

Add the package repositories

Add the Signal Sciences `yum` repositories.

Red Hat Stream CentOS 9

```
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/9/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 8

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/8/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 7

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 6

> **Note:** After Q2 2017, RHEL6 and CentOS 6 will exit Production Phase 2 according to the Red Hat Enterprise Linux Life Cycle. Only limited critical security fixes will be issued. You will need to review the lifecycle document for details and plan appropriately.

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Install the NGINX module

1. Install the Signal Sciences NGINX module by running the following command:
   - NGINX Plus 29

     ```
     sudo apt-get install nginx-module-sigsci-nxp=1.23.4*
     ```

- NGINX Plus 27

  ```
  sudo yum install nginx-module-sigsci-nxp-1.21.6*
  ```

- NGINX Plus 26

  ```
  sudo yum install nginx-module-sigsci-nxp-1.21.5*
  ```

- NGINX Plus 25

  ```
  sudo yum install nginx-module-sigsci-nxp-1.21.3*
  ```

- NGINX Plus 24

  ```
  sudo yum install nginx-module-sigsci-nxp-1.19.10*
  ```

- NGINX Plus 23

  ```
  sudo yum install nginx-module-sigsci-nxp-1.19.5*
  ```

- NGINX Plus 22

  ```
  sudo yum install nginx-module-sigsci-nxp-1.19.0*
  ```

- NGINX Plus 21

  ```
  sudo yum install nginx-module-sigsci-nxp-1.17.9*
  ```

- NGINX Plus 20

  ```
  sudo yum install nginx-module-sigsci-nxp-1.17.6*
  ```

- NGINX Plus 19

  ```
  sudo yum install nginx-module-sigsci-nxp-1.17.3*
  ```

- NGINX Plus 18

  ```
  sudo yum install nginx-module-sigsci-nxp-1.15.10*
  ```

- NGINX Plus 17

  ```
  sudo yum install nginx-module-sigsci-nxp-1.15.7*
  ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the `pid /run/nginx.pid;` line:

   ```
   load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
   ```

3. Restart the NGINX service to initialize the new module.

   - CentOS 7/RHEL 7 or higher

     ```
     systemctl restart nginx
     ```

   - CentOS 6/RHEL 6

     ```
     restart nginx
     ```

Cisco Threat Response / SecureX

Cisco Threat Response (CTR) is a tool used by incident responders that aggregates data from various Cisco security products like AMP for Endpoints, Firewall, Umbrella, Email Security, and Stealthwatch in addition to data from certain third party products including Signal Sciences. Within CTR, an investigator can perform a lookup against some object (file hash, URL, IP address) and CTR will fetch data from all of the products that are integrated including any indicators of compromise and associated metadata.

Installation

Note: The user setting up the CTR integration must have permission to create API Access Tokens.

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. Create an API Access Token for your user.
4. Generate an **Authorization Bearer Token** from this API Access Token by base64 encoding a string composed of the email address associated with your user, a colon, and the API Access Token you generated. An example of this in Javascript is:

   ```
   btoa("user@example.com:api-access-token") = "YW5keUBleGFtcGxlY29ycC5jb206ZXhhbXBsZXRva2Vu"
   ```

5. Log in to your SecureX console.
6. Click the **Integrations** tab. The integrations menu page appears.
7. From the **Integrations** menu in the navigation bar on the left, select **Available Integrations**. The list of available integrations appears.
8. Locate the **Signal Sciences Next-Gen WAF** in the list of available modules and click **Add New Module**. The add new module menu page appears.
9. In the **Module Name** field, leave the default name or enter a custom name. Custom names are useful if you plan to have multiple integrations for several cloud instances.
10. In the **URL** field, enter `https://dashboard.signalsciences.net/api.v0/corps/<corpname>/ctr`.
    - Your `<corpname>` is present in the address of your Signal Sciences console, such as `https://dashboard.signalsciences.net/corps/<corpname>/overview`.
    - Your `<corpname>` can also be retrieved from the List Corps API endpoint.Your corp name is the string that appears in the URL after logging into the Signal Sciences console).
11. In the **Authorization Bearer Token** field, enter the base64-encoded token you generated in Step 3.
12. Click **Save**.

Using the Cisco Threat Response Integration

Once the integration is installed, any lookups within CTR that include an IP address that's been flagged by SigSci will return a record of the event in the Observables widget under Sightings and Indicators.

The Sighting will show when the IP address was flagged, the URL that was targeted, and a link back to the flagged IP address event within the SigSci console. The Indicator will describe the attack signal that was associated with the flagged IP address (i.e., XSS).

Pivotal Container Services (PKS) Setup

The Signal Sciences Pivotal Container Service (PKS) integration is set up in almost the same manner as a generic Kubernetes install. The main difference is access to the Kubernetes cluster for PKS is done by logging in via the provided `pks` client binaries from the PKS install.

Installation

There is nothing specific to do to integrate with PKS. Integration is the same as a generic Kubernetes install. The only difference is access to the Kubernetes cluster for PKS which is done by logging in via the provided `pks` client binaries from the PKS install. Additional documentation for PKS can be found here.

1. Set up your environment.

   ```
   # Credentials filename
   export KUBECONFIG=pks-creds.yaml
   ```

2. Log in to PKS using your URL and your username and password.

   ```
   pks login -a <your-url> -u <user> -p <password> -k
   ```

3. Create the credentials file (from KUBECONFIG).

   ```
   pks get-credentials <cluster-name>
   ```

5. Deploy your application following normal Kubernetes instructions. Confirm the configuration has been set up correctly by running commands on the remote cluster, such as listing the pods:

   ```
   kubectl get pods
   ```

6. Install Signal Sciences by following the instructions for integrating Signal Sciences into Kubernetes.

Debian NGINX 1.14.1+

Add the package repositories

Add the version of the Debian package repository that you want to use.

Debian 11 - Bullseye

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ bullseye
sudo apt-get update
```

Debian 10 - Buster

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ buster ma
sudo apt-get update
```

Debian 9 - Stretch

```
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ stretch main
EOF
sudo apt-get update
```

Debian 8 - Jessie

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ jessie main
EOF
sudo apt-get update
```

Debian 7 - Wheezy

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ wheezy main
EOF
sudo apt-get update
```

Install the module with apt

> **NOTE:** If you are using the backports repository with Debian 9, you will want to install the `nginx-module-sigsci-bp-nxo` module.

1. Install the Signal Sciences NGINX module by running the following command, replacing `NN.NN` with your NGINX version number:

   ```
   sudo apt-get install nginx-module-sigsci-nxo=1.NN.NN*
   ```

```
load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
```

3. Restart the NGINX service to initialize the new module.

```
sudo service nginx restart
```

About data storage and privacy

We store and make available request and response data via the web interface and API. Due to our redaction process, only non-sensitive or benign portions of the request are ever sent to the platform backend.

Limitations and considerations

Keep these things in mind:

- Data can only be extracted within 24 hours of its creation.
- We store request and response data for 30 days and then delete it.
- We use the collected request data to help identify and block attacks to your web application. We never attribute any data back to your organization or end users.

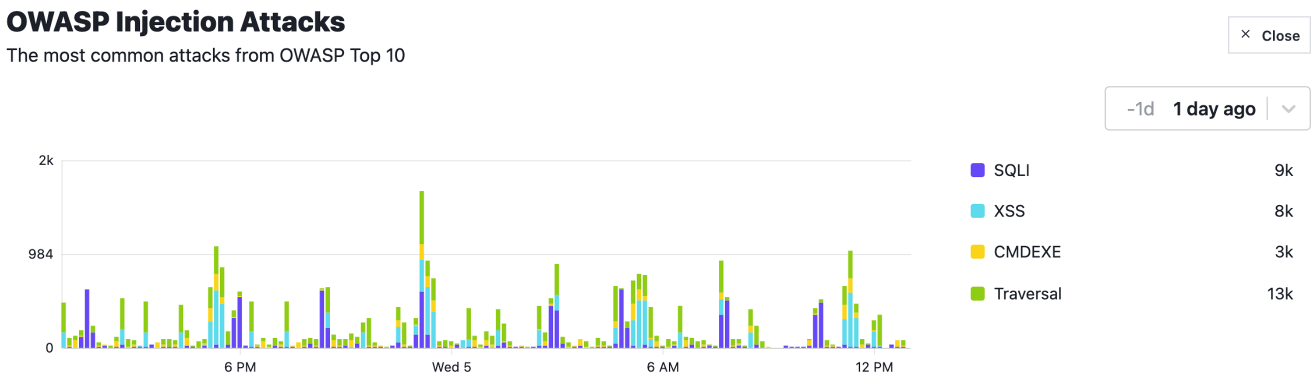
Response data storage

We only collect metadata (e.g., response codes and response headers) from response records.

Request data storage

From request records, we collect and store two types of data:

- **Time series data:** the number of signals (e.g., XSS, SQLi, 404s) observed per minute. All time series data is available via graphs in the web interface.

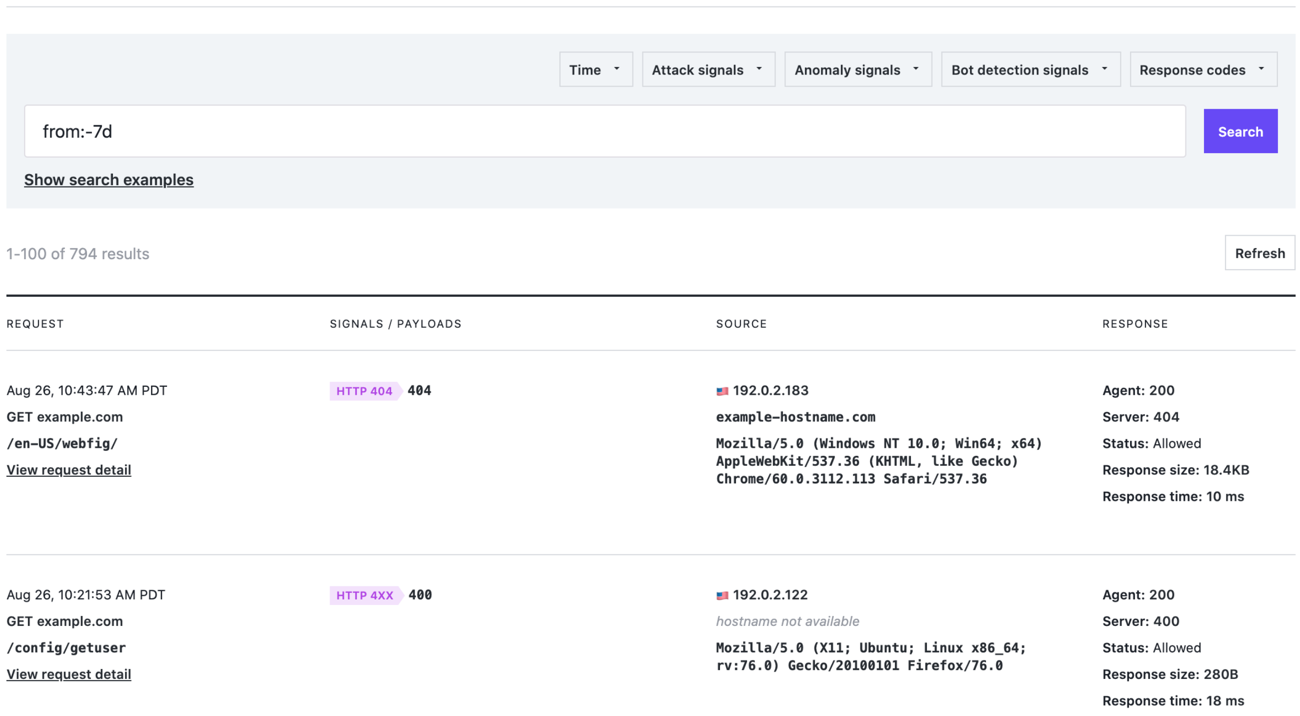


OWASP Injection Attacks

The most common attacks from OWASP Top 10

-1d 1 day ago

- **Individual request data:** detailed information about requests (e.g., originating IP address and request parameters). We store individual request data based on storage categories, site alerts, and the value of the **Request logging** setting for request rules.



from:-7d

1-100 of 794 results

REQUEST	SIGNALS / PAYLOADS	SOURCE	RESPONSE

Aug 26, 10:43:47 AM PDT
GET example.com
`/en-US/webfig/`
View request detail
	HTTP 404 404	192.0.2.183
		`example-hostname.com`
		`Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36`
			Agent: 200
			Server: 404
			Status: Allowed
			Response size: 18.4KB
			Response time: 10 ms

Aug 26, 10:21:53 AM PDT GET example.com `/config/getuser` View request detail	HTTP 4XX 400	192.0.2.122 *hostname not available* `Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0`	Agent: 200 Server: 400 Status: Allowed Response size: 280B Response time: 18 ms

How request data storage works

When requests are made to your web application, the Signal Sciences agent tags the requests with the appropriate signals and sends the signals to our cloud-hosted collection and analysis system. The system then counts the number of requests that were tagged with a particular signal during one minute periods and makes this data available via time series graphs in the web interface.

The Signal Sciences agent also determines which incoming requests we should store individual request data for. Individual request data is detailed information about a request record (e.g., originating IP address and parameters). To identify the requests that need capturing, the agent uses:

- the value of the **Request logging** menu from request rules. Specifically, we log requests that meet the criteria of a request rule with a **Request logging** value of `Sampled`.
- site alerts when the agent mode is `Blocking` or `Not blocking`. Specifically, when a system site alert flags an IP address, we log a sample of subsequent requests that are tagged with an attack signal and that are from that IP address. When a system site alert flags an IP address, we log a sample of the subsequent requests from that IP address.
- storage categories, which are based on signal type. For example, we store the individual request data for all requests that are tagged with the `SQLI` attack signal because requests that are tagged with an attack signal fall into the all storage category.

After identifying the requests that need capturing, the agent redacts sensitive data from the selected requests. By default, the agent redacts certain data (e.g., passwords, session tokens, and tracking cookies). The agent also redacts custom fields that you identify. For example, if your password field is named `foobar` instead of `password`, you can create a custom redaction for the `foobar` field.

Next, the agent sends the redacted requests to our system, and our system makes the individual request data available via the web interface and API.

We store both the time series data and the individual request data for 30 days and then delete it.

Storage categories

Storage categories help determine which request records we store individual request data for. They are based on the type of signals that requests are tagged with.

Storage category	Category applies to	What data is stored
All	Requests that contain at least one attack signal (e.g., SQLi and XSS) or one CVE signal applied by a virtual patching rule	We store individual request data and time series data from all requests that fit into this storage category.

Sampled	Requests that don't fit into the all storage category and that contain at least one custom signal or one anomaly signal (e.g., HTTP 404 Errors and Tor traffic)	We store individual request data from a random sample of requests that fit into this storage category. We also store time series data from all requests that fit into this storage category.
Time series only	Requests that only contain informational signals from API or ATO templated rules	We don't store individual request data from requests that fit into this storage category. However, we store time series data from all requests that fit into this storage category.
Not stored	Requests that aren't tagged with a signal	We don't store individual request data from requests that fit into this storage category.

Deleting stored data

If you find information in the raw data that you want to delete, submit a support request with the date range that you want us to scrub.

About rules

Rules are configurations that define when the Next-Gen WAF should:

- allow, block, rate limit, or tag requests.
- prevent requests from being tagged with certain built-in signals.

The configurations can apply to multiple sites or individual sites:

- **Corp-level rules:** rules that apply to all sites or multiple, specific sites in your corp. You can create and manage corp rules via the Corp Rules page.
- **Site-level rules:** rules that apply to one specific site. You can create and manage site rules via the Site Rules and Templated Rules pages.

How rules work

Rules define how the Signal Sciences application should handle requests to the web applications you're protecting. The Signal Sciences agent uses your active rules to determine what should happen to individual requests (e.g., allow, block, rate limit, or tag). The agent then performs any tagging decisions and sends the decisions to allow, block, or rate limit requests to the appropriate entity for you particular deployment method. The entity enacts the agent's decisions.

Rules precedence

When rules conflict, the Signal Sciences agent uses the following logic to determine which rule should take precedence:

- a rule with an allow action always takes precedence over a rule with a block action. For example, if you create a rule to block a range of IP addresses and a rule to allow one specific IP address within that range, requests from that IP address will be allowed because the allow rule takes precedence.
- a corp rule usually takes precedence over a site rule. The only time a corp rule doesn't take precedence over a site rule is when the site rule has an allow action.

Types of rules

There are four types of rules:

- **Request rules:** allow, block, or tag certain requests on an individual basis. For example, you could make a rule to block all requests with specific headers, requests to certain paths, or requests originating from specific IP addresses.
- **Advanced rate limiting rules:** block or tag requests from individual clients when a threshold (e.g., 100 requests in 1 minute) is passed. For example, you could make a rule to rate limit requests made to your site's login page to prevent account takeover attacks. If too many failed login attempts are made from a specific IP address, it's reasonable to suspect that person is trying to guess a password and break into another person's account. The rate limit rule will block that IP address from the login path for a set amount of time and prevent them from continuing to guess passwords.
- **Signal exclusion rules:** prevent requests from being tagged with certain signals. Signal exclusion rules help prevent false positives. For example, let's say you have an internal CMS where employees can post raw HTML. If employees try to post raw HTML that look like a Cross-Site Scripting (XSS) attack, their requests might get tagged with the `XSS` system signal and then blocked. To prevent false positives and your well-meaning employees from being accidentally blocked, you could create a signal exclusion rule to prevent requests that are coming from your VPN IP and post HTML from being tagged with the `XSS` signal.
- **Templated rules:** partially pre-constructed rules that can help you protect against Common Vulnerabilities and Exposures (CVE) and gain visibility into registrations, logins, and API requests. For example, you can enable the `GraphQL API Query` templated rule to track GraphQL API requests.

Limitations and considerations

When working with the Signal Sciences agent, keep the following things in mind:

- Per our agent end-of-support policy, we support agent versions that are under two years old, and on a quarterly cadence, we deprecate and no longer support agent versions that are older than two years.
- Check the Agent Release Notes to see what's new in the agent.

Working with the agent auto-update service

The agent auto-update service checks the Signal Sciences package downloads site for a new version of the agent and updates the agent when a new version is available.

> **Important:** This information is part of a beta release. For additional details, read our product and feature lifecycle descriptions.

Limitations and considerations

When setting up the agent auto-update service, keep the following in mind:

- The agent auto-update service is only compatible with agents on Debian 8 or higher, Red Hat CentOS 7 or higher, and Ubuntu 18.04 or higher.
- The agent auto-update service updates an agent by uninstalling the old package version and installing the latest version. Due to the agent's brief downtime during upgrade, we recommend scheduling the update when your website or web application receives low traffic.

Enable the agent auto-update service

Once the agent is installed, you can enable the agent auto-update service:

1. Enable the agent auto update service.

   ```
   sudo systemctl enable --now sigsci-agent-update.timer
   ```

2. Optionally, customize the agent auto update timer. By default, the check for new versions is performed on the second Thursday of the month.

   ```
    sudo systemctl edit sigsci-agent-update.timer

    [Timer]
    OnCalendar=
    OnCalendar=Thu *-*-08,09,10,11,12,13,14 03:00:00
    RandomizedDelaySec=8h
   ```

Disable the agent auto-update service

To disable the agent auto-update service, run the following command:

```
sudo systemctl disable --now sigsci-agent-update.timer
```

Upgrading the Agent on Ubuntu-Debian systems

To manually upgrade agents on Ubuntu or Debian systems, follow these steps:

1. **Upgrade the Agent package**

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get install sigsci-agent
   ```

2. **Restart the Agent** After successfully upgrading the package, restart your agent:

   Ubuntu 14.04 and lower:

   ```
   sudo restart sigsci-agent
   ```

   Ubuntu 15.04 or higher:

   ```
   sudo systemctl start sigsci-agent
   ```

1. **Upgrade the Agent Package**

   ```
   yum -q makecache -y --disablerepo=* --enablerepo=sigsci_*
   ```

   ```
   yum install sigsci-agent
   ```

2. **Restart the Agent**

   RHEL 6/CENTOS 6

   Under EL6, the Agent is managed via upstart. Restart the agent by running:

   ```
   sudo restart sigsci-agent
   ```

   RHEL 7/CENTOS 7

   From EL7, Red Hat have migrated to SystemD as their default process supervisor. Restart the agent by running:

   ```
   sudo systemctl restart sigsci-agent
   ```

Upgrading the Agent on Windows systems

To manually upgrade agents on Windows systems, follow these steps:

1. **Upgrade the Agent Package**

   Download and install the latest agent MSI.

   Download: Windows MSI

2. **Restart the Agent Service**

   From the UI

   1. Open services.msc
   2. Select "Signal Sciences Agent"
   3. Right click and select restart

   From the CLI

   1. Open up a dos prompt
   2. run `net stop sigsci-agent`
   3. run `net start sigsci-agent`

Using Signal Sciences

- Web interface
- Account info
- Agent alerts
- Agent mode
- Agent response codes
- Data storage and privacy
- Header Links
- Rules
- Signals
- Site Alerts
- Sites

About agent response codes

Agent response codes indicate the Signal Sciences agent's decision to allow or block requests to your web application. Specifically, the 200 agent response code indicates the request should be allowed, and agent response codes greater than or equal to 301 indicate the request should be blocked.

You can view the agent response code for an individual request by navigating to the request details page for that request.

determine what should happen to the request (e.g., allow or block). Based on the decision, the agent assigns the request an agent response code. The agent sends this code along with the request details to the appropriate entity for your deployment method.

The entity then continues processing the request and sends the requesting client the appropriate HTTP status code. Due to internal business logic, the entity may return a HTTP status code that differs from the agent response code. For example, a request may have a 200 agent response code but a 302 HTTP status code if the entity contains additional logical.

Types of agent response codes

There are two types of agent response codes:

- **Custom agent response codes (blocking response codes):** codes greater than or equal to 301. These codes indicate a request should be blocked. By default, blocked requests receive a 406 agent response code. However, you can change this default behavior.
- **System agent response codes:** codes that indicate the request should be allowed or that the request wasn't processed correctly.

Notable agent response codes

Notable agent response codes include:

Agent response code	Description
-2	Indicates the request wasn't processed correctly. For information on how to troubleshoot this response code, visit our Troubleshooting agent response codes guide.
-1	Indicates the request wasn't processed correctly. For information on how to troubleshoot this response code, visit Troubleshooting agent response codes.
0	Indicates the request wasn't processed correctly. For information on how to troubleshoot this response code, visit our Troubleshooting agent response codes guide.
200	Indicates the request should be allowed. This is similar to an `HTTP 200 OK` response.
301	Indicates the request should be redirected. Visit Using redirect custom response codes to learn more.
302	Indicates the request should be redirected. Visit Using redirect custom response codes to learn more.
406	Indicates the request should be blocked (similar to an `HTTP 406 NOT ACCEPTABLE` response). By default, all blocked requests return a 406. You can update the site default blocking response code from 406 to an alternative custom response code and create rules with a block action to return a specific custom response codes.
499	Indicates the client closed the connection mid-request. For information on how to troubleshoot this timeout error, visit our Troubleshooting agent response codes guide.
504	Indicates the gateway did not receive a response from the user's upstream origin in the allotted time specified. For information on how to troubleshoot this timeout error, visit our Troubleshooting agent response codes guide.

About signals

Signals are labels that describe requests. Requests are tagged with signals based on the logic of your active rules. Per our data storage policy, the type of signals that requests are tagged with help determine which individual request data is stored and available in the web interface. You can find and search for requests that have been tagged with a specific signal on the Requests page.

Limitations and considerations

When working with signals, keep the following things in mind:

- The Essentials platform does not support custom signals.
- Depending on the platform you have purchased, you can monitor signals for a site via the Signals Dashboard page or the Signals page.

Platform	Supported page
Essentials	Signals page
Professional	Signals Dashboard page
Premier	Signals Dashboard page

How signals work

When requests are made to your web application, the Signal Sciences agent uses your active rules to identify which requests need to be tagged with a signal and then tags them with the appropriate signal. The system then counts the number of requests that get tagged with a particular signal during one minute periods and makes this data available via time series graphs on the Signals Dashboard and Signals pages.

data for requests that haven't been tagged with a signal.

Types of signals

There are two main types of signals:

- **Custom:** signals that you create at the corp-level and site-level to track request behavior that is particular to your web applications.
- **System:** signals that we create to track common attacks, anomalies, and behaviors (e.g., requests to your API and account login and registration activity).

Filtering requests by signal

On the Requests page, you can use the `tag` field to filter requests by a specific signal.

Signal type	Description
System signal	The search syntax is `tag: <system-signal>`. You will need to replace `<system-signal>` with the name of the system signal that you want to search for.
Corp-level custom signal	The search syntax is `tag: corp.<corp-custom-signal>`. You will need to replace `<corp-custom-signal>` with the name of the corp custom signal that you want to search for. The Corp Signals page lists the custom signals that were created at the corp level.
Site-level custom signal	The search syntax is `tag: site.<site-custom-signal>`. You will need to replace `<site-custom-signal>` with the name of the site custom signal that you want to search for. The Site Signals page lists your custom signals that were created at the site level.

About sites

A site (also known as a workspace) is a single web application, bundle of web applications, API, or microservice that the Next-Gen WAF can protect from attacks. Sites contain various configurations that determine how monitoring agents process incoming requests. These configurations enable request logging and blocking and define what type of requests should be allowed, logged, or blocked.

Every site belongs to a corp (also known as a corporation). A corp is a company hub for managing all sites, users, and corp-level configurations. Users are authenticated against a corp and can be members of different sites in that corp.

When defining the scope of your site, consider how you want to compartmentalize data, rules, and user access. For example, you may want to create sites based on an environment type (e.g., development, staging, and production) or region (e.g., APAC, EU, and US). Read our Managing sites guide for more information.

Monitoring your site

You can monitor your site's traffic and performance via the web interface. For example:

- the Site Overview page contains high-level site metrics organized into multiple dashboards.
- the Requests page lists a sample of individual requests that have been tagged with signals.
- the Observed Source page tracks IP addresses that have been or will be flagged soon, and the Events page contains a historical record of all flagged IP addresses within the last 30 days.
- the agent details page provides a summary of the status and performance of the agent.

About the agent mode

Agent mode is a site-level setting that determines how the Signal Sciences agent handles request processing. Options include:

- **Blocking:** enables request blocking and logging. This option actively protects your web application and provides visibility into your web traffic. Legitimate traffic is still allowed.
- **Not Blocking:** enables request logging. This option provides visibility into your web traffic but doesn't actively protect your site.
- **Off:** disables request processing. The agent doesn't block or log requests. This option doesn't uninstall the agent.

About the Blocking option

When the Agent mode menu is set to `Blocking`, the Next-Gen WAF:

- logs requests based on our storage policy.
- blocks malicious requests from reaching your web servers and doing harm. Site alerts and rules define the criteria used to evaluate and block individual requests.

About the Not Blocking option

When the Agent mode menu is set to `Not Blocking`, the Next-Gen WAF logs requests based on our storage policy and all traffic is allowed.

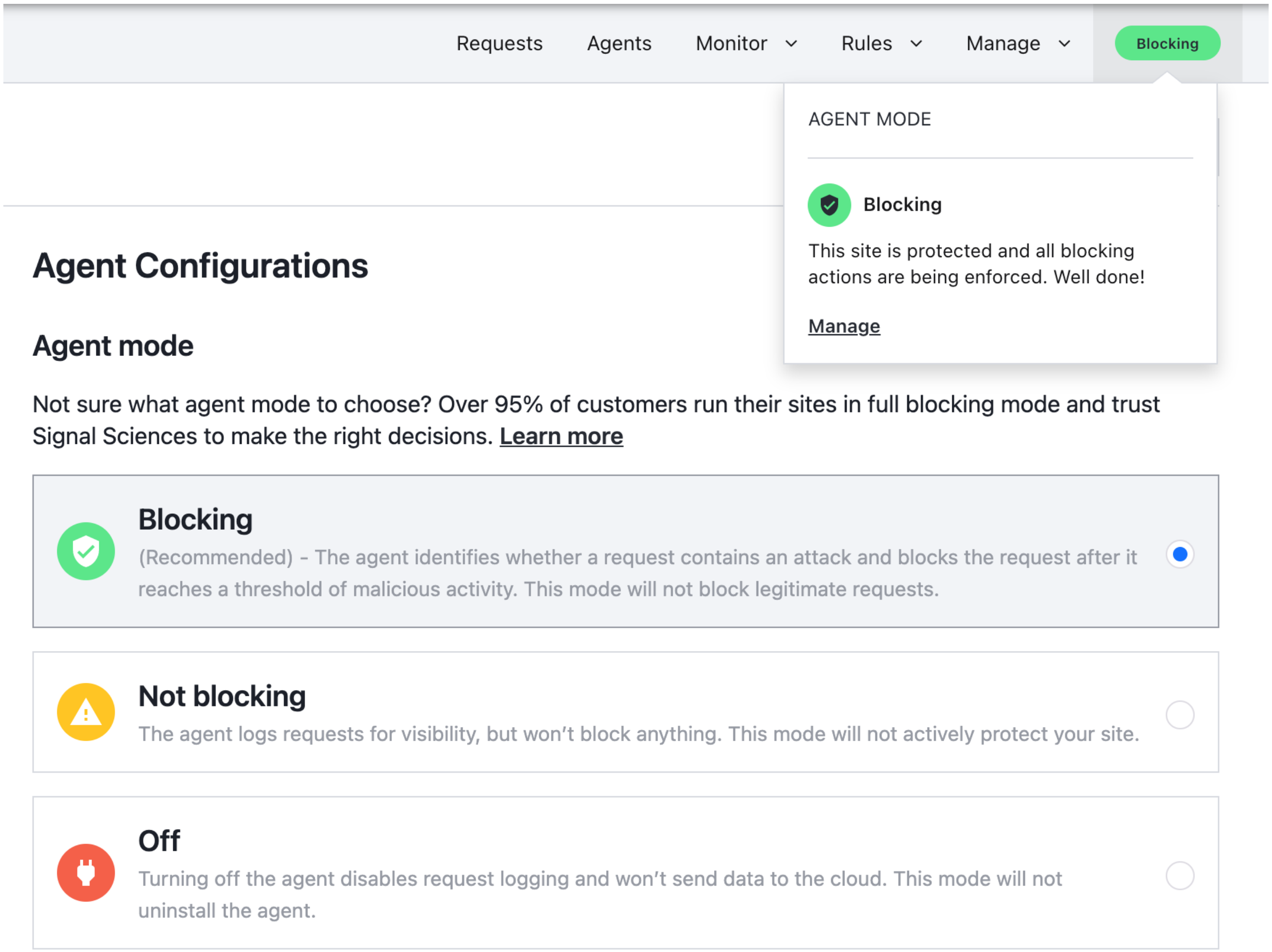
> **IMPORTANT:** The `Not blocking` agent mode never blocks requests. Requests that match rules with a block action will be allowed.

Changing the agent mode

You can change the agent mode for a site via our API and the web interface. To use the web interface to change the agent mode, compete the following steps:

> **NOTE:** Users with an Observer role cannot change the agent mode.

1. Log in to the Signal Sciences console.
2. From the corp navigation bar, use the **Sites** menu to select a site.
3. From the site navigation bar, click the agent mode indicator.
4. Click the **Manage** link. The Agent Configurations form appears.



Blocking

This site is protected and all blocking actions are being enforced. Well done!

Manage

Agent Configurations

Agent mode

Not sure what agent mode to choose? Over 95% of customers run their sites in full blocking mode and trust Signal Sciences to make the right decisions. Learn more

Blocking
(Recommended) - The agent identifies whether a request contains an attack and blocks the request after it reaches a threshold of malicious activity. This mode will not block legitimate requests. ●

Not blocking
The agent logs requests for visibility, but won't block anything. This mode will not actively protect your site. ○

Off
Turning off the agent disables request logging and won't send data to the cloud. This mode will not uninstall the agent. ○

5. From the **Agent mode** menu, select the agent mode for the site. Options include:
    - **Blocking:** enables request blocking and logging. This option actively protects your web application and provides visibility into your web traffic. Legitimate traffic is still allowed.
    - **Not Blocking:** enables request logging. This option provides visibility into your web traffic but doesn't actively protect your site.
    - **Off:** disables request processing. The agent doesn't block or log requests. This option doesn't uninstall the agent.
6. Click the **Update** button.

application programming interface (API).

You must have a Signal Sciences account to be able to access the web interface controls. Contact sales@fastly.com to create an account. Once your account is set up, you can navigate to the controls via the Signal Sciences platform login page at https://dashboard.signalsciences.net/.

About the corp navigation bar

The corp navigation bar provides access to corp features and functions. It contains the following controls:

- a switcher menu that provides direct access to both the Fastly and Signal Sciences applications from a single location. The switcher menu appears as nine squares in a three by three grid.
- the name of your corp which links to the Corp Overview page where you can view relevant data about your corp and its sites
- the Sites menu where you can select the site that you want to work with.
- the Corp Rules menu where you can access corp rules, lists, and signals
- the Corp Manage menu where you can access sites, users, integrations, audit logs, and configurations
- the Help menu where you can access documentation and support tickets
- the My Profile menu where you can access account settings and API access tokens and sign out of the web interface

About the site navigation bar

The site navigation bar provides access to site features and functions. It is only accessible when you are working with a site. To access the site navigation bar, select a site from the Sites menu on the corp navigation bar.

The site navigation bar contains the following controls:

- the name of your site which links to the Site Overview page where you can view metrics for your site
- the Requests page where you can search for requests that were made in the last 30 days
- the Agents page where you can view a list of your agents
- the Signals page where you can monitor site signals

  NOTE: The Signals page is only included with the Essential platform.
- the Monitor menu where you can access events, observed sources, and signals
- the Rules menu where you can access site rules, lists, alerts, and redactions
- the Manage menu where you can access site-specific settings, integrations, configurations, and audit logs
- the agent mode indicator which specifies how the monitoring agent handles requests that are tagged with attack signals from a flagged IP address. When enabled, the agent can either log and block the requests or log but not block the requests. When disabled, the agent does not log or block requests.

Account info

These articles describe how to manage account access and security:

- Automating user management (IdP)
- Enabling and disabling two-factor-authentication
- Linking Fastly accounts
- Managing users
- Monitoring account activity with audit logs
- Setting up single sign-on (SSO)
- Using user roles and permissions

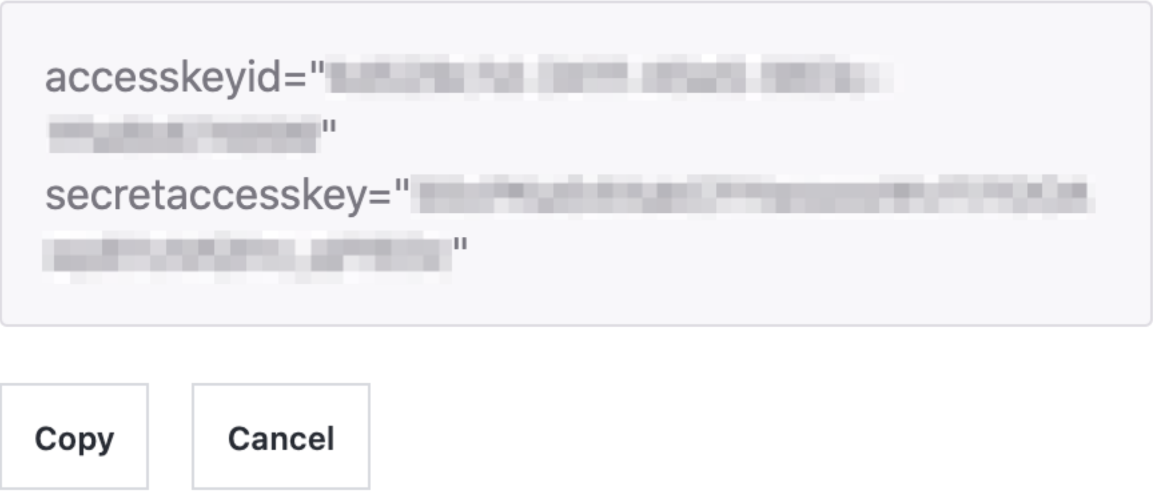
AWS Elastic Container Service (ECS) Setup

Introduction

This guide shows how to create a deployment in AWS ECS to add Signal Sciences in a sidecar configuration. This deployment setup is compatible with both Fargate and EC2 launch types.

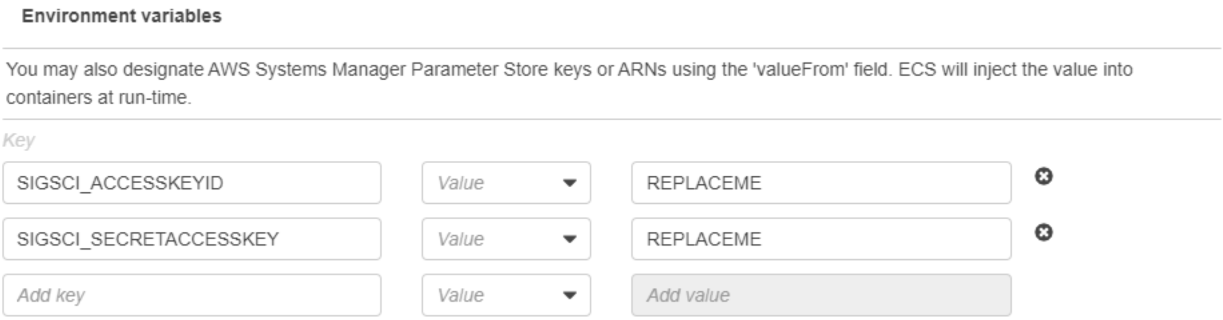
1. Create a new task definition.
2. Select either **Fargate** or **EC2**.
3. Under **Volumes**, click **Add volume** to add the Shared Volume for the containers to use for the Unix Socket file. The Add volume window appears.
4. In the **Add Volume** window:
    1. In the **Name** field, enter a name for the volume.
    2. Select the type of **Bind mount**.
    3. Click **Add**.
5. On the main Task page, click **Add Container**. The Add container window appears.
6. In the **Add Container** window:
    1. In the **Container name** field, enter a Display Name for the container.
    2. In the **Image** field, enter a name for the Docker image. For example, `username/example-app:latest`.
    3. Under **Port mappings**, add any ports that should be available for your app.
7. Create the container.
8. Click **Add Container** to add a second container for the Signal Sciences Agent. The Add container window appears.
9. In the **Add Container** window:
    1. In the **Container name** field, enter `sigsci-agent`.
    2. In the **Image** field, enter `signalsciences/sigsci-agent:latest`.
    3. Under **Port mappings**, add any ports that should be available for your app.
10. In the **Resource Limits** section, modify the base `ulimits`:
    1. Under `ulimits`, add `nofile` to the `limit name`.
    2. Set the soft limit to `65335`.
    3. Set the hard limit to `65335`.
11. Locate the **Agent Keys** for your Signal Sciences site:
    1. Log in to the Signal Sciences console.
    2. From the **Sites** menu, select a site if you have more than one site.
    3. Click **Agents** in the navigation bar. The agents page appears.

    4. Click **View agent keys**. The agent keys window appears.
    5. Copy the **Agent Access Key** and **Agent Secret Key**.



accesskeyid="[redacted]"
secretaccesskey="[redacted]"

12. In the **Environment** section in AWS, enter the **Agent Access Key** and **Agent Secret Key** for your site as **Environment variables** named `SIGSCI_ACCESSKEYID` and `SIGSCI_SECRETACCESSKEY`.



You may also designate AWS Systems Manager Parameter Store keys or ARNs using the 'valueFrom' field. ECS will inject the value into containers at run-time.

Key		
SIGSCI_ACCESSKEYID	Value	REPLACEME
SIGSCI_SECRETACCESSKEY	Value	REPLACEME
Add key	Value	Add value

13. Under **Mount Points** in the **Storage and Logging** section of the Container Definition, select the same mount point as the previous Container.

14. Create the container.

15. Finish creating the task definition.

16. From the **Actions** menu, select **Run Task** or **Create Service** and run on one of your configured clusters.

Example JSON configuration

Note: You will need to replace all instances of `REPLACEME` in this example JSON.

```
{
    "ipcMode": null,
    "executionRoleArn": "arn:aws:iam::REPLACEME:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "dnsSearchDomains": null,
            "logConfiguration": {
                "logDriver": "awslogs",
                "secretOptions": null,
                "options": {
                    "awslogs-group": "/ecs/sigsci-example",
                    "awslogs-region": "us-west-1",
                    "awslogs-stream-prefix": "ecs"
                }
            },
                    "hostPort": 8080,
                    "protocol": "tcp",
                    "containerPort": 8080
                }
            ],
            "command": null,
            "linuxParameters": null,
            "cpu": 0,
            "environment": [
                {
                    "name": "apache_port",
                    "value": "8080"
                },
                {
                    "name": "sigsci_rpc",
                    "value": "/var/run/sigsci.sock"
                }
            ],
            "dnsServers": null,
            "mountPoints": [
                {
                    "readOnly": null,
                    "containerPath": "/var/run",
                    "sourceVolume": "run"
                }
            ],
            "workingDirectory": null,
            "secrets": null,
            "dockerSecurityOptions": null,
            "memory": null,
            "memoryReservation": null,
            "volumesFrom": [],
            "stopTimeout": null,
            "image": "signalsciences/sigsci-agent:latest",
            "startTimeout": null,
            "firelensConfiguration": null,
            "dependsOn": null,
            "disableNetworking": null,
            "interactive": null,
            "healthCheck": null,
            "essential": true,
            "links": null,
            "hostname": null,
            "extraHosts": null,
            "pseudoTerminal": null,
            "user": null,
            "readonlyRootFilesystem": null,
            "dockerLabels": null,
            "systemControls": null,
            "privileged": null,
            "name": "apache"
        },
        {
            "dnsSearchDomains": null,
            "logConfiguration": {
                "logDriver": "awslogs",
                "secretOptions": null,
                "options": {
                }
            },
            "entryPoint": null,
            "portMappings": [],
            "command": null,
            "linuxParameters": null,
            "cpu": 0,
            "environment": [
                {
                    "name": "SIGSCI_ACCESSKEYID",
                    "value": "REPLACEME"
                },
                {
                    "name": "SIGSCI_SECRETACCESSKEY",
                    "value": "REPLACEME"
                }
            ],
            "ulimits": [
                {
                    "name": "nofile",
                    "softLimit": 65335,
                    "hardLimit": 65335
                }
            ],
            "dnsServers": null,
            "mountPoints": [
                {
                    "readOnly": null,
                    "containerPath": "/var/run",
                    "sourceVolume": "run"
                }
            ],
            "workingDirectory": null,
            "secrets": null,
            "dockerSecurityOptions": null,
            "memory": null,
            "memoryReservation": null,
            "volumesFrom": [],
            "stopTimeout": null,
            "image": "signalsciences/sigsci-agent:latest",
            "startTimeout": null,
            "firelensConfiguration": null,
            "dependsOn": null,
            "disableNetworking": null,
            "interactive": null,
            "healthCheck": null,
            "essential": true,
            "links": null,
            "hostname": null,
            "extraHosts": null,
            "pseudoTerminal": null,
            "user": null,
            "readonlyRootFilesystem": null,
            "dockerLabels": null,
            "systemControls": null,
            "privileged": null,
            "name": "agent"
        }
    "family": "sigsci-example",
    "pidMode": null,
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "host",
    "cpu": "2048",
    "inferenceAccelerators": null,
    "proxyConfiguration": null,
    "volumes": [
        {
            "efsVolumeConfiguration": null,
            "name": "run",
            "host": {
                "sourcePath": null
            },
            "dockerVolumeConfiguration": null
        }
    ],
    "tags": []
}
```

Debian NGINX 1.10-1.14

Add the package repositories

Add the version of the Debian package repository that you want to use.

Debian 10 - Buster

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ buster ma
sudo apt-get update
```

Debian 9 - Stretch

```
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ stretch main
EOF
sudo apt-get update
```

Debian 8 - Jessie

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ jessie main
EOF
sudo apt-get update
```

Debian 7 - Wheezy

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ wheezy main
EOF
sudo apt-get update
```

support. You must first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

Install the Lua NGINX Module

Install the dynamic Lua NGINX Module appropriate for your NGINX distribution.

NGINX.org distribution

1. Install the Lua NGINX Module.

   - NGINX 1.12.1 or higher

     ```
     sudo apt-get install nginx-module-lua
     ```

   - NGINX 1.11

     ```
     sudo apt-get install nginx111-lua-module
     ```

   - NGINX 1.10

     ```
     sudo apt-get install nginx110-lua-module
     ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the line that starts with `pid`:

   ```
   load_module modules/ndk_http_module.so;
   load_module modules/ngx_http_lua_module.so;
   ```

3. Restart the NGINX service to initialize the new module.

   ```
   sudo service nginx restart
   ```

Debian distribution

Enable Lua by installing the `nginx-extras` package.

```
sudo apt-get install nginx-extras && sudo service nginx restart
```

Check that Lua is loaded correctly

Load the following config (e.g., `sigsci_check_lua.conf`) with NGINX to verify that Lua has been loaded properly:

```
# Config just to test for lua jit support
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
# Also, for some nginx-1.10.nn installed from nginx-extras package, you may
# need to specify the load directives.
# Given the above uncomment the following:
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768
    # multi_accept on;
}
http {
init_by_lua '

local m = {}
```

```
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
if not r then
  error("ERROR: No lua jit support: No support for SigSci Lua module")
else

  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
  end

end

';

}
```

Example of a successfully loaded config and its output

```
$ nginx -t -c <your explicit path>/sigsci_check_lua.conf

nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
nginx: the configuration file <your explicit path>/sigsci_check_lua.conf syntax is ok
nginx: configuration file <your explicit path>/sigsci_check_lua.conf test is successful
```

Install the NGINX module

1. Install the module.

   ```
   apt-get install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX Service to initialize the new module

   - Debian 8 or higher

     ```
     sudo systemctl unmask nginx && sudo systemctl restart nginx
     ```

   - Debian 7

     ```
     sudo service nginx restart
     ```

Agent StatsD Metrics

StatsD Metrics

Metrics can be reported through StatsD to the service of your choice using the `statsd-address` agent configuration flag.

Metrics can be filtered using the `statsd-metrics` agent configuration flag.

The following metrics are reported through StatsD:

Metric	Type	Description
`sigsci.agent.waf.total`	counter	The number of requests inspected
`sigsci.agent.waf.error`	counter	The number of errors while attempting to process a request
`sigsci.agent.waf.allow`	counter	The number of allow decisions
`sigsci.agent.waf.block`	counter	The number of block decisions
`sigsci.agent.waf.perf.decision_time.50pct`	gauge	The 50th percentile of the decision time (in milliseconds)
`sigsci.agent.waf.perf.decision_time.95pct`	gauge	The 95th percentile of the decision time (in milliseconds)
`sigsci.agent.waf.perf.decision_time.99pct`	gauge	The 99th percentile of the decision time (in milliseconds)
`sigsci.agent.waf.perf.queue_time.50pct`	gauge	The 50th percentile of the queue time (in milliseconds)
`sigsci.agent.waf.perf.queue_time.95pct`	gauge	The 95th percentile of the queue time (in milliseconds)
`sigsci.agent.waf.perf.queue_time.99pct`	gauge	The 99th percentile of the queue time (in milliseconds)
`sigsci.agent.rpc.connections.open`	gauge	The number of open RPC connections
`sigsci.agent.runtime.cpu_pct`	gauge	CPU percent used by the agent
`sigsci.agent.runtime.mem.sys_bytes`	gauge	Memory used by the agent
`sigsci.agent.runtime.uptime`	gauge	Agent uptime
`sigsci.agent.signal.NAME`	counter	Number of NAME signals

Golang Module Install

Download and install prerequisites

The Golang module requires two prerequisite packages to be installed: MessagePack Code Generator and the Signal Sciences custom tlstext package.

Install these packages using the `go get` command to download and install these packages directly from their GitHub repositories:

```
go get -u -t github.com/tinylib/msgp/msgp
go get -u -t github.com/signalsciences/tlstext
```

Download and extract the Golang module

1. Download the latest version of the Golang module:

   ```
   curl -O -L https://dl.signalsciences.net/sigsci-module-golang/sigsci-module-golang_latest.tar.gz
   ```

2. Extract the Golang module to `$GOPATH/src/github.com/signalsciences`:

   ```
   sudo mkdir -p $GOPATH/src/github.com/signalsciences
   sudo tar -xf sigsci-module-golang_latest.tar.gz -C $GOPATH/src/github.com/signalsciences
   ```

Wrap your application

You will need to wrap your application in the Signal Sciences Golang module handler for the module to process requests and secure your application.

> **Note:** How to best wrap your application will depend on how your application is designed. The steps listed below are provided as an example, but the methods listed may not be ideal for your specific application. More information about the Golang `http` package, including alternative methods, can be found here.

1. In the `import` section of your Golang application, add the following line to import the Golang module:

   ```
   sigsci "github.com/signalsciences/sigsci-module-golang"
   ```

2. Create a new ServeMux in your `main()` function to be used with the module:

   ```
   muxname := http.NewServeMux()
   ```

3. Add functions to the ServeMux by adding `mux.handleFunc` lines. For example, functions named `hellofunc` and `examplefunc` can be added with lines such as these:

4. Wrap your ServeMux in the Signal Sciences Golang module by adding lines similar to this example:

```
wrappername, err := sigsci.NewModule(muxname)
if err != nil {
    log.Fatal(err)
}
```

5. Call the wrapper in the method your application uses to serve HTTP requests. For example, if you're using the `ListenAndServe` method, then you would use call the wrapper with:

```
http.ListenAndServe("127.0.0.1:80", wrappername)
```

Example Application

Below is an example hello world application with the Signal Sciences Golang module successfully integrated:

```
package main

import (
     "fmt"
     "log"
     "net/http"

     sigsci "github.com/signalsciences/sigsci-module-golang"
)

func hellofunc(w http.ResponseWriter, r *http.Request) {
     fmt.Fprintf(w, "Hello, world")
}

func examplefunc(w http.ResponseWriter, r *http.Request) {
     fmt.Fprintf(w, "Example function output")
}

func main() {
     muxname := http.NewServeMux()
     muxname.HandleFunc("/hello", hellofunc)
     muxname.HandleFunc("/example", examplefunc)

     wrappername, err := sigsci.NewModule(muxname)
     if err != nil {
          log.Fatal(err)
     }

     http.ListenAndServe("127.0.0.1:80", wrappername)
}
```

Example Helloworld Test Web Application

Hello world Test Web Application

This uses the `helloworld` example included with the Signal Sciences Golang module as a test web application named `helloworld`.

See: `main.go` in the `sigsci-module-golang` `helloworld` example

Dockerfile

Dockerfile to build the `signalsciences/example-helloworld` container:

```
docker build . -t signalsciences/example-helloworld:latest
```

```
FROM golang:1.13
```

```
# Install sigsci golang module (with examples)
RUN go get github.com/signalsciences/sigsci-module-golang

# Use the helloworld example as the test app
WORKDIR /go/src/github.com/signalsciences/sigsci-module-golang/examples

ENTRYPOINT [ "go", "run", "./helloworld" ]
```

Kubernetes Deployment File

Kubernetes `example-helloworld` deployment file (without the Signal Sciences Agent):

```
kubectl apply -f example-helloworld.yaml
```

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld
  labels:
    app: helloworld
spec:
  ports:
  - name: http
    port: 8000
    targetPort: 8000
  selector:
    app: helloworld
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld
  labels:
    app: helloworld
spec:
  replicas: 2
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
        # Address for the app to listen on
        - localhost:8000
        ports:
        - containerPort: 8000
```

Debian NGINX 1.9 or lower

Add the package repositories

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ buster ma
sudo apt-get update
```

Debian 9 - Stretch

```
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ stretch main
EOF
sudo apt-get update
```

Debian 8 - Jessie

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ jessie main
EOF
sudo apt-get update
```

Debian 7 - Wheezy

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ wheezy main
EOF
sudo apt-get update
```

Enable Lua for NGINX

Some older versions of NGINX don't support native loading of Lua modules. Therefore, we require NGINX to be built with the third party `ngx_lua` module. Because most older versions of NGINX do not support dynamically loadable modules, you will likely need to rebuild NGINX from source.

To assist you, we provide pre-built drop-in replacement NGINX packages already built with the `ngx_lua` module. This is intended for users who prefer not to build from source, or who either use a distribution-provided package or an official NGINX provided package. These pre-built packages are built to support much older distributions and are not gpg signed.

Flavors

We support three flavors of NGINX. These flavors are based on what upstream package we've based our builds on. All our package flavors are built according to the official upstream maintainer's build configuration with the addition of the `ngx_lua` and `ngx_devel_kit` modules.

Our provided flavors are:

- **Distribution** - The distribution flavor is based off the official distribution-provided NGINX packages. For Debian-based Linux distributions (Red Hat and Debian) these are the based off the official Debian NGINX packages.
- **Stable** - The stable flavor is based off the official NGINX.org stable package releases.
- **Mainline** - The mainline flavor is based off the official NGINX.org mainline package releases.

Flavor version support

The following versions are contained in the various OS and flavor packages:

OS	Distribution	Stable	Mainline
Debian 8 (Jessie)	1.6.2	1.8.1	1.9.10
Debian 7 (Wheezy)	1.2.1	1.8.1	1.9.10

The versions are dependent on the upstream package maintainer's supported version.

```
wget -qO - https://apt.signalsciences.net/nginx/gpg.key | sudo apt-key add -
```

2. Create a new file `/etc/apt/sources.list.d/sigsci-nginx.list` with the following content based on your OS distribution and preferred flavor:

    - Distribution flavor

OS	`sigsci-nginx.list` content
Debian 8 (Jessie)	`deb https://apt.signalsciences.net/nginx/distro jessie main`
Debian 7 (Wheezy)	`deb https://apt.signalsciences.net/nginx/distro wheezy main`

    - Stable flavor

OS	`sigsci-nginx.list` content
Debian 8 (Jessie)	`deb https://apt.signalsciences.net/nginx/stable jessie main`
Debian 7 (Wheezy)	`deb https://apt.signalsciences.net/nginx/stable wheezy main`

    - Mainline flavor

OS	`sigsci-nginx.list` content
Debian 8 (Jessie)	`deb https://apt.signalsciences.net/nginx/mainline jessie main`
Debian 7 (Wheezy)	`deb https://apt.signalsciences.net/nginx/mainline wheezy main`

3. Update the `apt` caches.

    ```
    apt-get update
    ```

4. Uninstall the default NGINX.

    ```
    sudo apt-get remove nginx nginx-common nginx-full
    ```

5. Install the version of NGINX provided by Signal Sciences.

    ```
    sudo apt-get install nginx
    ```

Check Lua is loaded correctly

To verify Lua has been loaded properly load the following config (`sigsci_check_lua.conf`) with NGINX:

```
# Config just to test for lua jit support
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
# Also, for some nginx-1.10.nn installed from nginx-extras package, you may
# need to specify the load directives.
# Given the above uncomment the following:
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768
    # multi_accept on;
}
http {
init_by_lua '
```

```
if ngx then
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
if not r then
  error("ERROR: No lua jit support: No support for SigSci Lua module")
else

  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
  end

end

';

}
```

If the config is successfully loaded, the above script will create the following output:

```
$ nginx -t -c <your explicit path>/sigsci_check_lua.conf

nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
nginx: the configuration file <your explicit path>/sigsci_check_lua.conf syntax is ok
nginx: configuration file <your explicit path>/sigsci_check_lua.conf test is successful
```

Install the NGINX module

1. Install the module.

   ```
   apt-get install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX Service to initialize the new module.

   - Debian 8 or higher

     ```
     sudo systemctl unmask nginx && sudo systemctl restart nginx
     ```

   - Debian 7

     ```
     sudo service nginx restart
     ```

Reverse Proxy Mode

The Agent can be configured to run as a reverse proxy allowing it to interact directly with requests and responses without the need for a module. Running the Agent in reverse proxy mode is ideal when a module for your web service does not yet exist or you do not want to modify your web service configuration - for example, while testing the product. In this mode, the agent sits inline as a service in front of your web service.

the Agent will not function with any modules.

Reverse Proxy Listener Configuration

The reverse proxy now supports an arbitrary number of listeners (before only 1 each of HTTP and TLS). Each listener is now configured in a `revproxy-listener` block. Each block is defined by a unique name in the format `[revproxy-listener.NAME]`. Each block has its own set of directives for that listener. Multiple blocks are supported, but all blocks *must* be at the end of the configuration file after all other global options.

For example, to configure a simple HTTP (no encryption) listener, update the agent.conf file (default: `/etc/sigsci/agent.conf`) to include the following configuration block as shown below that creates an HTTP reverse proxy listener named `example1`:

```
[revproxy-listener.example1]
listener = "http://203.0.113.13:80"
upstreams = "http://192.168.1.2:80"
```

The `listener` option is the address the Agent will listen on in the form of a URL, and the `upstreams` option defines the upstream hosts that the Agent will proxy requests to. The upstream hosts are a comma-separated list of URLs. The scheme of the URLs specify the protocol that will be used for listening and proxying to the upstreams.

> **Note:** When you add more than one host in a listener's `upstreams` option, requests are dynamically distributed between upstream hosts via round-robin load balancing. If your load balancer is configured for sticky sessions (session affinity), you will need to create a separate listener for every upstream host.

To configure a TLS encrypted listener, use the `https` scheme for the `listener` option and configure the `tls-key` and `tls-cert` options to point to files containing the key and cert. The `upstreams` scheme determines the protocol used to proxy to the upstream hosts.

Encrypt traffic to Upstream

```
[revproxy-listener.example2]
listener = "https://203.0.113.13:8080"
upstreams = "https://192.168.1.2:8443,https://192.168.1.3:8443"
tls-cert = "/etc/sigsci/server-cert.pem"
tls-key = "/etc/sigsci/server-key.pem"
```

Terminating TLS at the Agent

This option is similar to the above with the only difference being the scheme used in the `upstreams`.

```
[revproxy-listener.example3]
listener = "https://203.0.113.13:8443"
upstreams = "http://192.168.1.2:8001,http://192.168.1.2:8002"
tls-cert = "/etc/sigsci/server-cert.pem"
tls-key = "/etc/sigsci/server-key.pem"
```

> **Note:** In both options, the cert and key files can be the same file provided you concatenate both key and cert into one file.

After you have completed the desired configuration, reload the `sigsci-agent` configuration for the changes to take effect. On most systems this can be done by sending a SIGHUP signal to the agent process ID (e.g., `kill -HUP 12345` where 12345 is the PID) or just restarting the agent.

The `[revproxy-listener.NAME]` configuration and its available options are documented on the agent configuration page.

Alternative configuration without a configuration file

If you are not using a configuration file, then you cannot use the new block format above and you must instead use an alternative format. This format can be used with a single `--revproxy-listener` command line option or via a single `SIGSCI_REVPROXY_LISTENER` environment variable.

Generic format for the alternative revproxy-listener value

```
listener1:{opt=val,...}; listener2:{...}; ...
```

Some example from above are repeated here in the alternative format.

Simple HTTPS listener

```
SIGSCI_REVPROXY_LISTENER="example2:{listener=https://203.0.113.13:443,upstreams=https://192.168.1.2:8443,tls-cert=
```

Multiple listeners can be specified in a single option by separating each listener definition with a semicolon (`;`).

Multiple listeners

```
SIGSCI_REVPROXY_LISTENER="example1:{...}; example2:{...}"
```

Side Effects and Limitations

HTTP header names are normalized

The agent in reverse proxy mode will normalize all header names by capitalizing the first letter in each word. For example, `example-header` becomes `Example-Header`.

HTTP header order may not be maintained

Due to technical limitation, the agent in reverse proxy mode does not allow for tracking and maintaining the order of headers. The order of headers may change when sent to the upstream server. For example:

```
GET /test HTTP/1.1
Host: example.com
X-Example-Header: example
X-Test-Header: test
X-Other-Header: other
Accept: */*
```

This request may arrive at the upstream server as:

```
GET /test HTTP/1.1
Host: example.com
Accept: */*
X-Test-Header: test
X-Other-Header: other
X-Example-Header: example
```

Added headers

By default, the following headers are added to the upstream request:

- `X-Forwarded-For`
- `X-Forwarded-Host`
- `X-Forwarded-Proto`
- `X-Forwarded-Server`

In agent v3.7+, each listener can be configured with `minimal-header-rewriting = true` and these additional headers will not be added/modified. These headers will still be passed through if they exist in the request. Additionally, configuring a listener to not trust the proxy headers with `trust-proxy-headers = false` will strip these headers before sending to the upstream.

Additionally, the following Signal Sciences headers will be added regardless of the above configurations:

- `X-Sigsci-Agentresponse`
- `X-Sigsci-Tags` (only if there were signals added)

HTTP/1.0 to upstream is upgraded to HTTP/1.1

Any HTTP/1.0 requests processed by the agent in reverse proxy mode will be upgraded to HTTP/1.1 when sent to the upstream. This means:

- HTTP keepalives are enabled by default
- HTTP/1.1 version is used in the request line
- The HTTP Host header is added
- The `Accept-Encoding: gzip` header is added

HTTP/0.9 is not supported

For example, `GET /` is a request in HTTP/0.9 format and would result in a `400` error. By contrast, `GET / HTTP/1.0` is in the supported HTTP/1.0 format, which specifies the HTTP version.

Failing open

When the agent is running in reverse proxy mode, requests that have failed open are not sent to the Signal Sciences cloud backend and therefore won't be visible on the requests page of the console.

Next Steps

Verify the agent and module installation and explore module options.

Debian NGINX-Plus

Add the package repositories

Add the version of the Debian package repository that you want to use.

Debian 11 - Bullseye

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ bullseye
sudo apt-get update
```

Debian 10 - Buster

```
sudo apt-get update
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/sigsci.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/sigsci.gpg] https://apt.signalsciences.net/release/debian/ buster ma
sudo apt-get update
```

Debian 9 - Stretch

```
sudo apt-get install -y apt-transport-https wget gnupg
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ stretch main
EOF
sudo apt-get update
```

Debian 8 - Jessie

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ jessie main
EOF
sudo apt-get update
```

Debian 7 - Wheezy

```
sudo apt-get install -y apt-transport-https wget
wget -qO - https://apt.signalsciences.net/release/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/sigsci-release.list <<-'EOF'
deb https://apt.signalsciences.net/release/debian/ wheezy main
EOF
sudo apt-get update
```

Install the NGINX module

1. Install the Signal Sciences NGINX module by running the following command:

   ○ NGINX Plus 29

```
sudo apt-get install nginx-module-sigsci-nxp=1.23.2*
```

- NGINX Plus 27

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.21.6*
  ```

- NGINX Plus 26

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.21.5*
  ```

- NGINX Plus 25

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.21.3*
  ```

- NGINX Plus 24

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.19.10*
  ```

- NGINX Plus 23

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.19.5*
  ```

- NGINX Plus 22

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.19.0*
  ```

- NGINX Plus 21

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.17.9*
  ```

- NGINX Plus 20

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.17.6*
  ```

- NGINX Plus 19

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.17.3*
  ```

- NGINX Plus 18

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.15.10*
  ```

- NGINX Plus 17

  ```
  sudo apt-get install nginx-module-sigsci-nxp=1.15.7*
  ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the `pid /run/nginx.pid;` line:

   ```
   load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
   ```

3. Restart the NGINX service to initialize the new module.

   ```
   sudo service nginx restart
   ```

Glossary

Term	Definition
Admin	A user role that has limited access to corp configurations, can edit specific sites, and can invite users to sites.
Agent	One of the main components of the Signal Sciences architecture. The agent receives requests from modules and quickly decides whether those requests contain attacks or not. The agent then passes their decision back to the module.
Agent alerts	Custom alerts that trigger notifications whenever: - The average number of requests per second (RPS) for all agents across all sites reaches a user-specified threshold - The number of online agents reaches a user-specified threshold.
Agent mode	Determines whether to block requests, not block requests, or entirely disable request processing.
Allow	An agent decision to allow a request through.

requests originating from known scanners.

API access tokens	Permanent tokens used to access the Signal Sciences API. Users can connect to the API using their email and access token.
Attacks	Malicious requests containing attack payloads designed to hack, destroy, disable, steal, gain unauthorized access, and otherwise take harmful actions against a corp's sites.
Audit log	An audit of activity, changes, and updates made to a site or corp.
Blocking	An agent mode that blocks subsequent attacks from a flagged IP address after it has been identified as malicious. Blocking mode still allows legitimate traffic through if the requests do not contain attacks.
Cards	Visual charts of data that can be monitored and customized on site dashboards.
Cloud engine	One of the main components of the Signal Sciences architecture. The cloud engine collects metadata to help improve agent detections and decisions.
Configurations	A set of features that users can customize to meet their business needs. Configurations include: rules, lists, signals, alerts, integrations, site settings, and user management.
Corp (Corporation)	A company hub for monitoring all site activity and managing all sites, users, and corp configurations. Users are authenticated against a corp and can be members of different sites in that corp.
Dashboards	The corp and site homepages. The site dashboard gives visibility into specific types of attacks and anomalies. The corp dashboard gives a snapshot of all top site activity including which sites have the most attack requests, blocked requests, and flagged IP addresses.
Events	Actions that Signal Sciences takes as the result of regular threshold-based blocking, templated rules, site alerts, and rate limit rules. This includes any occurrence that happens on the Events page, such as a flagged IP address. Events are automatically system generated.
Flagged IP addresses	An IP address that has been flagged for exceeding thresholds.
Header links	External data like Splunk or Kibana that connects with request data from Signal Sciences.
Integrations	DevOps toolchain apps that send activity notifications to users. Examples include Slack, Datadog, PagerDuty, mailing lists, and generic webhooks.
IP Anonymization	IP addresses are converted to anonymous IPv6 addresses so that Signal Sciences will not know the actual IP address, which causes the IP address to appear anonymous in the dashboard.
Lists	Sets of custom data used in corp and site rules, such as a list of countries a corp doesn't do business with. Lists include sets of countries, IP addresses, strings, and wildcards.
Log	In not blocking mode, requests that would have been blocked are logged and allowed to pass through instead.
Module	One of the main components of the Signal Sciences architecture. The module receives and passes requests to the agent. It then enforces the agent's decisions to either allow, log, or block those requests.
Monitor	To observe and keep watch over corp and site events.
Monitor view	The site dashboard in a TV-friendly format.
Not blocking	The default agent mode. In this mode, attacks are logged but not blocked and the site is not actively protected.
Notification	Any product message sent internally or externally. External notifications are sent through integrations when activity happens (e.g., a Slack notification is sent when a new site is created).
Observer	A user role that can view sites they are assigned to, but cannot edit any configurations.
Off	An agent mode that stops sending traffic to Signal Sciences and disables all request processing.
Owner	A user role that has access to all corp configurations, can edit every site, and can manage users.
Rate limit rule	A type of rule that allows you to use the Advanced Rate Limiting feature to define arbitrary conditions and automatically begin to block or tag requests that pass a user-defined threshold.
Redactions	Sensitive data that is not sent to the Signal Sciences backend for privacy reasons. Signal Sciences redacts some sensitive data by default, such as credit card numbers and social security numbers. In addition to the default redactions, users can specify their own custom redactions.
Request rule	A type of rule that allows you to define arbitrary conditions to block, allow, or tag requests.
Requests	Information that is sent from the client to the server over the hypertext transfer protocol (HTTP). Signal Sciences protects over a trillion production requests per month.
Response time	The amount of time between when a request was received by the server and when the server generated a response.
Role	Every user is assigned one role: owner, admin, user, or observer.
Rules	A configuration that defines conditions to block, allow, or tag requests or exclude built-in signals.
Sampling	The act of taking a random sample of certain types of requests to be stored and available in the console.

Signal exclusion rule	A type of rule that allows you to define arbitrary conditions to exclude a specific system signal (such as `XSS`).
Signal Sciences	The overall platform that protects a corp's sites.
Site (Workspace)	A single web application, bundle of web applications, API, or microservice that Signal Sciences can protect from attacks. Users can monitor events, set up blocking mode to block attacks, and create custom configurations on sites.
Site alerts	A custom alert that allows users to define thresholds for when to flag, block, or log an IP address.
Suspicious IP addresses	IP addresses that are approaching thresholds, but have not yet met or exceeded them.
Templated rule	A type of partially pre-constructed rule that, when filled out, allows you to block, allow, or tag certain types of requests.
Thresholds	A limit either set by Signal Sciences or custom set by users that must be exceeded for a certain event to happen. For example, suspicious IP addresses must exceed a certain threshold to become flagged.
User (role)	A user role that can edit site configurations on sites they are assigned to.
Users	All of the people who manage, edit, or just observe activity. A user belongs to a particular corp and is identified by an email address and password. A user can be a member of one or more sites.
Virtual Patch	A virtual patch prevents attacks of a known vulnerability in a module or framework by not allowing the attacks to reach the web app. This buys time to fix the underlying vulnerability while the virtual patch is protecting the app.

Amazon Linux NGINX 1.14.1+

Requirements

Our distribution release depends on the EPEL repository. You will need to ensure your system also has it installed.

Add the package repositories

Add the version of the NGINX package repository that you want to use:

NGINX version 1.18.0+ running either Amazon Linux 2 / Amazon Linux 2018.03

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey
      https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

NGINX version 1.14.1 < 1.17.9 on Amazon Linux 2

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

NGINX version 1.14.1 < 1.17.9 on Amazon Linux 2018.03

> **Note:** After Q2 2017, RHEL6 and CentOS 6 will exit Production Phase 2 according to the Red Hat Enterprise Linux Life Cycle. Only limited critical security fixes will be issued. You will need to review the lifecycle document for details and plan appropriately.

```
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Install the NGINX module

1. Install the Signal Sciences NGINX module by running the following command, replacing “`NN.NN`” with your NGINX version number:
   - NGINX version 1.18.0+ running either Amazon Linux 2 / Amazon Linux 2018.03

     ```
     sudo yum install nginx-module-sigsci-nxo-1.NN.NN*
     ```

   - NGINX version 1.14.1 < 1.17.9 on Amazon Linux 2 / Amazon Linux 2018.03

     ```
     sudo yum install nginx-module-sigsci-nxo-amzn-1.NN.NN*
     ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`) add the following lines to the global section after the `pid /run/nginx.pid;` line:

   ```
   load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
   ```

3. Restart the NGINX service to initialize the new module.
   - Amazon Linux 2

     ```
     systemctl restart nginx
     ```

   - Amazon Linux 2018.03

     ```
     restart nginx
     ```

Amazon Linux NGINX 1.10-1.14

Requirements

Our distribution release depends on the EPEL repository. You will need to ensure your system also has it installed.

For Red Hat CentOS 6, we currently only support Amazon Linux 2018.03 or earlier.

Add the package repositories

Add the version of the Red Hat CentOS package repository that you want to use.

Red Hat CentOS 7

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 6

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Enable Lua for NGINX

Some older versions of NGINX don't support native loading of Lua modules. Therefore, we require NGINX to be built with Lua and LuaJIT support. You must first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

Install the Lua NGINX Module

Install the dynamic Lua NGINX Module appropriate for your NGINX distribution:

NGINX.org distribution

- NGINX 1.12.1 or higher

  ```
  sudo yum install nginx-module-lua-`rpm -q --qf "%{VERSION}" nginx`
  ```

- NGINX 1.11

  ```
  sudo yum install nginx111-lua-module
  ```

- NGINX 1.10

  ```
  sudo yum install nginx110-lua-module
  ```

Amazon distribution

- NGINX 1.12.1 or higher

  ```
  sudo yum install nginx-module-lua-amzn-`rpm -q --qf "%{VERSION}" nginx`
  ```

- NGINX 1.11

  ```
  sudo yum install nginx111-lua-module
  ```

- NGINX 1.10

  ```
  sudo yum install nginx110-lua-module-amzn
  ```

Enable the Lua NGINX Module

1. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`) add the following lines to the global section after the line that starts with `pid`:

   ```
   load_module /usr/lib64/nginx/modules/ndk_http_module.so;
   load_module /usr/lib64/nginx/modules/ngx_http_lua_module.so;
   ```

2. Restart the NGINX service to initialize the new module:

   - Amazon Linux 2

     ```
     systemctl restart nginx
     ```

   - Amazon Linux 2015.09.01

     ```
     restart nginx
     ```

Check that Lua is loaded correctly

```
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
# Also, for some nginx-1.10.nn installed from nginx-extras package, you may
# need to specify the load directives.
# Given the above uncomment the following:
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768;
    # multi_accept on;
}
http {
init_by_lua '

local m = {}
local ngx_lua_version = "dev"

if ngx then
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
if not r then
  error("ERROR: No lua jit support: No support for SigSci Lua module")
else

  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
  end

end

';

}
```

Example of a successfully loaded config and its output

```
$ nginx -t -c <your explicit path>/sigsci_check_lua.conf

nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
```

Install the NGINX module

1. Install the module.

   ```
   sudo yum install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX service to initialize the new module.

   - Amazon Linux 2

     ```
     systemctl restart nginx
     ```

   - Amazon Linux 2015.09.01

     ```
     restart nginx
     ```

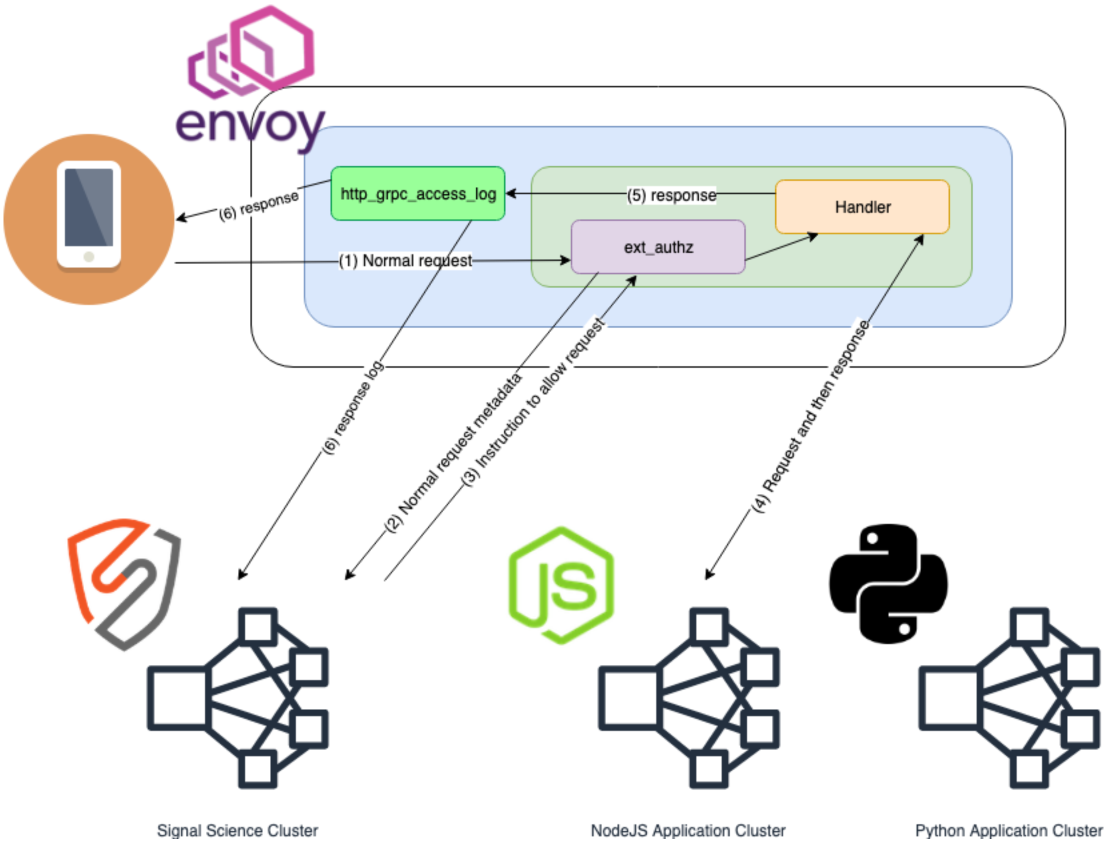
Envoy Proxy gRPC Authorization Mode

Overview

Support is available for the Envoy Proxy via builtin Envoy gRPC APIs implemented in the `sigsci-agent` running as a gRPC server. Envoy v1.11.0 or later is recommended, however, Envoy v1.8.0 or later is supported with limited functionality as noted in the documentation below.

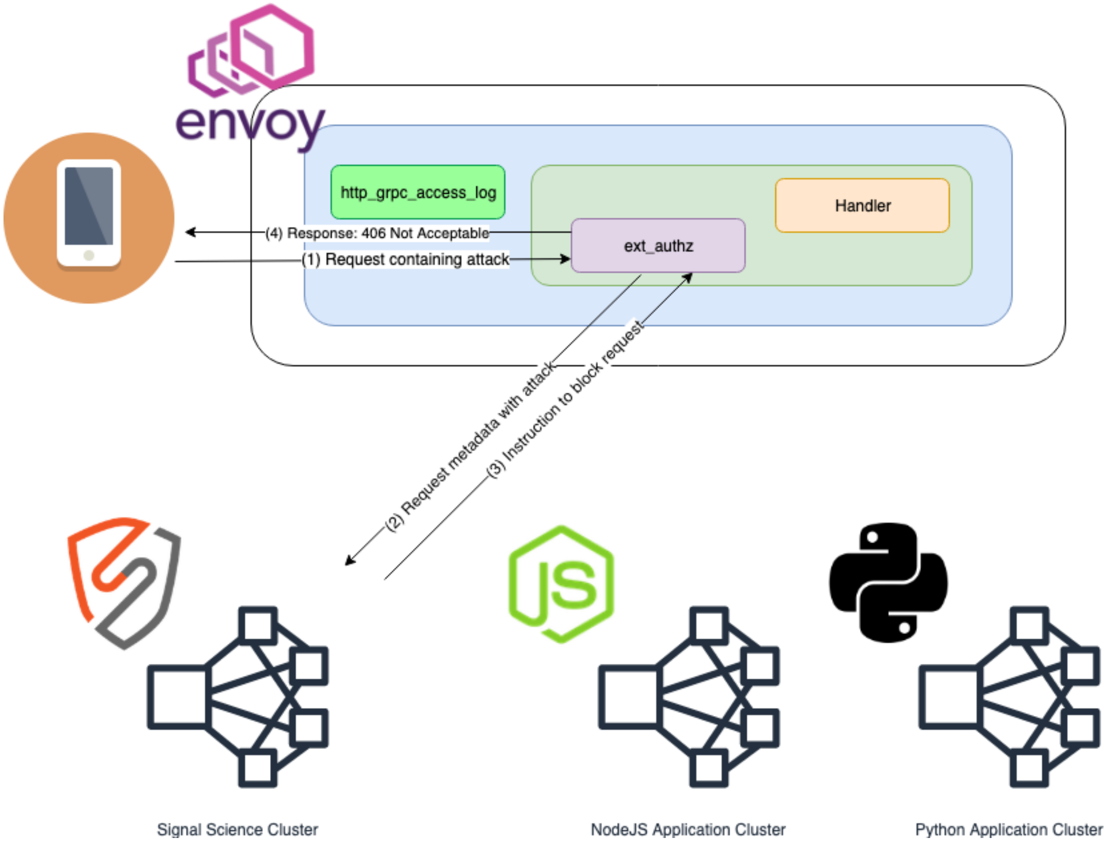
Currently Envoy (as of v1.11) does not support a bidirectional gRPC API for inspecting traffic. There are instead two separate gRPC APIs available to inspect traffic. The External Authorization HTTP filter (`envoy.ext_authz`) gRPC API allows the request to be held while waiting inbound request inspection, which allows for a request to be blocked if required. An additional gRPC AccessLog Service gRPC API can then be used to inspect the outbound request data. Using these two APIs together with the `sigsci-agent` running as a gRPC server allows for inspection in both directions using only Envoy builtin APIs. This allows web application inspection without installing a module for every upstream application. In this case the `sigsci-agent` is acting as the module.

Request Allowed (normal) Processing


envoy
http_grpc_access_log
(5) response
Handler
(6) response
ext_authz
(1) Normal request
(6) response log
(2) Normal request metadata
(3) Instruction to allow request
(4) Request and then response
Signal Science Cluster
NodeJS Application Cluster
Python Application Cluster

metadata is extracted for processing via the `sigsci-agent`.
2. Request metadata is sent to the `sigsci-agent` via gRPC `ext_authz` API
3. The `sigsci-agent` sends back an 'allow request' response allowing the request through the `ext_authz` HTTP filter to continue normal Envoy request processing.
4. Request makes it through any additional HTTP filters to the Handler, which processes the request and generates the response.
5. Request/Response metadata is extracted via the Envoy `gRPC AccessLog Service (als)` asynchronously for processing via the `sigsci-agent`.
6. In parallel, additional metadata, such as response headers and the HTTP status code, is sent to the `sigsci-agent` via gRPC `als` API for further processing while the response data is sent back to the originating client.

Request Blocked Processing


envoy
http_grpc_access_log
Handler
ext_authz
(4) Response: 406 Not Acceptable
(1) Request containing attack
(2) Request metadata with attack
(3) Instruction to block request
Signal Science Cluster
NodeJS Application Cluster
Python Application Cluster

This is the flow if the `sigsci-agent` blocks a request from being processed through Envoy.

1. Client request received by Envoy and routed to the Envoy `External Authorization (ext_authz)` HTTP filter where request metadata is extracted for processing via the `sigsci-agent`.
2. Request metadata is sent to the `sigsci-agent` via gRPC `ext_authz` API
3. The `sigsci-agent` sends back a 'block request' response, disallowing the request to continue being processed by the HTTP filter chain.
4. This triggers the `ext_authz` filter to generate a HTTP 406 response, blocking the request from any further processing.

Signal Sciences Agent Configuration

The `sigsci-agent` is normally installed as a sidecar via Kubernetes with a slightly different configuration than a normal install.

The `sigsci-agent` must be configured to run with an Envoy gRPC listener instead of the normal RPC listener. To do this, configure the Envoy gRPC listener via the `envoy-grpc-address` agent configuration option, which will then start instead of the default RPC listener.

Setting the configuration value in the `sigsci-agent` config file:

```
envoy-grpc-address = "0.0.0.0:8000"
```

Or setting the configuration value in the `sigsci-agent` environment:

configuration.

```
envoy-grpc-cert = "/path/to/cert.pem"
envoy-grpc-key = "/path/to/key.pem"
```

Or

```
SIGSCI_ENVOY_GRPC_CERT=/path/to/cert.pem
SIGSCI_ENVOY_GRPC_KEY=/path/to/key.pem
```

Additionally, it is recommended to enable response data processing. To do this, the `sigsci-agent` must be configured to expect response data from Envoy by setting the `envoy-expect-response-data` agent configuration option. By default, response data is ignored in the `sigsci-agent` as this is an optional Envoy configuration option in order to better support older versions of Envoy. If you are running Envoy v1.10 or higher, then you should enable this option.

Setting the configuration value in the `sigsci-agent` config file:

```
envoy-expect-response-data = 1
```

Or setting the configuration value in the `sigsci-agent` environment:

```
SIGSCI_ENVOY_EXPECT_RESPONSE_DATA=1
```

Some aspects of inspection in the `sigsci-agent` can be configured but generally should be left as the default. See `inspection-*` agent configuration for more details.

Envoy Configuration

Envoy must to be configured with an External Authorization HTTP filter (`envoy.ext_authz`) before the main handler filter to process request data and (optionally, though recommended) a gRPC AccessLog Service to process response data. To do this, multiple configuration items must to be added to the Envoy configuration: a cluster to handle the gRPC calls via the `sigsci-agent`, the `envoy.ext_authz` HTTP filter before the main handler, and the `envoy.http_grpc_access_log` service added to the `access_log` section of the HTTP listener filter if response data is to be enabled.

Adding the Signal Sciences Agent Cluster

A cluster must be added which is configured with the Envoy gRPC address used in the `sigsci-agent` configuration. Currently load balancing will not work correctly if response data is enabled as there is not a way to enable consistent hashing for gRPC services in Envoy (yet), so it is recommended not to configure load balancing at this time unless only the `envoy.ext_authz` API is being used without response data inspection.

```
  clusters:
  - name: sigsci-agent-grpc
    connect_timeout: 0.2s
    type: strict_dns
    #lb_policy: LEAST_REQUEST
    http2_protocol_options: {}
    #tls_context: {}
    ### You can also use 'hosts' below, but this is deprecated
    load_assignment:
      cluster_name: sigsci-agent-grpc
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: sigsci-agent
                port_value: 8000
```

The `address` is a resolvable hostname or IP for the `sigsci-agent` and the `port_value` must match that configured in the `sigsci-agent` configuration for the `envoy-grpc-address` option.

> via envoy-grpc-cert and envoy-grpc-key. If TLS is configured in the sigsci-agent, then just the empty tls_context must be configured (e.g., `tls_context: {}`) to let Envoy know to connect via TLS. If certificate validation is desired, then `validation_context` must be configured in the `tls_context` to specify a `trusted_ca` filename to use for validation. As gRPC services are HTTP/2 based, the `http2_protocol_options: {}` option is required so that traffic is sent to the `sigsci-agent` cluster as HTTP/2.

Adding the Envoy External Authorization HTTP Filter

The listener must have an External Authorization HTTP filter (`envoy.ext_authz`) added before the main handler which points at the `sigsci-agent` cluster.

```
http_filters:
- name: envoy.filters.http.ext_authz
  typed_config:
    "@type": type.googleapis.com/envoy.extensions.filters.http.ext_authz.v3.ExtAuthz
    transport_api_version: V3
    grpc_service:
        envoy_grpc:
            cluster_name: sigsci-agent-grpc
        timeout: 0.2s
    failure_mode_allow: true
    with_request_body:
      # Maximum request body bytes buffered and sent to the sigsci-agent
      max_request_bytes: 8192
      # NOTE: If allow_partial_message is set false, then any request over
      # the above max bytes will fail with an HTTP "413 Payload Too Large"
      # so it is recommended to set this to true.
      allow_partial_message: true
      # NOTE: By default, envoy carries the HTTP request body as a UTF-8 string
      # and it fills the body @ # field. To pack the request body as raw bytes,
      # set pack_as_bytes to true.
      pack_as_bytes: true
- name: envoy.filters.http.router
    typed_config:
      "@type": type.googleapis.com/envoy.extensions.filters.http.router.v3.Router
```

> **Note:** `failure_mode_allow: true` is so that this will fail open, which is recommended. And `timeout` allows failing *with the defined failure mode* (true for fail open, false for fail closed) after a given time duration. Once this is done, all HTTP requests will be first sent to the `envoy.ext_authz` filter handled by the `sigsci-agent` cluster. The `sigsci-agent` will then process requests and deny auth with a 406 HTTP status code if the request is to be blocked or allow the request through to the next HTTP filter if it is allowed. Any additional HTTP request headers are also added to the request as they are in other modules.

Adding the Envoy gRPC AccessLog Service

> **Note:** This is a recommended, but optional step. If it is configured in Envoy, then the agent *must* also be configured to expect response data by setting the `envoy-expect-response-data` agent configuration option as noted in the Signal Sciences Agent Configuration section. The Envoy External Authorization (`envoy.ext_authz`) HTTP Filter can only process request data. As the `sigsci-agent` needs the response data for full functionality, a gRPC AccessLog Service must be set up to send the response data to the `sigsci-agent`. To do this an `access_log` section must be added to the Envoy configuration under the listener filter (typically under the `envoy.http_connection_manager` filter) if it does not already exist. If it does exist, then it must be appended to.

Refer to the `access_log` configuration option of the HTTP Connection Manager for more details. An `envoy.http_grpc_access_log` entry must be added here (in addition to any other existing access log entries).

Recommended Configuration (see Current Limitations for further customizations to minimize limitations):

```
access_log:
- name: envoy.http_grpc_access_log
   typed_config:
    "@type": type.googleapis.com/envoy.extensions.access_loggers.grpc.v3.HttpGrpcAccessLogConfig
```

```
  grpc_service:
    envoy_grpc:
      cluster_name: sigsci-agent-grpc
    timeout: 0.2s
additional_request_headers_to_log:
# These sigsci-agent headers are required for correct processing:
- "x-sigsci-request-id"
- "x-sigsci-waf-response"
# Optionally, additional headers can be added that should be recorded:
- "accept"
- "content-type"
- "content-length"
additional_response_headers_to_log:
- "date"
- "server"
- "content-type"
- "content-length"
```

Current Limitations

Here are the current limitations when using the `sigsci-agent` with Envoy Proxy. As support for Envoy Proxy improves in the future, these limitations will be addressed and should be reduced.

No request bodies are processed by default

Prior to Envoy v1.10.0, the Envoy External Authorization did not send the request body. In all versions of Envoy, the request body is not included in the `ext_authz` call by default and it will not be inspected by the `sigsci-agent` unless configured.

For Envoy v1.10.0 or higher, support to include the request body is built in to the `envoy.ext_authz` configuration and it is now possible to configure the `with_request_body` in this section of the Envoy configuration as noted above.

For Envoy v1.11.0 or higher, support was extended to be able to detect partial bodies more accurately.

For HTTP/2 (and gRPC) support Envoy must be running a version later than v1.12.1. In Envoy v1.10.0 - v1.12.1 Envoy is not properly sending the request body using `with_request_body`.

However, it is possible to work around this Envoy limitation using Lua until an Envoy upgrade is possible. The following is an example Lua filter that can be used to pass on gRPC based bodies to the `sigsci-agent` for inspection:

To do this, the Lua HTTP filter (`envoy.lua`) HTTP filter can be configured before the `envoy.ext_authz` filter to add an internal `x-sigsci-encoded-body` header with this data. A small snippet of Lua code must be added to extract the body and add it to the request as follows:

```
http_filters:
- name: envoy.lua
  config:
    inline_code: |
      -- Add a special header to pass the encoded body
      function envoy_on_request(req)
        local len = 0
        local reqbody
        -- Determine the body length
        local cl = req:headers():get("content-length")
        if cl ~= nil then
          len = tonumber(cl)
        end
        -- gRPC does not have a content-length header to limit the body before buffering
        if len == 0 and req:headers():get("content-type") == "application/grpc" then
          -- Triggers buffering
          len = req:body():length()
        end
        -- Limit body length sent to the agent (adjust as needed)
```

```
                    -- Encode the body for use in a header value
                    local enc, t = string.gsub(reqbody, "[^%w]", function(chr)
                      return string.format("%%%02X",string.byte(chr))
                    end)
                    req:headers():add("x-sigsci-encoded-body", enc)
                  end
                end
          - name: envoy.ext_authz
            config:
              grpc_service:
                envoy_grpc:
                  cluster_name: sigsci-agent-grpc
                timeout: 0.2s
              failure_mode_allow: true
#              with_request_body:
#                max_request_bytes: 8192
#                allow_partial_message: true
          - name: envoy.router
            config: {}
```

No TLS handshake metadata is extracted

There is not currently a means for the `sigsci-agent` to see the TLS handshake metadata (e.g., cipher and protocol version) used in the originating request as this is not (yet) available in Envoy. Any TLS handshake metadata based signals will not be seen in the product for this site.

The following system signals are currently not supported due to this limitation:

- WEAKTLS

Only minimal request headers are recorded by default if there were only response-based signals

If the request was inspected by the `envoy.ext_authz` filter and no signals were issued, then the response will be processed by the `envoy.http_grpc_access_log` service. If a signal is found in the response data, then only minimal request headers will be recorded with the signal due to the API not being sent all request headers by default. However, if additional request headers are desired to be recorded, then these should be added via the `additional_request_headers_to_log` option of the `access_log` configuration in Envoy.

Currently these headers will automatically be added:

- Host
- User-Agent
- Referer
- X-Forwarded-For

Two `sigsci-agent` specific headers must be added. Additionally any additional request headers can be added explicitly via `additional_request_headers_to_log`:

```
additional_request_headers_to_log:
# These sigsci-agent headers are required for correct processing:
- "x-sigsci-request-id"
- "x-sigsci-waf-response"
# Optionally, additional headers can be added that should be recorded:
- "accept"
- "content-type"
- "content-length"
- "x-real-ip"
```

No response headers are processed by default

Similar to above with minimal request headers not being processed by the `envoy.http_grpc_access_log` service, there are no response headers sent to this API by default. Any headers that are desired to be recorded must be explicitly listed in the `additional_response_headers_to_log` option of the `access_log` configuration in Envoy as there is not currently any means to wildcard this. The following are recommended.

```
- "content-type"
- "content-length"
```

Next Steps

Verify the agent and module installation and explore module options.

Amazon Linux NGINX 1.9 or lower

Requirements

Our distribution release depends on the EPEL repository. You will need to ensure your system also has it installed.

For Red Hat CentOS 6, we currently only support Amazon Linux 2018.03 or earlier.

Add the package repositories

Add the version of the Red Hat CentOS package repository that you want to use.

Red Hat CentOS 7

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Red Hat CentOS 6

> **Note:** After Q2 2017, RHEL6 and CentOS 6 will exit Production Phase 2 according to the Red Hat Enterprise Linux Life Cycle. Only limited critical security fixes will be issued. You will need to review the lifecycle document for details and plan appropriately.

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/6/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Enable Lua for NGINX

Some older versions of NGINX don’t support native loading of Lua modules. Therefore, we require NGINX to be built with the third-party `ngx_lua` module. Because most older versions of NGINX do not support dynamically loadable modules, you will likely need to rebuild NGINX from source.

To assist you, we provide pre-built drop-in replacement NGINX packages already built with the `ngx_lua` module. This is intended for users who prefer not to build from source, or who either use a distribution-provided package or an official NGINX provided package. These pre-built packages are built to support much older distributions and are not gpg signed.

Flavors

We support three flavors of NGINX. These flavors are based on what upstream package we’ve based our builds on. All our package flavors are built according to the official upstream maintainer’s build configuration with the addition of the `ngx_lua` and `ngx_devel_kit` modules.

- distributions (Red Hat and Debian) these are the based off the official Debian NGINX packages. For Red Hat based Linux distributions we've based them off the EPEL packages as neither Red Hat or CentOS ship an NGINX package in their default distribution.
- **Stable** - The stable flavor is based off the official NGINX.org stable package releases.
- **Mainline** - The mainline flavor is based off the official NGINX.org mainline package releases.

Flavor version support

The following versions are contained in the various OS and flavor packages:

OS	Distribution	Stable	Mainline
Amazon Linux 2015.09.01	unsupported	1.8.1	1.9.10

The versions are dependent on the upstream package maintainer's supported version.

Yum repository setup for Amazon Linux 2015.09.01

1. Create a file `/etc/yum.repos.d/sigsci_nginx.repo` with the following contents:

    - Distribution (Amazon Linux 2015.09.01) flavor

        Note: Our distribution release depends on the EPEL repository, you will need to ensure your system also has it installed.

        ```
        [sigsci_nginx]
        name=sigsci_nginx
        priority=1
        baseurl=https://yum.signalsciences.net/nginx/distro/el6/$basearch
        repo_gpgcheck=1
        gpgcheck=0
        enabled=1
        gpgkey=https://yum.signalsciences.net/nginx/gpg.key
        sslverify=1
        sslcacert=/etc/pki/tls/certs/ca-bundle.crt

        [sigsci-nginx-noarch]
        name=sigsci_nginx_noarch
        priority=1
        baseurl=https://yum.signalsciences.net/nginx/distro/el6/noarch
        repo_gpgcheck=1
        gpgcheck=0
        enabled=1
        gpgkey=https://yum.signalsciences.net/nginx/gpg.key
        sslverify=1
        sslcacert=/etc/pki/tls/certs/ca-bundle.crt
        ```

    - Stable (Amazon Linux 2015.09.01) flavor

        ```
        [sigsci_nginx]
        name=sigsci_nginx
        priority=1
        baseurl=https://yum.signalsciences.net/nginx/stable/el6/$basearch
        repo_gpgcheck=1
        gpgcheck=0
        enabled=1
        gpgkey=https://yum.signalsciences.net/nginx/gpg.key
        sslverify=1
        sslcacert=/etc/pki/tls/certs/ca-bundle.crt
        ```

    - Mainline (Amazon Linux 2015.09.01) flavor

        ```
        [sigsci_nginx]
        name=sigsci_nginx
        priority=1
        baseurl=https://yum.signalsciences.net/nginx/mainline/el6/$basearch
        ```

```
    gpgkey=https://yum.signalsciences.net/nginx/gpg.key
    sslverify=1
    sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

2. Rebuild the `yum` cache for the Signal Sciences repository.

   ```
   yum -q makecache -y --disablerepo=* --enablerepo=sigsci_*
   ```

3. Install the version of NGINX provided by Signal Sciences.

   ```
   yum install nginx
   ```

Check Lua is loaded correctly

To verify Lua has been loaded properly, load the following config (`sigsci_check_lua.conf`) with NGINX:

```
# Config just to test for lua jit support
#
# Test from commandline as follows:
# nginx -t -c <explicit path>/sigsci_check_lua.conf
#

# The following load_module directives are required if you have installed
# any of: nginx110-lua-module, nginx111-lua-module, or nginx-lua-module
# for your nginx.org installation.
# Also, for some nginx-1.10.nn installed from nginx-extras package, you may
# need to specify the load directives.
# Given the above uncomment the following:
#
# load_module modules/ndk_http_module.so;
# load_module modules/ngx_http_lua_module.so;

events {
    worker_connections 768;
    # multi_accept on;
}
http {
init_by_lua '

local m = {}
local ngx_lua_version = "dev"

if ngx then
  -- if not in testing environment
  ngx_lua_version = tostring(ngx.config.ngx_lua_version)
  ngx.log(ngx.STDERR, "INFO:", " Check for jit: lua version: ", ngx_lua_version)
end

local r, jit = pcall(require, "jit")
if not r then
  error("ERROR: No lua jit support: No support for SigSci Lua module")
else

  if jit then
    m._SERVER_FLAVOR = ngx_lua_version .. ", lua=" .. jit.version
    if os.getenv("SIGSCI_NGINX_DISABLE_JIT") == "true" then
      nginx.log(ngx.STDERR, "WARNING:", "Disabling lua jit because env var: SIGSCI_NGINX_DISABLE_JIT=", "true")
    end
    ngx.log(ngx.STDERR, "INFO:", " Bravo! You have lua jit support=", m._SERVER_FLAVOR)
  else
    error("ERROR: No luajit support: No support for SigSci")
```

```
';


}
```

If the config is successfully loaded, the above script will create the following output:

```
$ nginx -t -c <your explicit path>/sigsci_check_lua.conf

nginx: [] [lua] init_by_lua:9: INFO: Check for jit: lua version: 10000
nginx: [] [lua] init_by_lua:22: INFO: Bravo! You have lua jit support=10000, lua=LuaJIT 2.0.4
nginx: the configuration file <your explicit path>/sigsci_check_lua.conf syntax is ok
nginx: configuration file <your explicit path>/sigsci_check_lua.conf test is successful
```

Install the NGINX module

1. Install the module with `yum`.

   ```
   sudo yum install sigsci-module-nginx
   ```

2. Add the following to your NGINX configuration file (located by default at `/etc/nginx/nginx.conf`) in the `http` context:

   ```
   include "/opt/sigsci/nginx/sigsci.conf";
   ```

3. Restart the NGINX service to initialize the new module.

   - Amazon Linux 2

     ```
     systemctl restart nginx
     ```

   - Amazon Linux 2015.09.01

     ```
     restart nginx
     ```

IBM HTTP Server

Limitations and considerations

To install the IBM HTTP Server:

- IHS must be installed in `/opt/IBM/HTTPServer`. If IHS is installed in a different path, use the appropriate path for your IHS installation.
- IHS must be installed on CentOS. If assistance is needed with another platform, contact Support.

Installation

1. Install the Signal Sciences agent for your OS.

2. If you're on IHS 9.0.0 or higher, download the Signal Sciences module package:

   Note: Replace `<VERSION>` with the latest module version found here: https://dl.signalsciences.net/?prefix=sigsci-module-apache/
   If using IBM HTTP Server (IHS), the Signal Sciences Apache module is the appropriate module to install as IHS is based on Apache HTTP Server.

   1. Download the Apache module.

      ```
      wget https://dl.signalsciences.net/sigsci-module-apache/<VERSION>/centos/el6/sigsci-module-apache-<VERSIO
      ```

   2. Extract the Apache module.

      ```
      tar -xzf sigsci-module-apache-<VERSION>.el6-1.x86_64.tar.gz
      ```

3. If you're on IHS 8.5* or lower, download the Signal Sciences module package:

   Note: Replace `<VERSION>` with the latest module version found here: https://dl.signalsciences.net/?prefix=sigsci-module-apache/

   1. Download the Apache module.

      ```
      wget https://dl.signalsciences.net/sigsci-module-apache/<VERSION>/centos/el7/sigsci-module-apache-<VERSIO
      ```

   2. Extract the Apache module.

      ```
      tar -xzf sigsci-module-apache-<VERSION>.el7-1.x86_64.tar.gz
      ```

4. Copy the module to the IBM HTTP Server modules directory.

   ```
   cp mod_signalsciences.so /opt/IBM/HTTPServer/modules
   ```

5. In `/opt/IBM/HTTPServer/conf/httpd.conf`, add the `LoadModule` directive.

   ```
   LoadModule signalsciences_module modules/mod_signalsciences.so
   ```

6. Restart the IBM HTTP Server.

   ```
   /opt/IBM/HTTPServer/bin/apachectl restart
   ```

Amazon Linux NGINX-Plus

Requirements

Our distribution release depends on the EPEL repository, you will need to ensure your system also has it installed.

For Red Hat CentOS 6, we currently only support Amazon Linux 2018.03 or earlier.

Add the package repositories

Add the Signal Sciences `yum` repositories.

Amazon Linux 2023

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/amazon/2023/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Amazon Linux 2

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
name=sigsci_release
baseurl=https://yum.signalsciences.net/release/el/7/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://yum.signalsciences.net/release/gpgkey https://dl.signalsciences.net/sigsci-agent/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Install the NGINX module

1. Install the Signal Sciences NGINX module by running the following command:

   - NGINX Plus 29

```
sudo apt-get install nginx-module-sigsci-nxp=1.23.2*
```

- NGINX Plus 27

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.21.6*
  ```

- NGINX Plus 26

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.21.5*
  ```

- NGINX Plus 25

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.21.3*
  ```

- NGINX Plus 24

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.19.10*
  ```

- NGINX Plus 23

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.19.5*
  ```

- NGINX Plus 22

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.19.0*
  ```

- NGINX Plus 21

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.17.9*
  ```

- NGINX Plus 20

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.17.6*
  ```

- NGINX Plus 19

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.17.3*
  ```

- NGINX Plus 18

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.15.10*
  ```

- NGINX Plus 17

  ```
  sudo yum install nginx-module-sigsci-nxp-amzn-1.15.7*
  ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`), add the following lines to the global section after the `pid /run/nginx.pid;` line:

   ```
   load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
   ```

3. Restart the NGINX service to initialize the new module.

   - Amazon Linux 2 & Amazon Linux 2023

     ```
     systemctl restart nginx
     ```

   - Amazon Linux 2018.03

     ```
     restart nginx
     ```

Alpine Linux NGINX 1.15.3+

Requirements

The Signal Sciences NGINX module for Alpine Linux requires NGINX v1.15.3 or higher.

Add the package repositories

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.18/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.18 - Bare Metal

```
sudo apk update && sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys
sudo echo https://apk.signalsciences.net/3.18/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Alpine 3.17 - Container

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.17/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.17 - Bare Metal

```
sudo apk update && sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys
sudo echo https://apk.signalsciences.net/3.17/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Alpine 3.16 - Container

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.16/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.16 - Bare Metal

```
sudo apk update && sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys
sudo echo https://apk.signalsciences.net/3.16/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Alpine 3.15 - Container

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.15/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.15 - Bare Metal

```
sudo apk update && sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys
sudo echo https://apk.signalsciences.net/3.15/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Alpine 3.14 - Container

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.14/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.14 - Bare Metal

```
sudo apk update && sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys
sudo echo https://apk.signalsciences.net/3.14/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Alpine 3.13 - Container

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.13/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.13 - Bare Metal

```
sudo echo https://apk.signalsciences.net/3.13/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Alpine 3.12 - Container

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.12/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.12 - Bare Metal

```
sudo apk update && sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys
sudo echo https://apk.signalsciences.net/3.12/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Alpine 3.11 - Container

```
apk update && apk add wget
wget https://apk.signalsciences.net/sigsci_apk.pub ; mv sigsci_apk.pub /etc/apk/keys
echo https://apk.signalsciences.net/3.11/main | tee -a /etc/apk/repositories && apk update
```

Alpine 3.11 - Bare Metal

```
sudo apk update && sudo apk add wget
sudo wget -q https://apk.signalsciences.net/sigsci_apk.pub ; sudo mv sigsci_apk.pub /etc/apk/keys
sudo echo https://apk.signalsciences.net/3.11/main | sudo tee -a /etc/apk/repositories && sudo apk update
```

Verify the download key

Verify the downloaded key contains the proper key by running the following command:

```
openssl rsa -pubin -in /etc/apk/keys/sigsci_apk.pub -text -noout
```

Expected modulus output:

```
Public-Key: (2048 bit)
Modulus:
    00:bb:23:1a:ef:0d:61:8f:8d:55:aa:ad:01:84:43:
    6c:46:42:42:ab:5b:ec:4e:4b:e2:e6:b6:e7:3d:45:
    b7:96:70:fe:16:95:aa:09:f1:90:82:40:e4:30:2b:
    9e:2a:03:e9:74:63:55:66:f0:db:8c:b9:5b:f8:45:
    5f:ad:4e:7a:14:da:02:83:c2:36:a0:84:74:a0:bb:
    f9:3f:03:c8:fe:80:6a:95:0c:17:22:55:40:30:18:
    51:d9:30:db:7c:1b:d0:06:4e:a9:51:1a:31:0e:33:
    f0:6e:ad:53:98:31:a5:ac:a3:a1:44:83:72:a1:ca:
    78:e3:24:70:ab:7a:0e:66:32:3b:f6:c9:90:16:dc:
    89:d0:52:7a:50:a8:f8:59:0a:34:12:2e:85:11:f5:
    80:0d:d4:7d:a7:7b:3b:d7:d9:1e:28:ed:bb:f7:08:
    2e:9f:73:a5:23:d8:53:b4:7e:21:dd:ae:92:4a:d0:
    5b:86:21:9c:82:05:21:29:eb:c1:ab:91:cd:1a:7b:
    95:6d:43:d3:1a:a9:62:2b:b0:95:9e:cf:18:82:64:
    02:f9:38:7e:7f:47:9f:d9:f3:ac:fd:2c:30:ff:75:
    b1:11:27:1c:7a:d6:ca:04:19:f8:31:80:42:e9:4a:
    0d:ab:d5:b8:ad:f2:35:31:a5:3f:98:19:99:fc:29:
    e8:4f
Exponent: 65537 (0x10001)
```

Install the NGINX module

1. Install the Signal Sciences NGINX module by running the following command, replacing `NN.NN` with your NGINX version number:

   ```
   apk add nginx-module-sigsci-nxo-1.NN.NN
   ```

2. In your NGINX config file (located by default at `/etc/nginx/nginx.conf`) add the following lines to the global section after the `pid /run/nginx.pid;` line:

3. Restart the NGINX service to initialize the new module.

```
sudo service nginx restart
sudo rc-service nginx restart
```

Automating user management (IdP)

An identity provider (IdP) is a system that stores and manages users' digital identities. We support automated user management through Okta.

Provisioning users via Okta enables you to automatically synchronize user access to your sites and their specific permission levels (also know as roles) for those sites. Specifically, you can:

- **Push new users.** New users created through Okta can be created in Signal Sciences.
- **Push profile updates.** Updates made to the user's profile through Okta can be pushed to Signal Sciences.
- **Push user deactivation and reactivation.** Deactivating the user or disabling the user's access to the application through Okta will delete the user in the third party application. Reactivating the user in Okta will recreate the user.

Limitations and considerations

When using Okta as your IdP, keep the following things in mind:

- A user that is provisioned by Okta cannot be modified or deleted in Signal Sciences. All changes must happen inside of Okta.
- Signal Sciences only accepts email addresses with letters that are lowercase. Email addresses with uppercase letters will result in erroneous behavior.
- If an existing user has the same email address as a user being provisioned within Okta, the accounts will be consolidated. Users won't have to be re-provisioned upon setup, but the new group assignments will override existing role and permissions.

Prerequisites

Before configuring the IdP, complete the following prerequisites:

- In your Signal Sciences account, enable Single Sign-On to use Okta as your SSO provider.
- In Okta, create a Signal Sciences application if you do not already have one. Follow the instructions listed in the Okta Signal Sciences application, which provide specific configuration information.
- Using our API, create an API Access Token in Signal Sciences and store it in a secure location for use later in this guide.

Configuring the IdP

To configure automated user management through Okta, follow these steps.

Enter configuration information

In the **Provisioning** tab of the Signal Sciences Okta application, enable provisioning. Enter the following information:

- **SCIM connector base URL:** Enter `https://dashboard.signalsciences.net/api/v0/corps/<corpname>/scim/v2` where `<corpname>` is the "name" of your Corp.
  - Your `<corpname>` is present in the address of your Signal Sciences console, such as `https://dashboard.signalsciences.net/corps/<corpname>/overview`.
  - Your `<corpname>` can also be retrieved from the List Corps API endpoint.
- **Unique identifier field for users:** Select **Email**.
- **Supported provisioning actions:** Select **Push New Users** and **Push Profile Updates**.
- **Authentication Mode:** Select **HTTP Header**.
- **Authorization:** Generate a Bearer Token from the API Access Token you generated earlier. The Bearer Token is created by base64 encoding a string composed of the email address associated with your user, a colon, and the API Access Token you generated.
  - An example command for creating a **Bearer Token** in bash:

    ```
    echo -n "user@example.com:c9e4bbc5-a5c4-19d3-b31f-691d8b2139fe" | base64
    ```

  - An example command for creating a **Bearer Token** in JavaScript:

    ```
    btoa("<signal_sciences_email>:<signal_sciences_access_token>") = "YW5keUBleGFtcGxlY29ycC5jb206ZXhhbXBsZXR
    ```

Test configuration

Click **Save** to save this configuration and proceed.

Enable provisioning features

After the settings are saved, select **Enable** for the following under **Provisioning to App**:

- Create Users
- Update User Attributes
- Deactivate Users

Click **Save** to save these settings and proceed.

After enabling provisioning, you may see a message that unmapped attributes exist on the application. This will not prevent provisioning; however, if you wish to map Signal Sciences attributes to your base Okta user profile, you may do so by mapping the following attributes:

- `userType` should be mapped onto a string attribute that will represent the user's `role`. The value of this must be a valid `role`: `owner`, `admin`, `user`, or `observer`.
- `entitlements` should be mapped onto a string array attribute that will represent the user's `sites`. This should be set to a string array representing the shortnames of sites the user should have access to, such as `www.example.com`.

Assigning a group or user to the application

The following instructions apply to assigning groups, though users will follow a nearly identical process.

1. In the Signal Sciences Okta application, click on **Assignments**. The assignments menu page appears.
2. From the **Assign** menu, select **Assign to Groups**. The group assignment menu page appears.
3. Select a group of users to provision to Signal Sciences. A window appears requesting additional attributes.
4. Select the **Role** for the assigned group. This can be one of **owner**, **admin**, **user**, or **observer**.
5. Click **Add Another** to add a site. This is the "short name" of the site that appears in your Site settings.
6. Click **Save and Go Back**.

Managing users with the IdP

User management includes both updates to attributes and user deletion.

Updating users

Updates to the group/user attributes will be synchronized to Signal Sciences including:

- The user's real name
- The user's assigned Signal Sciences role
- The user's assigned Signal Sciences sites

Signal Sciences does not support updating the user's email address, as it is the primary identifier for the user.

Deleting users

Signal Sciences users are removed via provisioning in a few ways:

- Remove the user from a group assigned to the Signal Sciences application
- Directly remove the user from the Signal Sciences application if they are directly assigned
- Deactivating the user in Okta

The user will be re-created if the user is reactivated or re-assigned to the Signal Sciences Okta application.

Troubleshooting

SCIM Provisioning was added to the Okta application in December 2020. If you have a Signal Sciences application in Okta that was created before December 2020, you may need to create a new Signal Sciences application in Okta in order to use SCIM provisioning.

If you have questions or difficulties with the Okta integration, reach out to our Support team for assistance.

Upgrading the NGINX Lua Module

> Check the NGINX Changelog to see what's new in the NGINX module.

Our Module package is distributed in our package repositories. If you haven't already, configure our repository on your system.

```
sudo apt-get update
```

NGINX 1.9 or Lower:

```
sudo apt-get install sigsci-module-nginx
```

NGINX 1.10.x:

```
sudo apt-get install sigsci-module-nginx nginx110-lua-module
```

NGINX 1.11.x:

```
sudo apt-get install sigsci-module-nginx nginx111-lua-module
```

NGINX 1.12.1 or higher:

```
sudo apt-get install sigsci-module-nginx nginx-module-lua
```

2. Restart NGINX. After upgrading the Lua module you'll need to restart your NGINX service.

Upgrading the Signal Sciences NGINX module on Red Hat/CentOS systems

1. Upgrade the NGINX Lua module package.

   ```
   sudo yum update
   ```

   NGINX 1.9 or Lower:

   ```
   sudo yum install sigsci-module-nginx
   ```

   NGINX 1.10.x:

   ```
   sudo yum install sigsci-module-nginx nginx110-lua-module
   ```

   NGINX 1.11.x:

   ```
   sudo yum install sigsci-module-nginx nginx111-lua-module
   ```

   NGINX 1.12.x:

   ```
   sudo yum install sigsci-module-nginx nginx-module-lua
   ```

2. **Restart NGINX**

   After upgrading the Lua module you'll need to restart your NGINX service.

Testing with attack tooling

After installing Signal Sciences, we recommend testing your setup by running attack tooling against your site to verify that attack data is being captured and blocking is working correctly.

While you can use any attack tooling for testing, we recommend using Nikto which tests a wide variety of vulnerabilities. While Nikto is running, Signal Sciences agents will identify any malicious or anomalous requests and send relevant metadata to our backend, after redacting any sensitive information.

This guide explains how to set up Nikto and run three different testing scenarios:

1. Testing attack tooling detection
2. Testing attack detection
3. Testing attack blocking

Before you begin

- Nikto requires Perl to be installed. Run `perl -v` to check if you have Perl installed on your system. If Perl is not found, you can download and install it from the Perl website.

Setting up Nikto

Nikto is a common open source tool used for running security tests against web servers. It can run on Linux, OS X, and Windows platforms. To set up Nikto:

1. Download the latest version of Nikto.
2. Using command prompt, navigate to the directory where you downloaded Nikto.
3. Enter `unzip nikto-master.zip` to unzip the file.
4. Enter `cd nikto-master/program/` to change directories to the program directory.
5. Run `./nikto.pl` to verify you are able to run Nikto. A default help message appears.

If you receive a permission denied error message, you can resolve the error by running `chmod +x nikto.pl` which makes the script executable. Then run `./nikto.pl` again.

Testing attack tooling detection

Using Nikto, you can test Signal Sciences' attack tooling detection capability.

To run this test:

1. Log in to the Signal Sciences console.
2. From the corp navigation bar, use the **Sites** menu to select the site you are conducting testing on. The Site Overview page appears.
3. Ensure the agent mode indicator in the site navigation bar displays **Not blocking**. In this mode, the agent logs requests but does not block anything. If the agent mode indicator displays **Blocking** or **Off**, update the behavior by clicking the agent mode indicator and then clicking the **Manage** link.
4. Using command prompt, enter `cd nikto-master/program/` to change directories to the program directory.
5. In a command prompt, run the following command to initiate the first Nikto scan of your site:

```
./nikto.pl -h http://www.example.com
```

While the attack is running, return to the Site Overview page in the Signal Sciences console and select the Overview dashboard from the dashboards menu. The Overview dashboard will display the attacks and anomalies within 30 seconds.

Testing attack detection

After verifying that Signal Sciences is detecting attack tooling, you can use Nikto to modify an attack to demonstrate an IP address being flagged due to injection attacks. You can do this by modifying the User-Agent string that is sent with each request.

To run this test:

1. Log in to the Signal Sciences console.
2. From the corp navigation bar, use the **Sites** menu to select the site you are conducting testing on. The Site Overview page appears.
3. From the **Events** card, click the **View** link on the IP address associated with the Nikto scanner host. The Events page appears.
4. Click the **Remove flag now** button on the flagged IP.
5. Click the **Remove flag** button.
6. Using command prompt, enter `cd nikto-master/program/` to change directories to the program directory.
7. Run the following command to initiate the Nikto scan:

```
./nikto.pl -useragent “MyAgent (Demo/1.0)” -h http://www.example.com
```

While the attack is running, return to the Site Overview page in the Signal Sciences console and select the Overview dashboard from the dashboards menu. The Overview dashboard will display the attacks and anomalies within 30 seconds. Unlike in the previous test, you should see signals from a variety of attacks, not just attack tooling. This means modifying the User-Agent string worked and the IP address will eventually be flagged based on the various attacks.

Testing attack blocking

malicious traffic from a particular IP (aggregated across all of our agents) and flag that IP if it exceeds specific thresholds in a 1, 10, or 60 minute window. Once an IP is flagged, we block all malicious traffic from that IP. Traffic is blocked for a default 24 hours. You can use site alerts to decrease the blocking time period. During the blocking time period, requests that don’t contain an attack will be allowed, preventing Signal Sciences from breaking normal traffic.

For the final test, enable blocking mode and use Nikto to demonstrate how Signal Sciences allows legitimate traffic to continue accessing the site while blocking malicious traffic from the same IP address. To perform this test, you will need to use a web browser that is on the same system you are running the scan from.

> **Note:** Before continuing, make sure to remove the scanning IP address from the flagged list.

To run this test:

1. Log in to the Signal Sciences console.
2. From the corp navigation bar, select the site you are conducting testing on. The Site Overview page appears.
3. Click on the agent mode indicator in the site navigation bar and click the **Manage** link. The Agent Configurations page appears.
4. Update the agent behavior to **Blocking**.
5. Click the **Update** button.
6. In a browser, access your website.
7. Using command prompt, enter `cd nikto-master/program/` to change directories to the program directory.
8. Run the following command to initiate the Nikto scan:

   ```
   ./nikto.pl -useragent “MyAgent (Demo/1.0)” -D V -T 9 -h http://www.example.com
   ```

While the scan is running:

- use the browser window to navigate your site to confirm that legitimate user traffic is not blocked.
- observe from the command shell window that requests containing attacks are blocked with a 406 response code. An HTTP 406 is used so as to not trigger operational alarms as a 500 or 404 would. Additionally, by using a unique code like 406, you can customize the error message that the server returns.

Repeat the scan as many times as desired.

You can also manually verify blocking by visiting your site with a malicious payload (e.g., `https://www.example.com/?q=<script>alert('xss')</script>`).

Datadog

Events Feed

Our Datadog event integration creates an event when IP addresses are flagged on Signal Sciences.

Adding a Datadog integration

1. Log in to Datadog.
2. Click **Integrations** in the navigation bar on the left. The integrations menu page appears.
3. Click the **APIs** tab. The integrations API menu page appears.
4. Click **Create API Key** button. The new API key menu appears.
5. Create a new API key by following the steps.
6. Copy the provided **API Key**.
7. Log in to the Signal Sciences console.
8. From the **Sites** menu, select a site if you have more than one site.
9. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
10. Click **Add site integration**. The add site integration menu page appears.
11. Select the **Datadog Alert** integration. The Datadog Alert integration setup page appears.
12. In the **API Key** field, enter the **API Key** created in Datadog.
13. Select if you want to be alerted regarding **All activity** or **Specific activity**.
14. If you selected **Specific activity**, use the **Activity** menu to choose which types of activities you want the integration to create alerts for.

Activity type	Description
`flag`	An IP address was flagged
`agentAlert`	An agent alert was triggered

Dashboard

Datadog has a default dashboard which is populated with StatsD metrics from the Signal Sciences agent. To use this functionality:

1. Find and install the Signal Sciences integration tile in Datadog integrations tab.
2. Confirm that the Datadog agent is configured to listen for StatsD events: https://docs.datadoghq.com/developers/dogstatsd/
3. Configure the Signal Sciences agent to use `dogstatsd`:
   - Add the following line to each agent's agent.config file:

     ```
     statsd-type = "dogstatsd"
     ```

   - When this is done the agent's `statsd` client will have tagging enabled and metrics such as `sigsci.agent.signal.<SIGNAL_TYPE>` will be sent as `sigsci.agent.signal` and tagged with `signal_type:<SIGNAL_TYPE>` (e.g., `sigsci.agent.signal.http404` => `sigsci.agent.signal tag signal_type:http404`).
   - If using Kubernetes to run the Datadog Agent, make sure to enable DogStatsD non local traffic as described in the Kubernetes DogStatsD documentation.
4. Configure the SigSci agent to send metrics to the Datadog agent by adding the following line to each agent's agent.config file:

   ```
   statsd-address="<DATADOG_AGENT_HOSTNAME>:<DATADOG_AGENT_PORT>"
   ```

5. Verify that the **Signal Sciences - Overview** dashboard is created and starting to capture metrics.

Anonymizing IP addresses

IP Anonymization is a site-level customization that changes the way Signal Sciences stores and uses remote client IP addresses. By default IP addresses are not anonymized. When a customer chooses to enable IP Anonymization, agents for a specific site will anonymize an IP address before sending it to the cloud. Signal Sciences will convert IP addresses into anonymized IPv6 addresses by performing a one-way hash. As a result, Signal Sciences databases will not have knowledge of the actual IP address and it will appear anonymized throughout the console.

Actual IP addresses are converted to anonymous IPv6 addresses using rfc7343.

The IP address is anonymized in all headers and data fields with the anonymized IPv6 address. In addition, the actual IP address is truncated by setting the last octet of an IPv4 IP address and the last 80 bits of an IPv6 address to zeros and stored as metadata on the record.

> **Note:** The following features will not work when IP Anonymization is enabled:
>
> - DNS lookups
> - CIDR support in the search console
> - Network Data Insights (partial functionality)

Enabling IP anonymization

To enable IP anonymization, complete the following steps:

1. Log in to the Signal Sciences console.
2. From the corp navigation bar, use the **Sites** menu to select a site.
3. From the **Manage** menu, select **Site Settings**. The Site Settings menu page appears.
4. Click the **Agent Configurations** tab. The Agent Configurations form appears.
5. Under **IP Anonymization**, select **Enabled**. A warning appears stating some functionality will not work with IP Anonymization enabled.
6. Click the **I understand** button.
7. Click the **Update** button.

Using system signals

The following information provides you with details about the various system signals:

interface.

- **Usable in:** outlines where a signal can be used. The options are Lists, Rate Limit Rules, Request Rules, or Signal Exclusions. None indicates that the signal may be provided but cannot be used outside of its informational context.
- **Description:** an outline of what the signal means or what it indicates.

Attacks

Attack signals are labels that describe malicious requests that contain attack payloads designed to hack, destroy, disable, steal, gain unauthorized access, and otherwise take harmful actions.

Long name	Short name	Usable in	Description
Attack Tooling	`USERAGENT`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Attack Tooling is the use of automated software to identify security vulnerabilities or to attempt to exploit a discovered vulnerability
AWS SSRF	`AWS-SSRF`	• Templated Rule	Server Side Request Forgery (SSRF) is a request which attempts to send requests made by the web application to target internal systems. AWS SSRF attacks use SSRF to obtain Amazon Web Services (AWS) keys and gain access to S3 buckets and their data.
Backdoor	`BACKDOOR`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	A backdoor signal is a request that attempts to determine if a common backdoor file exists on a system. The signal generally matches known backdoor filenames. Traditionally these filenames appear with PHP file extensions like `admin.php` and `r57.php`. For many users, when these paths return a 200 or a larger response than expected, it may indicate that their system has been compromised or they are unknowingly hosting a backdoor file.
Command Execution	`CMDEXE`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Command Execution is the attempt to gain control or damage a target system through arbitrary system commands by means of user input
Cross Site Scripting	`XSS`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Cross-Site Scripting is the attempt to hijack a user's account or web-browsing session through malicious JavaScript code
Directory Traversal	`TRAVERSAL`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Directory Traversal is the attempt to navigate privileged folders throughout a system in hopes of obtaining sensitive information
Log4J JNDI	`LOG4J-JNDI`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Log4J JNDI attacks attempt to exploit the Log4Shell vulnerability present in Log4J versions earlier than 2.16.0
SQL Injection	`SQLI`	• Lists	SQL Injection is the attempt to gain access to an application or obtain privileged information by executing arbitrary database queries

- Rate Limit Rules
- Request Rules
- Signal Exclusion

Anomalies

Anomaly signals are labels that describe abnormal requests. While not inherently malicious, abnormal requests may be indicative of unwanted or abusive traffic. Examples include malformed request data and requests originating from known scanners.

Long name	Short name	Usable in	Description
Abnormal Path	`ABNORMALPATH`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Abnormal Path indicates the original path differs from the normalized path (e.g., `/foo/./bar` is normalized to `/foo/bar`)
Bad Hop Headers	`BHH`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Bad Hop Headers indicate an HTTP smuggling attempt through either a malformed Transfer-Encoding (TE) or Content-Length (CL) header, or a well-formed TE and CL header
Blocked Requests	`BLOCKED`	None	Requests blocked by Signal Sciences
Code Injection PHP	`CODEINJECTION`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Code Injection is the attempt to gain control or damage a target system through arbitrary application code commands by means of user input.
Compression Detected	`COMPRESSED`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	The POST request body is compressed and cannot be inspected. For example, if a `Content-Encoding: gzip` request header is specified and the POST body is not plain text.
Datacenter Traffic	`DATACENTER`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Datacenter Traffic is non-organic traffic originating from identified hosting providers. This type of traffic is not commonly associated with a real end user.
Double Encoding	`DOUBLEENCODING`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Double Encoding checks for the evasion technique of double encoding html characters
Duplicate Header Names	`DUPLICATE-HEADERS`	• Lists • Rate Limit Rules	A request that has duplicate header field names. This may represent a programming error or an automated or malicious request. Current detected

		Rules • Signal Exclusion	Transfer-Encoding.
Fastly Unknown Backend	`FASTLY-UNKNOWN-BACKEND`		Indicates a request to a backend that does not exist in your edge security service.
Forceful Browsing	`FORCEFULBROWSING`	• Signal Exclusion	Forceful Browsing is the failed attempt to access admin pages
GraphQL API	`GRAPHQL-API`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Indicates a GraphQL API request.
GraphQL Duplicate Variables	`GRAPHQL-DUPLICATE-VARIABLES`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Indicates a GraphQL request that contains duplicated variables.
GraphQL IDE	`GRAPHQL-IDE`	• Rate Limit Rules • Request Rules	Indicates a request originating from a GraphQL Interactive Development Environment (IDE).
GraphQL Introspection	`GRAPHQL-INTROSPECTION`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Indicates an attempt to obtain the schema of a GraphQL API. The schema can be used to identify which resources are available, informing subsequent attacks.
GraphQL Max Depth	`GRAPHQL-DEPTH`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Indicates a request has reached or exceeded the maximum depth allowed on the server for GraphQL API queries
GraphQL Missing Required Operation Name	`GRAPHQL-MISSING-REQUIRED-OPERATION-NAME`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Indicates a request has multiple GraphQL operations but does not define which operation to execute.
GraphQL Syntax	`GRAPHQL-SYNTAX`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Indicates a request that contains invalid GraphQL syntax. This may be related to a programming error or a malicious request.
GraphQL Undefined Variable	`GRAPHQL-UNDEFINED-VARIABLES`	• Lists • Rate Limit Rules	Indicates a request made to a GraphQL API containing more variables than expected by a function. This can be used to obfuscate malicious requests.

		Rules • Signal Exclusion	
HTTP 403 Errors	`HTTP403`	• Signal Exclusion	Forbidden. This is commonly seen when the request for a url has been protected by the server’s configuration.
HTTP 404 Errors	`HTTP404`	• Signal Exclusion	Not Found. This is commonly seen when the request for a page or asset does not exist or cannot be found by the server.
HTTP 429 Errors	`HTTP429`	• Signal Exclusion	Too Many Requests. This is commonly seen when rate-limiting is used to slow down the number of active connections to a server.
HTTP 4XX Errors	`HTTP4XX`	• Signal Exclusion	4xx Status Codes commonly refer to client request errors
HTTP 500 Errors	`HTTP500`	• Signal Exclusion	Internal Server Error. This is commonly seen when a request generates an unhandled application error.
HTTP 503 Errors	`HTTP503`	• Signal Exclusion	Service Unavailable. This is commonly seen when a web service is overloaded or sometimes taken down for maintenance.
HTTP 5XX Errors	`HTTP5XX`	• Signal Exclusion	5xx Status Codes commonly refer to server related issues
HTTP Response Splitting	`RESPONSESPLIT`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Identifies when CRLF characters are submitted as input to the application to inject headers into the HTTP response
Invalid Encoding	`NOTUTF8`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Invalid Encoding can cause the server to translate malicious characters from a request into a response, causing either a denial of service or XSS
JSON Encoding Error	`JSON-ERROR`	• Signal Exclusion	A POST, PUT, or PATCH request body that is specified as containing JSON within the `Content-Type` request header but contains JSON parsing errors. This is often related to a programming error or an automated or malicious request.
Malformed Data in the request body	`MALFORMED-DATA`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	A POST, PUT or PATCH request body that is malformed according to the `Content-Type` request header. For example, if a `Content-Type: application/x-www-form-urlencoded` request header is specified and contains a POST body that is json. This is often a programming error, automated or malicious request.
Malicious IP Traffic	`SANS`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Signal Sciences regularly imports SANS Internet Storm Center list of IP addresses that have been reported to have engaged in malicious activity
Network Effect	`SIGSCI-IP`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Whenever an IP is flagged due to a malicious signal by our decision engine, that IP will be propagated to all customers. We then log subsequent requests from those IP addresses that contain any additional signal for the duration of the flag.

Missing `Content-Type` request header	`NO-CONTENT-TYPE`	• Rate Limit Rules • Request Rules • Signal Exclusion	A POST, PUT or PATCH request that does not have a `Content-Type` request header. By default application servers should assume `Content-Type: text/plain; charset=us-ascii` in this case. Many automated and malicious requests may be missing `Content Type`.
No User Agent	`NOUA`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Many automated and malicious requests use fake or missing User-Agents to make it difficult to identify the type of device making the requests
Null Byte	`NULLBYTE`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Null bytes do not normally appear in a request and indicate the request is malformed and potentially malicious
Private Files	`PRIVATEFILE`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Private files are usually confidential in nature, such as an Apache .htaccess file, or a configuration file which could leak sensitive information
Scanner	`SCANNER`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Identifies popular scanning services and tools
SearchBot Impostor	`IMPOSTOR`	• Templated Rule	Search bot impostor is someone pretending to be a Google or Bing search bot, but who is not legitimate
Site Flagged IP	`SITE-FLAGGED-IP`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Indicates a request was received from an IP that was flagged for exceeding attack thresholds for a specific site.This signal is only included with the Premier platform.
Tor Traffic	`TORNODE`	• Lists • Rate Limit Rules • Request Rules • Signal Exclusion	Tor is software that conceals a user's identity. A spike in Tor traffic can indicate an attacker trying to mask their location.
Weak TLS	`WEAKTLS`	• Signal Exclusion	Weak TLS. A web server's configuration allows SSL/TLS connections to be established with an obsolete cipher suite or protocol version. This signal is based on inspecting a small percent of requests. Also, some architectures and Signal Sciences' language SDK modules do not support this signal.
XML Encoding Error	`XML-ERROR`	• Signal Exclusion	A POST, PUT, or PATCH request body that is specified as containing XML within the `Content-Type` request header but contains XML parsing errors. This is

About the Corp Overview page

The Corp Overview page provides an at-a-glance view of all the sites in your corp, including which of your sites:

- is seeing the most traffic.
- is attacked the most.
- is seeing the most blocked traffic.
- has the most flagged, malicious IPs.

In addition to high-level stats, the Corp Overview page provides attack type and source breakdowns, enabling you to better visualize how your sites are being attacked.

Before you begin

Be sure you know how to access the web interface controls.

About the Corp Overview page

The Corp Overview page surfaces relevant data about your corp and its sites. To navigate to the Corp Overview page, click the name of your corp in the upper left corner of the web interface.

The Corp Overview page organizes data about your corp into the following sections:

- **Corp cards:** cards that represent request metrics as graphs.
- **Site Summaries:** a table listing the most frequent attack types and sources for a site.
- **Top Signals:** tables containing signal data.

You can use the Corp Overview page controls to change the time frame over which to display data for the entire page.

-7d 7 days ago

May 2022 June 2022 July 2022

Corp cards

The Corp Overview page surfaces request metrics for all of your sites through the following cards:

- **Request Volume**: the number of requests your corp receives, the number of requests that have at least one attack signal, and the number of requests that were blocked.
- **Attack Requests**: the number of malicious requests per site. The card displays a maximum of 10 sites.
- **Blocked Requests**: the number of requests that were blocked. The card displays a maximum of 10 sites.

Hovering over any part of a graph displays a timestamp indictor that updates itself as you move your cursor.

Site Summaries table

The Site Summaries table highlights the most frequent attack types and attack sources (e.g., the regions where the attacks originated) for each site. You can use a search bar to filter the table by site and the site menu to view all sites, sites with attack requests, or sites without attack requests. The table contains these columns:

- **Site**: the name of the site and the total number of requests the site has received.
- **Requests with Attack Signals**: the number of requests that were blocked by site alerts and the number of requests that have at least one attack signal.
- **Attack Signals**: the most frequent attack types for that site.
- **Countries**: the top three regions where the attacks originated.
- **Flagged IPs**: the number of IPs with subsequent malicious requests that were blocked due to a threshold of malicious requests being exceeded.

Top Signals tables

The Top Signals section surfaces signal data from your corp in the following tables:

- **Attack Signals**: displays data related to malicious requests.

- **Corp Signals**: displays data related to signals created at the corp level.

You can use a search bar to filter the tables by signal. The tables contain the following columns:

- **Signal**: the name of the signal.
- **Total Requests**: the number of requests that were tagged with the signal in your corp.
- **Top sites**: the sites that had the highest number of requests with that signal.
- **Requests per Site**: the number of requests tagged with that signal per site.

What’s next

Dig deeper into details about the web interface controls.

Converting requests to rules

From the Requests page, you can convert individual requests into pre-populated rules, enabling you to allow, block, rate limit, and tag similar requests. To convert a request into a rule, follow these steps:

1. Log in to the Signal Sciences console.

2. From the **Sites** menu, select a site if you have more than one site.

3. Click **Requests**. The requests page appears.

4. Locate or search for the request you want to convert into a rule.

5. Click **View request detail**. The request details page appears.

6. Click **Convert to rule** in the upper-right corner. The rule builder menu page appears.

7. Under **Type**, select the type of rule you want to make (**Request**, **Rate limit**, or **Signal exclusion**).

8. Under **Conditions**, select which characteristics of the request you want to convert into rule conditions. For example, selecting **IP Address** and **Path** will create conditions in the rule that look for the specific IP address and path featured in the request.

9. Click **Continue**. A pre-built rule with conditions featuring the request characteristics you selected

10. Under **Conditions**, modify the rule as needed by adding and editing rule conditions.

11. Under **Actions**, select which actions the rule should take (e.g., **Block**, **Allow**, or **Add signal**). Additional actions can be added by clicking **Add action**.

12. Under **Status**, optionally disable the rule by deselecting the **Always enabled** toggle. By default, rules are automatically enabled when created unless specifically disabled.

    You can optionally set the rule to automatically disable after a set period of time. Click **Change expiration** and select a duration from the menu.

13. In the **Description** field, enter a description for the rule.

14. Click **Create site rule**.

Managing sites

From the Sites page, you can add, edit, and remove sites in your corp.

Accessing the Sites page

To access the Sites page, follow these steps:

1. Log in to the Signal Sciences console.
2. From the corp navigation bar, click the **Corp Manage** menu and then select **Sites**. The Sites page appears.

Adding sites

To add a site, follow these steps:

1. From the **Sites** page, click the **Add site** button. The add site menu page appears.

3. Enter a short name for the new site in the **Short name** text box. The short name is used in URLs and the API (e.g., https://dashboard.signalsciences.net/corps/SHORT-NAME/).

   Note: By default, your corp has a limited number of sites. If you need more, contact support for assistance.

Editing sites

To edit a site, follow these steps:

1. From the **Sites** page, click the name of the site that you want edit. The Site Settings page appears and displays the Name form.
2. Fill out the fields on the **Name** form as follows:
   - In the **Display name** field, enter a friendly name for the site. The web interface uses the Display name.
   - In the **Short name** field, enter a short name for the site. The short name is used in the URL.
3. Click the **Update** button.
4. Click the **Agent Configurations** tab. The Agent Configurations form appears.
5. Fill out the Agent Configurations form as follows:
   - In the **Agent mode** menu, select the agent mode for the site.
   - In the **IP anonymization** menu, select **Disabled** to not anonymize IP addresses or select **Enabled** to convert IP addresses into anonymized IPv6 addresses.
   - In the **Client IP headers** area, add a header by clicking the **Add header** button and then entering the name of the header in the **Header** field. Remove a header by clicking the **Delete header** button to the right of a header name.
   - In the **Response code** field, enter a site default blocking response code. All blocking actions will return the site default blocking response code unless a different response code is specified in a rule. Supported response codes are 301, 302, and 400-599.
   - If you entered `301` or `302` in the **Response code (optional)** field then, in the **Redirect URL (optional)** field, enter the absolute or relative URL of the redirect location. See Using redirect custom response codes.
6. Click the **Update** button.
7. Click the **Users** tab. The list of users assigned to the site appears.
8. Manage the users assigned to the site.

Deleting sites

If you have an owner role, you can delete sites in your corp.

Limitations and considerations

A site cannot be deleted if it:

- Is the site you are currently accessing in the console
- Is the last site remaining for the corp
- Has users that aren't members of any other sites

If you would like to delete a site meeting any of the conditions listed above, reach out to our support team.

Deleting a site

To delete a site, follow these steps:

1. From the **Sites** page, click the name of the site that you want to delete. The Site Settings page appears.
2. Click the **Delete site** button. The delete site confirmation window appears.
3. Review the warnings associated with deleting a site and check the **I understand the consequences of deleting a site** box.
4. Click the **Delete** button.

Troubleshooting agent response codes

If something abnormal occurs during request processing, the Signal Sciences agent will return an error agent response code (e.g., -2, -1, and 499) that you can use to help resolve the issue.

Troubleshooting -2, -1, and 0 agent response codes

The -2, -1, and 0 agent response codes are error codes applied to requests that weren't processed correctly. There are a few reasons why this can happen but they tend to fall into two major categories:

- The post or response couldn't be matched to the request
- The module timed out waiting for a response from the agent

Request and response mismatch

Specific server response codes

The following server response codes cause NGINX to skip the phases that normally run. Due to their nature, they cause NGINX to finish processing the request without it being passed to the Signal Sciences module:

- 400 (Bad Request)
- 405 (Not Allowed)
- 408 (Request Timeout)
- 413 (Request Entity Too Large)
- 414 (Request URI Too Large)
- 494 (Request Headers Too Large)
- 499 (Client Closed Request)
- 500 (Internal Server Error)
- 501 (Not Implemented)

Look for NGINX return directives

Look for custom NGINX configurations or Lua code that could be redirecting the request. This is almost always due to `return` directives in an NGINX configuration file. There could be `return` directives used to redirect specific pages to `www`, `https`, or a new URL. The `return` directive stops all processing, causing the request to not be processed by the Signal Sciences module. For example:

```
location /oldurl {
     return 302 https://example.com/newurl/
}
```

These would need to be updated to force the request to be processed by our agent first. Calling the `rewrite_by_lua_block` directly allows you to force the Signal Sciences module to run first and then perform the return statement for NGINX:

```
location /oldurl {
     rewrite_by_lua_block {
          sigsci.prerequest()
          return ngx.exit(302 "https://example.com/newurl/")
     }
     #return 302 https://example.com/newurl/
}
```

Agent restarted

Request and response mismatches can also be due to restarting the agent. If the agent is restarted after the request is processed, but before the response is processed, the agent will not see the response and fail to attribute it to the request, resulting in an error agent response code.

Module timing out

When the module receives a request, it sends it to the agent for processing. The module then waits for a response from the agent (whether or not to block) for a set amount of time (typically 100ms). If the agent doesn't process the request within that time, the module will time out and default to failing open, allowing the request through. These requests that failed open will have error agent response codes applied to them.

Module timeouts are most commonly due to insufficient resources allocated to the agent. This can be a result of host or agent misconfiguration, such as the agent being limited to too few CPU cores.

This can also be due to a high volume of traffic to the host. If requests are coming in faster than the agent can process them, subsequent requests will be queued for processing. If a queued request reaches the timeout limit, then the module will fail open and allow the request through.

Similarly, certain rules designed specifically for penetration testing can take longer to run than traditional rules. This can result in requests queueing and timing out due to the increased processing time per request.

Look at response time

Requests that are timing out will have a high response time, exceeding the default timeout of 100ms.

Look at agent metrics

From the Agents page, you can access metrics for each agent. These metrics can help you diagnose the issue.

Connections dropped

The Connections dropped metric indicates the number of requests that were allowed through (or "dropped").

CPU usage

- The Agent CPU metric indicates the total CPU percentage for the number of cores in use by the agent. For example, if the agent were using 4 cores, then 400% would be the maximum.

CPU allocation and containerization

There are known issues with agents running within containers. It's possible for agents to have insufficient CPU to process requests, due to a low number of CPUs (cores) allocated to the container by the `cgroups` feature.

We recommend the container running the agent should be given at least 1 CPU. If both NGINX and the agent are running in the same container, then we recommend allocating at least 1.5 CPUs.

Troubleshooting the 499 agent response code and the 504 HTTP status code

If a client is making a request and the Cloud WAF Application Load Balancer (ALB) does not receive the first header byte within 60 seconds of the TCP connection being established, the requesting client will receive a 504, while the SigSci Agent will respond with a 499. This means the requesting client, if making a long-standing request through a browser, will receive a 504 error in the browser, while the SigSci Console will show a 499 for the request.

The long-standing request will need to be optimized to meet the 60 second threshold. If the request cannot be optimized, reach out to our support team for additional details.

Relevant timeouts in the Cloud WAF architecture

- The Cloud WAF agent has 60 seconds to start sending a response to the ALB
- The Cloud WAF agent has 10 seconds to negotiate TLS with the upstream
- The Cloud WAF agent has 30 seconds to establish an HTTP connection to the upstream

Further help

If you're unable to resolve an agent response code issue, generate an agent diagnostic package by running `sigsci-agent-diag`, which will output a `.tar.gz` archive with diagnostic information. Then, reach out to our support team for additional details. When you contact us, be sure to provide the diagnostic `.tar.gz` archive and include console links to the requests and agents affected.

Agent Configuration

For most installations, `accesskeyid` and `secretaccesskey` will be the only fields that require configuring; the default agent configuration will suffice for everything else. However, some environments will want to use additional options to better suit their environment.

The agent configuration is flexible enough to work in all environments. Most configuration options are available in three forms: config file, command line, and by setting environment variables.

> **IMPORTANT:** Agent configuration options listed as "experimental" are not fully developed and are subject to change. Use caution when building automated processes involving these options as their functionality may change as they mature.

Configuration Options

The following are the current configuration options (as of v4.43.0 on the linux platform). You can view these options on the installed Agent version by running with the `--usage` command line option.

Agent Configuration Options

`accesskeyid=`*string*

Set access key ID, required in most cases

`anonymous-ip-secret-key=`*string*

Set anonymous IP secret key. Default is to use `secretaccesskey` when generating anonymous IP addresses

`bypass-egress-proxy-for-upstreams`*[=true|false]* [EXPERIMENTAL]

Exclude all upstream traffic from using the egress proxy

Default: "false"

`cleaner-interval=`*time-duration*

How often to run cleanup routine

Default: "10s"

`client-ip-header=`*string*

Specify the request header containing the client IP address

Default: "X-Forwarded-For"

Specify the configuration file
Default: "/etc/sigsci/agent.conf"

`context-expiration`=*time-duration*

How long to keep request context to match with response before cleanup
Default: "10s"

`custom-request-headers`=*string* [EXPERIMENTAL]

Add custom headers to the RPC response, which will be added to the HTTP request by the module [format is CSV if name:val pairs with $AgentResponse, $RequestID, $TagList dynamic values]

`debug-log-all-the-things`*[=true|false]* [EXPERIMENTAL]

Log all the things
Default: "false"

`debug-log-blocked-requests`*[=true|false]* [EXPERIMENTAL]

Log when a request is blocked
Default: "false"

`debug-log-config-updates`=*integer* [EXPERIMENTAL]

Log when config updated or checked, 0=off, 1=updated, 2=more details
Default: "0"

`debug-log-connection-errors`=*integer* [EXPERIMENTAL]

Log when connections are dropped due an error. 0=off,1=on
Default: "0"

`debug-log-engine-errors`=*integer* [EXPERIMENTAL]

Log WAF engine errors: 0=off, 1=on, 2=verbose
Default: "1"

`debug-log-proxy-requests`*[=true|false]* [EXPERIMENTAL]

Generates debug output of proxied requests
Default: "false"

`debug-log-rpc-data`=*string* [EXPERIMENTAL]

Log (hexdump) raw RPC data to the given file

`debug-log-uploads`=*integer* [EXPERIMENTAL]

Log what is being sent to Signal Sciences: 0=off, 1=json, 2=json-pretty
Default: "0"

`debug-log-web-inputs`=*integer* [EXPERIMENTAL]

Log web inputs coming from the module: 0=off, 1=json, 2=json-pretty
Default: "0"

`debug-log-web-outputs`=*integer* [EXPERIMENTAL]

Log web outputs going back to the module: 0=off,1=json,2=json-pretty
Default: "0"

`debug-standalone`=*integer* [EXPERIMENTAL]

Bitfield: 0=normal, 1=no upload, 2=no download, 3=no networking, 4=use empty rules, 7=no net+empty rules
Default: "0"

`download-cdn-url`=*string* [EXPERIMENTAL]

CDN URL to check and download new configurations before checking download-url, empty string disables CDN fetch
Default: "https://wafconf.signalsciences.net"

`download-config-cache`=*string*

Filename to cache latest downloaded config (if relative, then base it on shared-cache-dir)

`download-config-version`=*integer* [EXPERIMENTAL]

Force the downloader to download a specific config version: 0=auto versioning
Default: "0"

`download-failover-url`=*string* [EXPERIMENTAL]

URL to check and download new configurations if download-url is not available
Default: "https://sigsci-agent-wafconf-us-west-2.s3.amazonaws.com"

`download-interval`=*time-duration* [EXPERIMENTAL]

How often to check for a new configuration
Default: "30s"

URL to check and download new configurations

Default: "https://sigsci-agent-wafconf.s3.amazonaws.com"

`envoy-expect-response-data`=*integer* [EXPERIMENTAL]

Expect response data from envoy: 0=response data is not expected and some dependent product features will not be available, 1=agent will wait for response data via http_grpc_access_log gRPC API

Default: "0"

`envoy-grpc-address`=*string* [EXPERIMENTAL]

Envoy gRPC address to listen on (unix domain socket path or host:port)

`envoy-grpc-cert`=*string* [EXPERIMENTAL]

Envoy gRPC optional TLS cert file (PEM format)

`envoy-grpc-key`=*string* [EXPERIMENTAL]

Envoy gRPC optional TLS key file (PEM format)

`haproxy-spoa-address`=*string* [EXPERIMENTAL]

Haproxy SPOA address to listen on (unix domain socket path or host:port)

Default: "unix:/var/run/sigsci-ha.sock"

`--help` (commandline only option)

Dump basic help text

`inspection-alt-response-codes`=*csv-integer* [DEPRECATED]

DO NOT USE: the alternative response code concept is deprecated - all codes 300-599 are now considered blocking codes and this option will be removed

`inspection-anomaly-duration`=*time-duration* [EXPERIMENTAL]

Envoy/revproxy global duration after which the request will be considered an anomaly and the response will be inspected even if nothing else was found in the request during inspection

Default: "1s"

`inspection-anomaly-size`=*integer* [EXPERIMENTAL]

Envoy/revproxy global response size limit which the request will be considered an anomaly and the response will be inspected even if nothing else was found in the request during inspection

Default: "524288"

`inspection-debug`*[=true|false]* [EXPERIMENTAL]

Envoy/revproxy global enable/disable inspection debug logging

Default: "false"

`inspection-max-content-length`=*integer* [EXPERIMENTAL]

Envoy/revproxy global max request content length that is allowed to be inspected

Default: "307200"

`inspection-timeout`=*time-duration* [EXPERIMENTAL]

Envoy/revproxy global inspection timeout after which the system will fail open

Default: "100ms"

`jaeger-tracing`*[=true|false]* [EXPERIMENTAL]

Enables jaeger tracing - configured with `JAEGER\_\*` environment variables (currently for envoy only)

Default: "false"

`--legal` (commandline only option)

Show legal information and exit

`local-networks`=*string*

Set local networks for determining the real client IP (CSV of CIDR, 'all', 'none', or 'private'). These are the networks trusted to set the client IP header.

Default: "all"

`log-out`=*string*

Log output location, 'stderr', 'stdout', or file name (NOTE: on Windows, important logs will be sent to the eventlog)

`max-backlog`=*integer*

Maximum RPC requests in queue (by default scaled with rpc-workers)

Default: "0"

`max-connections`=*integer*

Maximum in-flight RPC connections (by default scaled with rpc-workers)

Default: "0"

Reverse proxy only - maximum in-flight transactions that the engine can be inspecting, 0=unlimited
Default: "0"

`max-logs`=*integer*
Maximum number of log lines held while waiting to send upstream
Default: "1000"

`max-procs`=*string*
Maximum number or percentage of CPUs (cores) to use e.g max-procs=4 or max-procs="100%".

`max-records`=*integer*
Maximum number of records held while waiting to send (by default scaled with rpc-workers)
Default: "0"

`reverse-proxy`*[=true|false]* [DEPRECATED]
Enable the reverse proxy, which requires setting a listener and upstream
Default: "false"

`reverse-proxy-accesslog`=*string* [DEPRECATED]
Reverse proxy access log filename

`reverse-proxy-conn-idle-max`=*integer* [DEPRECATED]
Reverse proxy max idle connections
Default: "100"

`reverse-proxy-conn-idle-timeout`=*time-duration* [DEPRECATED]
Reverse proxy idle connection timeout
Default: "1m30s"

`reverse-proxy-conn-keepalive`=*time-duration* [DEPRECATED]
Reverse proxy connection TCP keepalive interval
Default: "30s"

`reverse-proxy-conn-timeout`=*time-duration* [DEPRECATED]
Reverse proxy connection (TCP handshake) timeout
Default: "30s"

`reverse-proxy-expect-continue-timeout`=*time-duration* [DEPRECATED]
Reverse proxy timeout waiting for continue after expect
Default: "1s"

`reverse-proxy-idle-timeout`=*time-duration* [DEPRECATED]
Reverse proxy idle timeout
Default: "0s"

`reverse-proxy-listener`=*string* [DEPRECATED]
Reverse proxy listener address:port

`reverse-proxy-pass-host-header`*[=true|false]* [DEPRECATED]
Pass the client supplied host header through to the upstream (including the upstream TLS handshake for use with SNI and certificate validation)
Default: "true"

`reverse-proxy-read-timeout`=*time-duration* [DEPRECATED]
Reverse proxy read timeout
Default: "0s"

`reverse-proxy-shutdown-timeout`=*time-duration* [DEPRECATED]
Reverse proxy shutdown timeout for transactions to complete
Default: "30s"

`reverse-proxy-tls`*[=true|false]* [DEPRECATED]
Enable the TLS reverse proxy, which requires setting a listener and upstream
Default: "false"

`reverse-proxy-tls-cert`=*string* [DEPRECATED]
Reverse proxy TLS certificate file (PEM format)

`reverse-proxy-tls-cipher-suites`=*csv-string* [DEPRECATED]
Reverse proxy TLS listener cipher suites [use --show-tls-cipher-suites for a list]
Default: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA"

Reverse proxy TLS handshake timeout

Default: “10s”

`reverse-proxy-tls-insecure-skip-verify`*[=true|false]* [DEPRECATED]

Insecurely skip reverse proxy upstream TLS verification

Default: “false”

`reverse-proxy-tls-key=`*string* [DEPRECATED]

Reverse proxy TLS private key file (PEM format)

`reverse-proxy-tls-listener=`*string* [DEPRECATED]

Reverse proxy TLS listener address:port

`reverse-proxy-tls-min-version=`*string* [DEPRECATED]

Reverse proxy TLS listener min version

Default: “1.0”

`reverse-proxy-tls-upstream=`*csv-string* [DEPRECATED]

Reverse proxy TLS upstream, comma separated address:port[:scheme] with default scheme=https

`reverse-proxy-trust-proxy-headers`*[=true|false]* [DEPRECATED]

Trust the incoming proxy (X-Forwarded-For*) header values

Default: “true”

`reverse-proxy-upstream=`*csv-string* [DEPRECATED]

Reverse proxy upstream, comma separated address:port

`reverse-proxy-write-timeout=`*time-duration* [DEPRECATED]

Reverse proxy write timeout

Default: “0s”

`revproxy-reload-on-update`*[=true|false]* [EXPERIMENTAL]

Reload the reverse proxy service config on agent config updates to support dynamic reconfiguration (only functions on OSes that support zero downtime restarts such as Linux >= 3.9 kernel)

Default: “false”

`rpc-address=`*string*

RPC address to listen on and serve modules from

Default: “unix:/var/run/sigsci.sock”

`rpc-version=`*integer* [DEPRECATED]

RPC protocol version

Default: “0”

`rpc-workers=`*integer* [EXPERIMENTAL]

RPC workers to use. If unset, then the `max-procs` value will be used

Default: “0”

`sample-percent=`*integer*

Sample input, 100=process everything, 0=ignore everything

Default: “100”

`secretaccesskey=`*string*

Set secretaccesskey, required along with accesskeyid in most cases

`server-flavor=`*string* [EXPERIMENTAL]

Server-flavor, allow distinguishing this revproxy install as a buildpack or other flavor.

`server-hostname=`*string*

Server hostname, default is to ask OS

`service-shutdown-timeout=`*time-duration*

Timeout waiting for pending transactions to complete during service shutdown

Default: “2s”

`shared-cache-dir=`*string* [EXPERIMENTAL]

Base directory for any cache files

Default: "/tmp/sigsci-agent.cache"

`--show-tls-cipher-suites` (commandline only option)

Show available TLS cipher suites and exit

Set the statsd address to send metrics to (e.g., `hostname:port` or `unix:///path/socket`)

`statsd-metrics=`*csv-string* [EXPERIMENTAL]

Set the statsd metrics filter (glob patterns allowed - assumed prefix if no patterns used)
Default: ""*

`statsd-type=`*string* [EXPERIMENTAL]

Set the statsd server type to enable advanced features (e.g., `statsd` or `dogstatsd`)
Default: “statsd”

`upload-log=`*string* [EXPERIMENTAL]

Log filename to write agent event data

`upload-log-header-map`*[=true|false]* [EXPERIMENTAL]

HTTP request,response header data in map format
Default: “false”

`upload-syslog`*[=true|false]* [EXPERIMENTAL]

Write agent event data to syslog
Default: “false”

`upload-url=`*string* [EXPERIMENTAL]

URL to upload agent data
Default: “https://c.signalsciences.net/0/push”

`--usage` (commandline only option)

Dump full usage text

`validate-config`*[=true|false]*

Validate the config entries provided to warn against potentially invalid entries
Default: “true”

`--version` (commandline only option)

Show version information and exit

`waf-data-log=`*string* [EXPERIMENTAL]

Filename to log WAF inspection data (currently JSON format). Using “eventlog” on Windows will send these to the eventlog. Requests are logged to the provided location if they have at least one signal added by default inspectors or if they have at least one signal added by a request rule with a Request logging value of Sampled.

`windows-eventlog-level=`*integer* [EXPERIMENTAL]

Set the windows eventlog level (use names that will be converted to integers: `debug`, `info`, `warning`, `error`, or `none`).
Default: “3”

Block Based Options

The following block based options are only available as such in a configuration file. In the configuration file, they must be after all other regular options in the file.

As an alternative to a configuration file these can be configured from a command-line option or environment variable in the following format:

--option=‘name1:{opt=val,…};name2:{opt=val,…}’
OR
SIGSCI_OPTION=‘name1:{opt=val,…};name2:{opt=val,…}’

`[revproxy-listener.NAME]`

Define named reverse proxy listener(s) with options (block or revproxy-listener=“name1:{opt=val,…};name2:{opt=val,…};…”)

revproxy-listener options:

`access-log=`*string*

Access log filename

`close-conn-on-request-smuggling`*[=true|false]* [DEPRECATED]

‘Connection: close’ header will be added to requests that appear to be HTTP Request Smuggling attacks
Default: “false”

Max idle connections in the upstream connection pool (0 will disable connection pooling)

Default: "100"

`conn-idle-timeout`=*time-duration*

Idle connection timeout for the upstream connection pool

Default: "1m30s"

`conn-keepalive`=*time-duration*

Connection keepalive interval for upstream connections

Default: "30s"

`conn-max-per-host`=*integer*

Maximum total number of upstream connections in any state per host (0 is unlimited). Connections over the limit will block until more are available

Default: "0"

`conn-timeout`=*time-duration*

Connection timeout for upstream connections

Default: "30s"

`enabled`*[=true|false]*

Enable/disable the reverse proxy listener

Default: "true"

`expect-continue-timeout`=*time-duration*

Timeout waiting for 'continue' after 'expect' for upstream traffic

Default: "1s"

`expose-raw-headers`*[=true|false]* [DEPRECATED]

This experimental option replaces 'close-conn-on-request-smuggling' functionality. The option will need to be enabled per each reverse proxy listener.

Default: "true"

`http2`*[=true|false]*

Enable HTTP/2 support for the listener

Default: "true"

`http2-upstreams`*[=true|false]*

Prefer HTTP/2 for the upstreams

Default: "true"

`idle-timeout`=*time-duration*

Network idle timeout for the listener

Default: "0s"

`inspect-websocket`*[=true|false]*

Enable/disable websocket inspection

Default: "false"

`inspection-alt-response-codes`=*csv-integer* [DEPRECATED]

DO NOT USE: the alternative response code concept is deprecated - all codes 300-599 are now considered blocking codes and this option will be removed

`inspection-anomaly-duration`=*time-duration*

Duration after which the request will be considered an anomaly and the response will be inspected even if nothing else was found in the request during inspection

Default: "1s"

`inspection-anomaly-size`=*integer*

Response size limit which the request will be considered an anomaly and the response will be inspected even if nothing else was found in the request during inspection

Default: "524288"

`inspection-debug`*[=true|false]*

Enable/disable inspection debug logging

Default: "false"

`inspection-max-content-length`=*integer*

Max request content length that is allowed to be inspected

Default: "307200"

`inspection-timeout`=*time-duration*

Inspection timeout after which the system will fail open

Default: "100ms"

Listener URL [scheme://address:port]

`log-all-errors`*[=true|false]*

Log all errors, not just common

Default: “false”

`minimal-header-rewriting`*[=true|false]*

Minimal header rewriting. If enabled, then only hop-by-hop headers will be removed as required by RFC-2616 sec 13.5.1. No proxy headers will be added/modified, though they will be passed through if trust-proxy-headers is set

Default: “false”

`pass-host-header`*[=true|false]*

Pass the client supplied host header through to the upstream (including the upstream TLS handshake for use with SNI and certificate validation)

Default: “true”

`read-timeout`*=time-duration*

Network read timeout for the listener

Default: “0s”

`remove-hop-header`*[=true|false]*

Unused hop headers will be removed from forwarded requests

Default: “true”

`request-timeout`*=time-duration*

Overall request timeout (will enable buffering, which may cause issues with streaming services)

Default: “0s”

`response-flush-interval`*=time-duration*

Interval to flush any buffered/streaming response data (0 disables forced flushes; -1 forces flushes after every write; interval values force flushes on a fixed time interval)

Default: “0s”

`response-header-timeout`*=time-duration*

Response header timeout waiting for upstream responses

Default: “0s”

`shutdown-timeout`*=time-duration*

Timeout waiting for pending transactions to complete during server shutdown

Default: “30s”

`tls-ca-roots`*=string*

TLS trusted certificate authority certificates file (PEM format)

`tls-cert`*=string*

TLS certificate file (PEM format)

`tls-cipher-suites`*=csv-string*

TLS listener cipher suites [use --show-tls-cipher-suites for a list]

Default: “TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA”

`tls-handshake-timeout`*=time-duration*

TLS handshake timeout for upstream connections

Default: “10s”

`tls-insecure-skip-verify`*[=true|false]*

Insecurely skip upstream TLS verification (for self signed certs, etc.)

Default: “false”

`tls-key`*=string*

TLS private key file (PEM format)

`tls-key-passphrase`*=string*

TLS private key passphrase in the format `type:data`, where type is one of: `pass` or `file` (EX: `pass:mypassword` or `file:/etc/secrets/tls-key-passphrase`)

`tls-min-version`*=string*

TLS listener min version

Default: “1.0”

`tls-verify-servername`*=string*

Force the servername used in upstream TLS verification; consider using pass-host-header first, but this may be required if neither the hostname used by the downstream client nor the hostname/ip used in the upstream URL is listed in the upstream TLS certificate

`trust-proxy-headers`*[=true|false]*

Trust the incoming proxy (X-Forwarded-For*) header values. If not trusted, then incoming proxy headers are removed before any additions

Default: "true"

`upstreams`=*csv-string*

Upstream, comma separated upstream URLs [scheme://address:port]

`write-timeout`=*time-duration*

Network write timeout for the listener

Default: "0s"

System Environment Options

These system level environment variable based options will also affect processing.

Environment Variables

`HTTP_PROXY` or `http_proxy`=*url* [DEPRECATED]

Proxy outbound HTTP requests through the proxy at the defined URL

`HTTPS_PROXY` or `https_proxy`=*url*

Proxy outbound HTTPS requests through the proxy at the defined URL (takes precedence over HTTP_PROXY for HTTPS requests)

`NO_PROXY` or `no_proxy`=*csv-url*

Comma separated list of URLs NOT to proxy or '*' for all URLs

The options are generally available in three forms, overridden in the following order:

1. In the configuration file (default: `/etc/sigsci/agent.conf`)
2. On the command line, prefixed with a double dash (--) (e.g., `--help`)
3. As an environment variable, all capitalized, prefixed with `SIGSCI_` and dashes changed to underscores (_) (e.g., the `max-procs` option would become the `SIGSCI_MAX_PROCS` environment variable)

There are a few exceptions:

- Informational options such as `--help`, `--legal`, and `--version` only make sense as command line options and are noted above.
- As of agent **v4.22.0**, the `HTTP_PROXY` environment variable is deprecated and will no longer be honored for https connections, HTTPS_PROXY must be used.
- The agent will honor the system `HTTPS_PROXY` environment variable allowing configuration of an egress HTTPS proxy URL for those sites where outbound access must be through a proxy (e.g., `HTTPS_PROXY=http://10.0.0.1:8080`).

Configuring a HTTPS Proxy for the Agent

If the system the agent is running on does not have direct internet access, it may need to be configured to access the internet via a HTTPS proxy. To do this, one or more of the `HTTPS_PROXY`, or `NO_PROXY` system environment variables will need to be configured. While on some systems this may be set system wide, it may be desireable to use the proxy for only the Signal Sciences agent.

Linux Package Based Systems (deb, rpm, etc)

On Linux and similar systems, the sigsci-agent service (systemd, upstart, init.d, etc.) will source in the `/etc/default/sigsci-agent` file containing `var=value` pairs. To set the proxy for the agent, add the environment variable configurations into this file, one per line.

For example, to use the HTTPS proxy at `10.0.0.1` on port 8080 add the following to `/etc/default/sigsci-agent`:

```
HTTPS_PROXY=http://10.0.0.1:8080
```

The `sigsci-agent` service will then need to be restarted.

Windows Based Systems

On Windows based system where the agent is run as a service, the environment variables can be set system wide, however this may require a system reboot for the services to see the change.

If the change only needs to be set for the Signal Sciences agent, then set the following registry entry to update the environment settings for only the `sigsci-agent` service:

- Add a Multi-String Value (REG_MULTI_SZ) registry entry if it does not already exist:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sigsci-agent\Environment
```

```
HTTPS_PROXY=http://10.0.0.1:8080
```

- The `sigsci-agent` service will then need to be restarted

This can be done manually using the `regedit.exe` or similar utility, or via the commandline with something like the following, replacing the URLs with the correct proxy URLs:

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sigsci-agent /v Environment /t REG_MULTI_SZ /d "HTTP
```

If more than one variable needs to be set, then separate each `var=value` with a NULL (`\0`) character in the above command, such as `"HTTPS_PROXY=http://10.0.0.1:8080\0NO_PROXY=http://localhost"`

Reverse Proxy Configuration

The agent may be configured to run as a reverse proxy. To learn more, see the reverse proxy configuration documentation.

> **Note:** Updating the `sigsci-agent` will remove all environment settings from the registry, therefore if one wishes to preserve any settings, consider downloading the `sigsci-agent_latest.zip` from Windows Agent Installation and replacing the executables.

Next Steps

Explore our module options and install the Signal Sciences module.

Module Configuration

We provide the ability to configure the Signal Sciences module. The following attributes are set by default, but may need to be modified to provide support for different environments. In the majority of cases modifying module configuration is not necessary. **Contact support if you need assistance or have questions regarding modifying module configuration**.

Apache

To modify the Signal Sciences module configuration in Apache you will need to add directives to your Apache configuration file (e.g., for CentOS it is httpd.conf, for Debian or Ubuntu it is apache.conf or apache2.conf). Note, these directives must be set after the Signal Sciences module is loaded.

Starting with release 1.6.0, the following directives replace any earlier ones. These directives are a renaming of the earlier ones but with the addition of the prefix `SigSci`.

Name	Description
`SigSciAgentTimeout`	Agent socket timeout (in milliseconds), default: `100`.
`SigSciAgentPostLen`	Maximum POST body site in bytes, default: `100000`
`SigSciAgentInspection`	Enable or disable the module, default: `On`
`SigSciAgentPort`	The local port (when using TCP) that the agent listens on, default: none. Note, if AgentPort is set then `AgentHost` must be a IP or hostname.
`SigSciAgentHost`	Host or IP Address, otherwise use `AgentHost` to specify the domain socket file. `/foo/bar.sock`
`SigSciEnableFixups`	Fixups is the phase in request processing after authorization but before the content handler. This setting toggles Signal Sciences fixups priority over post read request handling to allow the request to be seen before it's modified. (`On` or `Off`) - default is `Off`
`SigSciRunBeforeModulesList`	Signal Sciences module runs before the list of specified modules. Example: `mod_example.c mod_something.c`
`SigSciRunAfterModulesList`	Signal Sciences module runs after the list of specified modules. Example: `mod_example.c mod_something.c`

The following directives will be deprecated in favor of the new ones above with the `SigSci` prefix but are backwards compatible and will continue to work.

Name	Description
`AgentTimeout`	Agent socket timeout (in milliseconds), default: `100`.
`AgentPostLen`	Maximum POST body site in bytes, default: `100000`
`AgentInspection`	Enable or disable the module, default: `On`

	be a IP or hostname.	
`AgentHost`	Host or IP Address, otherwise use `AgentHost` to specify the domain socket file.	`/foo/bar.sock`

The following directives are deprecated and will be ignored.

Name	Description
`SigSciAltResponseCodes`	Specifying alternative codes on which to block is deprecated. Instead we now block on any response code within the range 300-599.

NGINX C Binary Module

To modify the Signal Sciences NGINX module configuration, you will need to add directives to the NGINX configuration file, located by default at `/etc/nginx/nginx.conf`.

In the global section, for example after the `pid /run/nginx.pid;` line:

```
load_module /etc/nginx/modules/ngx_http_sigsci_module.so;
```

For the NGINX.org package (`nxo`) only, add the following line:

```
load_module /etc/nginx/modules/ndk_http_module.so;
```

> **Note:** For the NGINX Plus package, there is no `load_module ndk_http_module.so` config required. The `ndk` module should be installed by the package `nginx-plus-module-ndk`.

Name	Description	Values	Default Value	Section
`sigsci_enabled`	Enable or disable the module	`on`, `off`	`on`	http, server or per location
`sigsci_debug`	Enable `sigsci_debug` only, doesn't affect other modules	`on`, `off`	`off`	http
`sigsci_handler_phase`	Phase in which the module processes request	`preaccess`, `access`, `precontent`, `rewrite`	`rewrite`	http
`sigsci_agent_max_post_len`	Maximum POST body size in bytes to be sent to agent	0 => don't send post body; else number bytes > 0	`100000`	http
`sigsci_agent_timeout`	Agent communication socket timeout in milliseconds	Milliseconds > 0	`100`	http
`sigsci_anomaly_resp_size`	Maximum response size in bytes. Larger than this is considered anomalous.	Bytes > 0	`524288`	http
`sigsci_anomaly_resp_time`	Maximum response time in milliseconds. Larger than this is considered anomalous.	Milliseconds > 0	`1000`	http
`sigsci_agent_host`	The IP address or a path to Unix domain socket the SignalSciences Agent listens on	Example: `tcp:localhost`	`unix:/var/run/sigsci.sock`	http
`sigsci_agent_port`	The TCP port that the agent listens on. Note: use only when `sigsci_agent_host` set to be an IP or hostname.	valid TCP port number	none	http
`sigsci_websocket_enabled`	Enable or disable WebSocket inspection	`on`, `off`	`off`	http, server or per location

> **Note:** `sigsci_websocket_enabled` is `off` by default. To enable it, it must be specified in the `http` section. Thereafter, it may be turned `off` and `on` in the `server` and `location` sections as needed.

```
# sigsci module settings
##
sigsci_debug          on;
sigsci_agent_timeout  200;
```

These examples show using `location` sections with the `sigsci_enabled` parameter:

```
# sigsci_enabled set to "on"
location /inspect/ {
    sigsci_enabled  on;
    proxy_pass      http://127.0.0.1:80/inspect/;
}

# sigsci_enabled set to "off"
location /noinspect/ {
    sigsci_enabled  off;
    proxy_pass      http://127.0.0.1:80/noinspect/;
}
```

Detailed example using `server` and `location` sections for the `sigsci_websocket_enabled` parameter:

```
http {

  # must be turned on in global section
  sigsci_websocket_enabled on;

  server {
      ...
      # turned off for this server section
      sigsci_websocket_enabled off;

      # websocket turned on for this location
      location /websenabled {
          sigsci_websocket_enabled on;
          proxy_pass http://websocket;
          ...
      }

      # websocket off for this location since it is off in server
      location /websdisabled {
          proxy_pass http://websocket;
          ...
      }
```

NGINX Lua Module

To modify the Signal Sciences Lua module for NGINX, changes can be made in the Signal Sciences Lua script, which by default is at `/opt/sigsci/nginx/sigsci.conf`.

Name	Description
`agenthost`	The IP address or path to Unix domain socket the SignalSciences Agent is listening on, default: `unix:/var/run/sigsci.sock`.
`agentport`	The local port (when using TCP) that the agent listens on, default: `12345`
`timeout`	Agent socket timeout (in milliseconds), default: `100`.
`maxpost`	Maximum POST body site in bytes, default: `100000`

Example configuration

```
sigsci.agenthost = "unix:/var/run/sigsci.sock"
sigsci.agentport = 12345
```

HAProxy

Configuration changes are typically not required for the HAProxy module to work. However, it is possible to override the default settings if needed. To do so, you must create an `override.lua` file in which to add these configuration directives. Then, update the `global` section of your HAProxy config file (`/usr/local/etc/haproxy/haproxy.cfg`) to load this over-ride config file.

Example of configuration

```
global
   ...
   lua-load /path/to/override.lua
   ...
```

Over-ride Directives

These directives may be used in your over-ride config file.

Name	Description
`sigsci.agenthost`	The IP address or path to unix domain socket the SignalSciences Agent is listening on, default: `/var/run/sigsci.sock` (unix domain socket).
`sigsci.agentport`	The local port (when using TCP) that the agent listens on, default: `nil`
`sigsci.timeout`	Agent socket timeout (in seconds), default: `1` (`0` means off).
`sigsci.maxpost`	Maximum POST body site in bytes, default: `100000`
`sigsci.extra\_blocking\_resp\_hdr`	User may supply a response header to be added upon 406 responses, default: ""

Example of over-ride configuration

```
sigsci.agenthost = "192.0.2.243"
sigsci.agentport = 9090
sigsci.extra_blocking_resp_hdr = "Access-Control-Allow-Origin: https://example.com"
```

IIS

Configuration changes are typically not necessary. By default, the module will use port 737 to communicate with the agent (or in v2.0.0+, if the agent was configured to use an alternate port, it will use that port). The configuration can be set via the MSI installer, the new `SigsciCtl.exe` utility in v2.0.0+, IIS Manager UI, via PowerShell, or using the `appcmd.exe` utility.

> **NOTE:** Ensure that the same port number is used by the both the module and the agent configurations.

Using the MSI

To set a configuration option when installing the MSI, specify the option on the command line in `option=value` format. For example:

```
msiexec /qn /i sigsci-module-iis_latest.msi agentHost=203.0.113.182 agentPort=737
```

Using SigsciCtl.exe

To set a configuration option via `SigsciCtl.exe` utility after install, use the `Configure-Module` command. For example:

```
"%PROGRAMFILES%\Signal Sciences\IIS Module\SigsciCtl.exe" Configure-Module agentHost=203.0.113.182 agentPort=737
```

To view the active configuration via the `SigsciCtl.exe` utility the `Get-Configs` command:

```
"%PROGRAMFILES%\Signal Sciences\IIS Module\SigsciCtl.exe" Get-Configs
```

This should output something similar to the following:

```
C:\WINDOWS\system32\inetsrv\config\schema:

Date                          Size Name
-------------------- ------------ --------------------------------
2020-02-13 03:12:56Z           677 SignalSciences_schema.xml

"SignalSciences" Configuration Section (Global):

                        Attribute Value
-------------------------------- ----------------------------------------------------------------
```

```
                       Debug False
              ReuseConnections False
                   MaxPostSize 100000
                   AnomalySize 524288
         AnomalyDurationMillis 1000
                 TimeoutMillis 200
```

Using PowerShell

To set a configuration option via PowerShell (modern Windows only) use the `-SectionPath "SignalSciences"` option such as follows:

```
Set-IISConfigAttributeValue -ConfigElement (Get-IISConfigSection -SectionPath "SignalSciences") -AttributeName "a
```

To list the configuration using PowerShell, run the following:

```
(Get-IISConfigSection -SectionPath "SignalSciences").RawAttributes
```

To reset the configuration to defaults using PowerShell, run the following:

```
Clear-WebConfiguration -Filter SignalSciences -PSPath 'IIS:\'
```

Using the appcmd.exe

To set a configuration option via the `appcmd.exe` command line tool use the `-section:SignalSciences` option. For example:

```
"%SYSTEMROOT%\system32\inetsrv\appcmd.exe" set config -section:SignalSciences -agentPort:737
```

To list the configuration using `appcmd.exe`, run the following. Default values will not be shown:

```
"%SYSTEMROOT%\system32\inetsrv\appcmd.exe" list config -section:SignalSciences
```

To reset the configuration to defaults using `appcmd.exe`, run the following:

```
"%SYSTEMROOT%\system32\inetsrv\appcmd.exe" clear config -section:SignalSciences
```

Language Modules

See language specific module pages for configuration details.

- Java
  - As a Servlet filter
  - As a Jetty handler
  - As a Netty handler
  - With Dropwizard
  - On WebLogic servers
- Node.js
- .NET

Compatibility & Requirements

> **NOTE:** Per our agent end-of-support policy, we support agent versions that are under two years old, and on a quarterly cadence, we deprecate and no longer support agent versions that are older than two years.

Processors

We support the following processors:

- **Intel.** All agent and module versions are compatible with Intel processors.

- **AMD.** All agent and module versions are compatible with AMD processors.

We do not currently provide ARM agent packages for Amazon Linux or Windows.

You can run the agent on ARM processors with the NGINX C Binary module v1.18.0+ on the following Linux distributions:

- Alpine (3.13 - 3.17)
- CentOS/RHEL (EL7 - EL9)

Alternatively, you can run the agent on ARM processors without a module in reverse proxy mode on the Linux distributions mentioned above.

Linux

The Signal Sciences Agent and Modules are supported on the following Linux distributions:

Distribution	Code Name	Version
Alpine		3.11
		3.12
		3.13
		3.14
		3.15
		3.16
		3.17
Amazon Linux		2
CentOS	Enterprise Linux 6	6.x
	Enterprise Linux 7	7.x
	Enterprise Linux 8	8.x
	Enterprise Linux 9	9.x
Debian	Wheezy	7.x
	Jessie	8.x
	Stretch	9.x
	Buster	10.x
Red Hat	Enterprise Linux 6	6.x
	Enterprise Linux 7	7.x
	Enterprise Linux 8	8.x
	Enterprise Linux 9	9.x
Ubuntu	Precise	12.04 LTS
	Trusty	14.04 LTS
	Xenial	16.04 LTS
	Bionic	18.04 LTS
	Focal	20.04 LTS
	Jammy	22.04 LTS

Only 64-bit environments are supported. If you need 32-bit support contact us.

Signal Sciences Module

The Signal Sciences Module is a lightweight module that integrates with your web server software or application and is the interface between incoming requests and our agent process. We support NGINX, Apache, and IIS web servers, the HAProxy proxy server, and several application languages (including .NET, Golang, Java, Node.js). Specific details for some of the more commonly deployed platform are listed below:

NGINX Web Servers

The NGINX modules provided by Signal Sciences are built specifically for the NGINX.org distributions of NGINX and may not be compatible with a custom build of NGINX. If switching to an NGINX.org distribution is not an option, reach out to our support team or your Signal Sciences account team for assistance.

The NGINX module is offered in two different variations, depending on the platform and what best meets your needs. We currently support:

C Binary

The NGINX Module is available in a variation built as a C binary, which requires no dependencies. Versions of NGINX.org supported by the C binary are:

- 1.23.0 - 1.23.1
- 1.22.0
- 1.21.0 - 1.21.6

- 1.17.9
- 1.17.8
- 1.17.7
- 1.17.6
- 1.17.5
- 1.17.4
- 1.17.3
- 1.17.2
- 1.17.1
- 1.17.0
- 1.16.1
- 1.16.0
- 1.15.12
- 1.15.10
- 1.15.9
- 1.15.8
- 1.15.7
- 1.15.3
- 1.14.1
- 1.12.2
- 1.10.3 (on Ubuntu 16.04 only)

Versions of NGINX Plus supported by the C binary are:

- 27-1 (1.21.6)
- 26-1 (1.21.5)
- 25-1 (1.21.3)
- 24-1 (1.19.10)
- 23-1 (1.19.5)
- 22-1 (1.19.0)
- 21-1 (1.17.9)
- 20-1 (1.17.6)
- 19-1 (1.17.3)
- 18-1 (1.15.10)
- 17-1 (1.15.7)

These C binary versions are kept up-to-date with stable releases and on demand for mainline releases.

Lua

Alternatively, a variation of the NGINX Module as Lua is available, which requires NGINX to be built with Lua and for LuaJIT support.

This version is written in Lua and requires your NGINX binary to be compiled with the third party `ngx_lua` module enabled. We also require the `ngx_lua` module be linked against the LuaJIT just-in-time byte code library for performance.

NGINX deployments vary from organization to organization, and we support two approaches to this installation:

- **Pre-built binary packages** - for all the OS platforms we support we provide three flavors or pre-built NGINX packages that are built with the required `ngx_lua` module.
- **Source builds** - for those organizations building NGINX internally from source, we have published our reference build guidelines that can be used to review and adapt for your own build process.

If you currently use a pre-built binary package of NGINX, either from the operating system's package collection or from the official NGINX package repositories let us know, and we can provide a suitable replacement package built with our required supporting modules. Contact us for more information.

The Lua variation of the NGINX module is supported on the following versions of NGINX:

Release	Versions
1.0	1.0.15
1.1.19	1.1.19

1.4	1.4.6
1.6	1.6.0, 1.6.1, 1.6.2
1.7	1.7.2, 1.7.4, 1.7.7, 1.7.8, 1.7.9
1.8	1.8.x
1.9	1.9.x
1.10	1.10.x
1.11	1.11.x
1.12	1.12.x

Apache Web Servers

Our Apache module is distributed in binary form as an Apache shared module and supports Apache version 2.2 and 2.4.

Microsoft Windows Servers

- IIS 7 or higher, Windows Server 2008R2 (Windows 7) or higher (64-bit)
- .NET 4.5 or higher

We currently only support 64-bit and 32-bit application pools on Windows 2012 or higher. We only support 64-bit application pools on Windows Server 2008R2.

Additionally, we only support 64-bit OSes. For older or 32-bit versions of Windows, it is possible to deploy the Signal Sciences Agent as a reverse proxy. If you have questions or require assistance with older or 32-bit versions of Windows, reach out to our support team.

HAProxy Servers

HAProxy module. Our HAProxy module is written in Lua and requires your HAProxy binary to be compiled with the `lua` module enabled. The HAProxy module requires HAProxy 1.8 or higher.

> **Note:** Although supported, there is a known issue with HAProxy 1.8 that may result in performance issues when the Signal Sciences module is installed. HAProxy has fixed this issue with HAProxy 2.2, but the fix will not be backported to 1.8. It is recommended to upgrade to HAProxy 2.2 or higher if possible, or use an alternate Signal Sciences deployment method (e.g., reverse proxy agent if HAProxy 1.8 must be used).

HAProxy SPOE module. Our HAProxy SPOE module does not require Lua. The HAProxy SPOE module requires HAProxy 1.8 or higher.

Node.js

0.10 or higher

Java

- Java 1.8 or newer
- Spring version 2.x
- Spring Boot Tomcat Starter 2.x
- Spring Boot Starter WebFlux 2.x
- Tomcat 8

Package Downloads

Agent

The Signal Sciences agent supports different combinations of operating systems and architecture types.

> **Note:** Per our agent end-of-support policy, we support agent versions that are under two years old, and on a quarterly cadence, we deprecate and no longer support agent versions that are older than two years.

Download the latest version of the agent or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The agent directory appears.

2. Click the version of the agent that you want to use.

3. Click the name of the operating system you want to use. Options include Alpine, CentOS or RHEL, Debian, Linux, Ubuntu, and Windows.

4. Click the version of the operating system that you want to use.

- `package-version`: the version of the Signal Sciences agent.
- `os-version`: the version of the operating system (OS).
- `architecture-type`: the architecture type. Types include ARM64 and AMD64.
- `file-type`: the type of file.

The elements are assembled as follows:

```
[base-name]_[package-version]~[os-version]_[architecture-type].[file-type]
```

For example, you can break down the `sigsci-agent_4.33.0~jammy_amd64.deb` package version as follows:

base-name	package-version	os-version	architecture-type	file-type
sigsci-agent	4.33.0	Jammy	AMD64	DEB

Apache

The Apache module supports different combinations of operating systems for the x86_64 (AMD64) architecture type. Download the latest version or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The Apache module directory appears.

2. Click the version of the module that you want to use.

3. Click the name of the operating system you want to use. Options include Alpine, CentOS or RHEL, Debian, and Ubuntu.

4. Click the version of the operating system that you want to use.

5. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:

   - `base-name`: the type of Signal Sciences package. The base name for every module package version is `sigsci-module-apache`.
   - `package-version`: the version of the Signal Sciences module.
   - `architecture-type`: the architecture type, which is x86_64 (AMD64) for all Apache module packages.
   - `os-version`: the version of the operating system (OS). Alpine packages do not use this element.
   - `file-type`: the type of file.

   Element delimiters in the file name depend on the OS of the package. File names are assembled as follows:

Operating system	Naming convention
Alpine	`[base-name]_[package-version]_[architecture-type].[file-type]`
CentOS or RHEL	`[base-name]-[package-version].[os-version].[architecture-type].[file-type]`
Debian	`[base-name]_[package-version]-[os-version]_[architecture-type].[file-type]`
Ubuntu	`[base-name]_[package-version]-[os-version]_[architecture-type].[file-type]`

The following table demonstrates how you can break down file names into their elements:

file-name	base-name	package-version	os-version	architecture-type	file-type
sigsci-module-apache_1.8.0_x86_64.apk	sigsci-module-apache	1.8.0		x86_64	APK
sigsci-module-apache-1.8.0.el8-1.x86_64.rpm	sigsci-module-apache	1.8.0	el8-1	x86_64	RPM

NGINX

The NGINX module supports different combinations of NGINX versions, operating systems, and architecture types. Download the latest version of the module, download a specific version for CentOS or RHEL, Debian, or Ubuntu, or download a specific version for Alpine.

NGINX for CentOS or RHEL, Debian, or Ubuntu

To download a specific version of the NGINX module for CentOS or RHEL, Debian, or Ubuntu, follow these steps:

1. Navigate to the Signal Sciences package downloads site. The NGINX module directory appears.

2. Click the version of the package that you want to use.

4. Click the version of the operating system that you want to use.

5. Click the file name of the NGINX module package version that you want to download. The file names are comprised of the following elements:

   - `base-name`: the type of Signal Sciences package. The base name for every NGINX module package version is `nginx-module-sigsci`.
   - `nginx-type`: the type of NGINX server that you're running. Types include open source NGINX (NXO) and NGINX Plus (NXP).
   - `nginx-version`: the version of the NGINX server.
   - `nginx-build-number`: the build number of the NGINX version.
   - `os-version`: the version of the operating system (OS).
   - `architecture-type`: the architecture type. Types include ARM64 (AArch64) and AMD64 (x86_64).
   - `file-type`: the type of file.

   The elements are assembled as follows:

   ```
   [base-name]-[nginx-type]_[nginx-version]-[nginx-build-number]~[os-version]_[architecture-type].[file-type]
   ```

   For example, you can break down the `nginx-module-sigsci-nxp_1.21.5-492~focal_arm64.deb` package version as follows:

base-name	nginx-type	nginx-version	nginx-build-number	os-version	architecture-type	file-type
nginx-module-sigsci	NGINX Plus	1.21.5	492	Focal	ARM64	DEB

NGINX for Alpine

To download a specific version of the NGINX module for Alpine, follow these steps:

1. Navigate to the Signal Sciences download site for Alpine. The NGINX module directory appears.

2. Click the version of Alpine that you want to use.

3. Click the **main/** directory link.

4. Click the architecture type that you use. Options include ARM64 (AArch64) and AMD64 (x86_64).

5. Click the file name of the Alpine NGINX module package version that you want to download. The file names are comprised of the following elements:

   - `base-name`: the type of Signal Sciences package. The base name for every NGINX module package version is `nginx-module-sigsci`.
   - `nginx-type`: the type of NGINX server that you're running. Types include open source NGINX (NXO) and NGINX Plus (NXP).
   - `nginx-version`: the version of the NGINX server.
   - `package-version`: the version of the Signal Sciences module package.
   - `alpine-build-number`: the build number of the Alpine version.

   The elements are assembled as follows:

   ```
   [base-name]-[nginx-type]-[nginx-version]-[package-version]-[alpine-build-number].[file-type]
   ```

   For example, you can break down the `nginx-module-sigsci-nxp-1.21.6-1.1.6-r0.apk` package version as follows:

base-name	nginx-type	nginx-version	package-version	alpine-build-number	file-type
nginx-module-sigsci	NGINX Plus	1.21.6	1.1.6	r0	APK

Heroku

Download the latest version of the Heroku module or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The Heroku module directory appears.

2. Click the version of the module that you want to use.

3. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:

   - `base-name`: the type of Signal Sciences package. The base name for every module package version is `sigsci-heroku-buildpack`.

The elements are assembled as follows:

```
[base-name]-[package-version].[file-type]
```

For example, you can break down the `sigsci-heroku-buildpack-0.2.2.tgz` package version as follows:

base-name	package-version	file-type
sigsci-heroku-buildpack	0.2.2	TGZ

IBM Cloud

Download the latest version of the IBM Cloud module or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The IBM Cloud module directory appears.
2. Click the version of the module that you want to use.
3. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:
    - `base-name`: the type of Signal Sciences package. The base name for every module package version is `sigsci-bluemix-buildpack`.
    - `package-version`: the version of the Signal Sciences module.
    - `file-type`: the type of file.

   The elements are assembled as follows:

   ```
   [base-name]-[package-version].[file-type]
   ```

   For example, you can break down the `sigsci-bluemix-buildpack-1.0.2.tgz` package version as follows:

base-name	package-version	file-type
sigsci-bluemix-buildpack	1.0.2	TGZ

Pivotal Platform & Pivotal Web Services (PWS)

Download the latest version of the Pivotal Platform and Pivotal Web Services module package or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The Pivotal Platform and Pivotal Web Services module directory appears.
2. Click the version of the module that you want to use.
3. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:
    - `base-name`: the type of Signal Sciences package. The base name for every module package version is `sigsci-cloudfoundry`.
    - `package-version`: the version of the Signal Sciences module.
    - `file-type`: the type of file.

   The elements are assembled as follows:

   ```
   [base-name]-[package-version].[file-type]
   ```

   For example, you can break down the `sigsci-cloudfoundry-0.1.4.tgz` package version as follows:

base-name	package-version	file-type
sigsci-cloudfoundry	0.1.4	TGZ

Java

Download the latest version of the Java module package or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The Java module directory appears.
2. Click the version of the module that you want to use.
3. Click the file name of the files that you want to download. Relevant files are as follows:

2.5.1.jar.

- **Javadoc JAR:** - a JAR file with a documentation static HTML site for the module. The file name contains the base name (i.e., `sigsci-module-java`), the module version, the JAR type (i.e., `javadoc`), and the file type. For example, the Javadoc JAR file name for module version 2.5.1 is `sigsci-module-java-2.5.1-javadoc.jar`.
- **Shaded JAR:** - a JAR file with the compiled Signal Sciences source code and the code for all library dependencies. The file name contains the base name (i.e., `sigsci-module-java`), the module version, the JAR type (i.e., `shaded`), and the file type. For example, the shaded JAR file name for module version 2.5.1 is `sigsci-module-java-2.5.1-shaded.jar`.
- **Sources JAR:** a JAR file with the module source code that hasn't been compiled. The file name contains the base name (i.e., `sigsci-module-java`), the module version, the JAR type (i.e., `sources`), and the file type. For example, the sources JAR file name for module version 2.5.1 is `sigsci-module-java-2.5.1-sources.jar`.

.NET

Download the latest version of the .NET module package or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The .NET module directory appears.

2. Click the version of the module that you want to use.

3. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:

   - `base-name`: the type of Signal Sciences package. The base name for every module package version is `SignalSciences.Module.DotNet`.
   - `package-version`: the version of the Signal Sciences module.
   - `file-type`: the type of file.

   The elements are assembled as follows:

   ```
   [base-name].[package-version].[file-type]
   ```

   For example, you can break down the `SignalSciences.Module.DotNet.1.6.1.nupkg` package version as follows:

`base-name`	`package-version`	`file-type`
`SignalSciences.Module.DotNet`	1.6.1	NUPKG

.NET Core

Follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The .NET Core module directory appears.

2. Click the version of the module that you want to use.

3. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:

   - `base-name`: the type of Signal Sciences package. The base name for every module package version is `SignalSciences.Module.DotNetCore`.
   - `package-version`: the version of the Signal Sciences module.
   - `file-type`: the type of file.

   The elements are assembled as follows:

   ```
   [base-name].[package-version].[file-type]
   ```

   For example, you can break down the `SignalSciences.Module.DotNetCore.1.3.0.nupkg` package version as follows:

`base-name`	`package-version`	`file-type`
`SignalSciences.Module.DotNetCore`	1.3.0	NUPKG

Node.js

Download the latest version of the Node.js module package or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The Node.js module directory appears.

2. Click the version of the module that you want to use.

- `package-version`: the version of the Signal Sciences module.
- `file-type`: the type of file.

The elements are assembled as follows:

```
[base-name]-[package-version].[file-type]
```

For example, you can break down the `sigsci-module-nodejs-2.1.2.tgz` package version as follows:

base-name	**package-version**	**file-type**
`sigsci-module-nodejs`	2.1.2	TGZ

IIS

Download the latest version of the IIS module package or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The IIS module directory appears.
2. Click the version of the module that you want to use.
3. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:
    - `base-name`: the type of Signal Sciences package. The base name for every module package version is `sigsci-module-iis`.
    - `architecture-type`: the architecture type, which is x64 (AMD64) for all ISS module packages.
    - `package-version`: the version of the Signal Sciences module.
    - `file-type`: the type of file.

    File names are assembled as follows:

    ```
    [base-name]-[architecture-type]-[package-version].[file-type]
    ```

    For example, you can break down the `sigsci-module-iis-x64-3.3.0.msi` package file name as follows:

base-name	**architecture-type**	**package-version**	**file-type**
`sigsci-module-iis`	x64	3.3.0	MSI

4. Optionally, click the file name of the related Secure Hash Algorithm 256-bit (SHA-256) key to download it. You can use the key to validate the module package that you downloaded.

HAProxy

Download the latest version of the HAProxy module package or follow these steps to download a specific version:

1. Navigate to the Signal Sciences package downloads site. The HAProxy module directory appears.
2. Click the version of the module that you want to use.
3. Click the name of the operating system you want to use. Options include Alpine, CentOS or RHEL, Debian, and Ubuntu.
4. Click the version of the operating system that you want to use.
5. Click the file name of the module package version that you want to download. The file names are comprised of the following elements:
    - `base-name`: the type of Signal Sciences package. The base name for every module package version is `sigsci-module-haproxy`.
    - `package-version`: the version of the Signal Sciences module.
    - `os-version`: the version of the operating system (OS).
    - `architecture-type`: the architecture type. Types include AMD64 or both AMD64 and ARM64.
    - `file-type`: the type of file.

    Element delimiters in the file name depend on the OS of the package. File names are assembled as follows:

Operating system	**Naming convention**
Alpine, Debian, Ubuntu	`[base-name]_[package-version]-[os-version]_[architecture-type].[file-type]`
CentOS or RHEL	`[base-name]-[package-version]_[os-version].[architecture-type].[file-type]`

    The following table demonstrates how you can break down file names into their elements:

`sigsci-module-haproxy_1.3.1-3.6_all.apk`	`sigsci-module-haproxy`	1.3.1	3.6	ARM64, AMD64	APK
`sigsci-module-haproxy-1.3.1_el8-1.noarch.rpm`	`sigsci-module-haproxy`	1.3.1	el8-1	AMD64	RPM

Using site dashboards

Site dashboards provide collections of metrics represented as cards on the Site Overview page. There are two types of dashboards:

- **System-generated:** These dashboards display cards that provide an overview of the most commonly useful system signals related to request anomalies and attacks directed at your site. You cannot modify system-generated dashboards.
- **Custom:** These dashboards display cards with metrics you've personally selected as useful system signals related to request anomalies and attacks directed at your site. Custom dashboards allow you to rearrange and edit the display of those signals to best suit your needs.

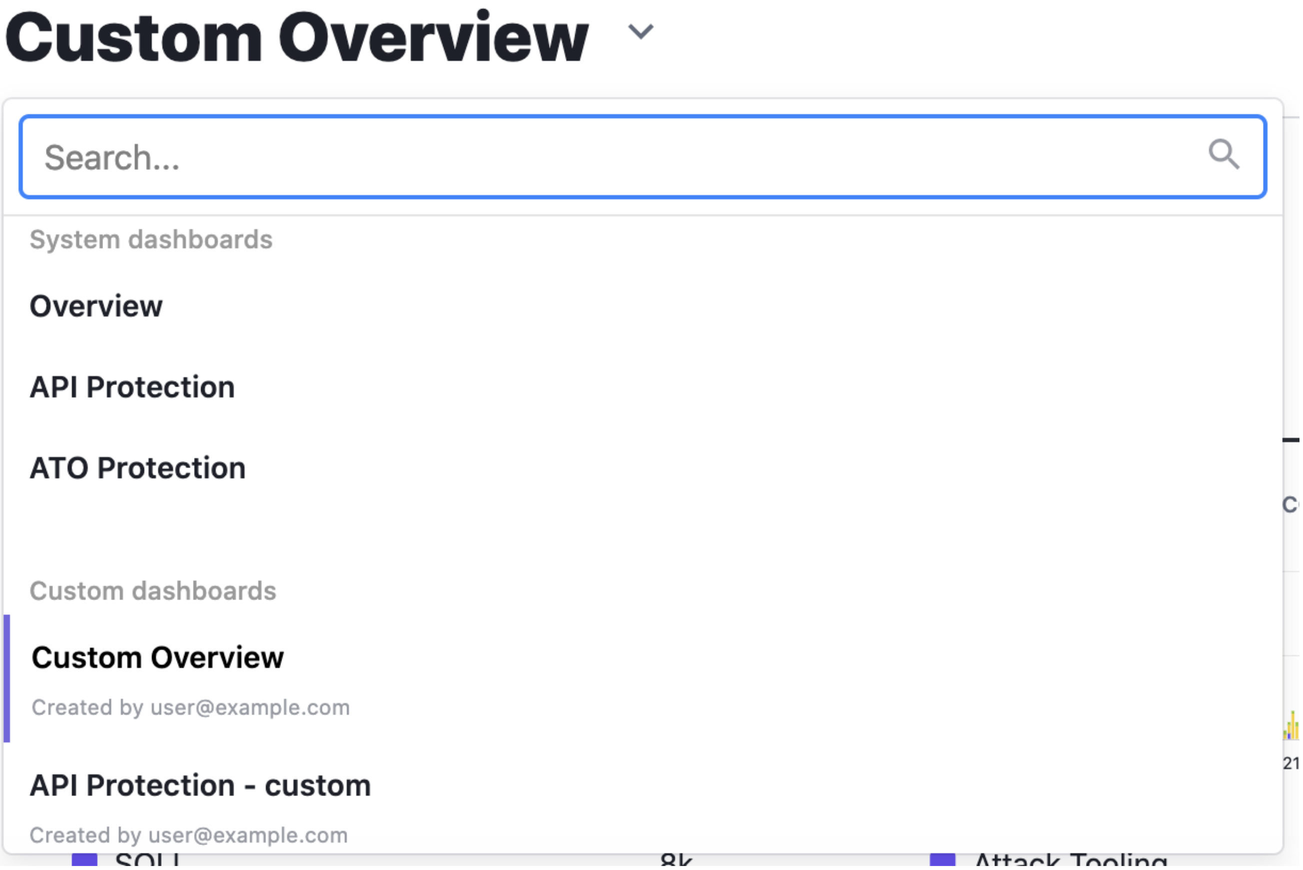
Limitations and considerations

Keep in mind that only Premier and Professional platforms support both system-generated and custom dashboards as part of your Site Overview. The Essentials platform only supports system-generated dashboards. For a complete list of available features at each platform level, check out our product description.

Viewing a dashboard

To view a system-generated or custom dashboard, follow these steps:

1. Click the name of your site in the upper left corner of the web interface. The Site Overview page appears.
2. Click the arrow next to the name of the current dashboard. The dashboards menu appears.



3. From the dashboards menu, select the dashboard you want to switch to. You can narrow down the list by using the search field. The selected dashboard appears.

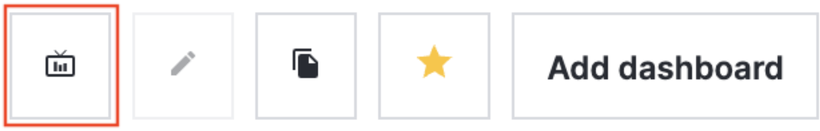
Viewing a dashboard in monitor view

mode, you can create a read-only URL so that you can view your dashboard on a TV.

To set up monitor view on a TV:

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.

2. Click the **Monitor view** icon. The Site Overview page appears in focus mode in the default grid view.



3. Click the **Share** button.

4. Click the **Read-only URL** switch.

5. Copy the link and open it on the TV you'd like to display the dashboard on.

You can change the focus mode view from the default grid view to carousel view by clicking the **Carousel** button. In the carousel view, the monitor will cycle through all cards on the Site Overview page. If necessary, you can generate a new URL, which invalidates the old URL. You can also disable the read-only URL altogether.

Setting a default dashboard

You can select a default dashboard that will automatically be selected when you log in to the Signal Sciences web interface.

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Default dashboard** icon, which appears as a star in the upper-right corner of the Site Overview page. The displayed dashboard becomes your default dashboard.

Working with custom dashboards

On the Site Overview page, you can create, duplicate, rename, and delete custom dashboards.

Creating a custom dashboard

To create a custom dashboard, follow these steps:

1. On the Site Overview page, click the **Add dashboard** button. The Add custom dashboard window appears.
2. Fill out the **Add custom dashboard** controls as follows:
   - In the **Name** field, enter the name of the new dashboard.
   - Optionally, click the **Choose default cards** link to display the default cards you can select to add to the custom dashboard.
3. Click the **Create dashboard** button. Your newly created dashboard appears on the Site Overview page.

Duplicating a dashboard

To duplicate a dashboard, follow these steps:

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Make a copy of this dashboard** icon, which appears as two stacked documents in the upper-right corner of the overview page. A duplicate of the selected dashboard appears on the Site Overview page.

Renaming a custom dashboard

To rename a custom dashboard, follow these steps:

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Edit dashboard** icon, which appears as a pencil in the upper-right corner of the Site Overview page. The Edit dashboard window appears.
3. In the **Name** field, enter a new name for the dashboard.
4. Click the **Update dashboard** button. The renamed dashboard appears on the Site Overview page.

Deleting a custom dashboard

dashboard.

2. Click the **Edit dashboard** icon, which appears as a pencil in the upper-right corner of the Site Overview page. The Edit dashboard form appears.
3. Click the **Delete dashboard** button. The delete dashboard confirmation window appears.
4. Click the **Delete dashboard** button. The dashboard is deleted.

Working with cards

You can surface relevant metrics on custom dashboards by adding cards that highlight meaningful data, editing cards to display specific signals, arranging the cards into a preferred layout, and deleting cards that aren't needed anymore.

Adding preset cards

To add a preset card to a custom dashboard, follow these steps:

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Add card** link in the empty card slot at the end of the dashboard. The Add card menu appears.
3. Select a preset card type. The Add card window appears.
4. Fill out the Add card window as follows:
    - In the **Title** field, enter a title for the card.
    - In the **Description** field, enter a description for the card.
    - From the **Signals** menu, select the signals the card will track. You can search for specific signals within the list by entering the name of the signal you want to search for.
5. Click the **Create card** button. The card is added to the dashboard.

Adding custom cards

To add a custom card to a custom dashboard, follow these steps:

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Add card** link in the empty card slot at the end of the dashboard. The Add card menu appears.
3. Select **Signals request chart:** to create a card with a bar graph or **Signals trend list:** to create a card that lists each signal and the percentage each signal increased or decreased over the selected time period. The Add card window appears.
4. Fill out the Add card window as follows:
    - In the **Title** field, enter a title for the card.
    - In the **Description** field, enter a description for the card.
    - From the **Signals** menu, select the signals the card will track. You can search for specific signals within the list by entering the name of the signal you want to search for.
5. Click the **Create card** button. The card is added to the dashboard.

Editing cards

To edit a card on a custom dashboard, follow these steps:

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Edit card** icon, which appears as a pencil when you hover over the upper-right corner of the card you want to edit. The Edit card form appears.
3. Fill out the **Edit card** window as follows:
    - In the **Title** field, enter a new title for the card.
    - In the **Description** field, enter a new description for the card.
    - From the **Signals** menu, remove signals by clicking the `x` icon in the name or add signals by selecting them from the menu.
4. Click the **Update card** button. The card is updated.

Rearranging cards

To arrange custom dashboard cards into a preferred layout, follow these steps:

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Move card** icon, which appears as a four-way arrow when you hover over the upper-right corner of a card, and then drag the card to the preferred location on the dashboard.

1. On the Site Overview page, select the relevant dashboard from the Dashboard menu. The Site Overview page displays the selected dashboard.
2. Click the **Edit card** icon, which appears as a pencil when you hover over the upper-right corner of the card. The Edit card window appears.
3. Click the **Delete card** button. The delete card confirmation form appears.
4. Click the **Delete** button. The card is removed from the dashboard.

Working with custom signals

Custom signals can be created to increase visibility into rules. Normally, requests that are immediately blocked or allowed by rules will not be visible in the console. To add visibility to immediately blocked or allowed requests, configure the rule to add a custom signal to the requests. A representative sample of requests that have been tagged with a custom signal will be listed in the Requests page of the console and can be found by searching for the custom signal.

Signals can be created on individual sites (Site Signals) as well as the corp as a whole (Corp Signals) to be easily used in multiple sites.

Limitations and considerations

When working with signals, keep the following things in mind:

- The Essentials platform does not support custom signals.
- Only Owners can create, edit, and delete corp-level signals.
- Signals are limited to 200 per corp plus 200 per site.

Viewing and Editing Signals

Corp Signals can be managed by going to **Corp Rules** > **Corp Signals**, while Site Signals can be managed by navigating to a specific site and going to **Rules** > **Site Signals**. Any signals you have created will be listed on these pages. Edit or remove any of the signals by clicking the **Details** button to the right of the signal.

Creating Signals

You can create custom signals at the corp-level and site-level.

Corp Signals

1. From the **Corp Rules** menu, select **Corp Signals**. The corp signals menu page appears.
2. Click **Add corp signal**. The add corp signal menu page appears.
3. In the **Signal name** field, enter the name of the custom signal.
4. In the **Description (optional)** field, you may enter an optional description for the custom signal.
5. Click **Create corp signal**.

   Note: Only Owners can create, edit, and delete Corp Signals.

Site Signals

1. From the **Site Rules** menu, select **Site Signals**. The site signals menu page appears.
2. Click **Add site signal**. The add site signal menu page appears.
3. In the **Signal name** field, enter the name of the custom signal.
4. In the **Description (optional)** field, you may enter an optional description for the custom signal.
5. Click **Create site signal**.

Using Signals

When creating a rule, the **Add signal** action can be used to tag requests processed by the rule with a custom signal. Select the appropriate signal or create a new signal by selecting **Create new signal** in the dropdown menu.

Redacting data

To maintain data privacy, Signal Sciences redacts sensitive data from requests before they reach the platform backend.

Selective data transfer and redaction

The Signal Sciences agent filters requests locally to determine if they contain an attack. Only requests that are marked as attacks or anomalies are then sent to the Signal Sciences backend after additional filtering and sanitizing are done. Once the agent identifies a potential

Sciences backend. Additionally, specific portions of the request are automatically redacted and never sent to the backend, including tokens, credentials, and known patterns such as credit card and social security numbers.

JSON API payloads

Signal Sciences automatically parses JSON key-value pairs and treats them like request parameters. The following sample requests demonstrate how redactions work within the context of a request.

The initial request:

```
POST /request HTTP/1.1
Content-Length: 72
Content-Type: application/json
Host: api.example.com
{"user":"user@api.example.com","password":"<script>alert(1)</script>mypassword","zip":94089}
```

What's sent to Signal Sciences:

```
POST /request HTTP/1.1
Host: api.example.com

password=
```

The initial request:

```
POST /request HTTP/1.1
Content-Length: 72
Content-Type: application/json
Host: api.example.com

{"user":"user@api.example.com","password":"mypassword","zip":"<script>alert(1)</script>94089"}
```

What's sent to Signal Sciences:

```
POST /request HTTP/1.1
Host: api.example.com

zip=<script>alert(1)</script>
```

Sensitive headers

Signal Sciences redacts the following from requests:

- Explicit names: `authorization`, `x-auth-token`, `cookie`, `set-cookie`
- Any names that contain: `-token`, `-auth`, `-key`, `-sess`, `-pass`, `-secret`
- Query strings from `referer` and `location`

The initial request:

```
POST /example?sort=ascending HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:35.0)
Accept: text/html, application/xhtml+xml
Content-Length: 57
Cookie: foo=bar

sensitive=hunter2&foobar=<script>alert(1)</script>&page=3
```

What's sent to Signal Sciences:

```
POST /example HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:35.0)

foobar=<script>alert(1)</script>
```

the entire contents of the sensitive parameter. These parameters include:

- `api_key`
- `password`
- `passwd`
- `pass`
- `pw`
- `user`
- `login`
- `loginid`
- `username`
- `email`
- `key`
- `id`
- `sid`
- `token`
- `request_token`
- `access_token`
- `csrfmiddlewaretoken`
- `oauth_verifier`
- `confirm_password`
- `password_confirmation`

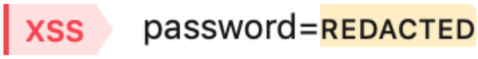
The initial request:

```
POST /example HTTP/1.1

username=<script>alert("jsmith")</script>
```

What's sent to Signal Sciences:

```
POST /example HTTP/1.1

username=[redacted]
```

The console clearly displays which parameters have been redacted. Redacted parameters are replaced with the word `REDACTED` highlighted in yellow.



XSS password=REDACTED

Sensitive patterns

Signal Sciences automatically redacts known patterns of sensitive information, which includes the following:

- **Credit card numbers:** values like `4111-1111-1111-1111` become `0000-0000-0000-0000`
- **Social security numbers:** values like `078-05-1120` become `000-00-0000`
- **GUIDs:** values like `3F2504E0-4F89-41D3-9A0C-0305E82C3301` become `0000000-0000-0000-0000-000000000000`
- **Bank account (IBAN) numbers:** values like `DE75512108001245126199` become `AA00aaaa0000000`

The initial request:

```
POST /example HTTP/1.1

credit_card_example=<script>alert("4111-1111-1111-1111")</script>
```

What's sent to Signal Sciences:

```
POST /example HTTP/1.1

credit_card_example=<script>alert("0000-0000-0000-0000")</script>
```

XSS card=<script>alert('REDACTED SSN')</script>

Custom redactions

In addition to the redactions listed above, you can also specify additional fields to redact from requests. For example, if your password field is named `foobar` instead of `password`, that field can be specified for redaction.

> **Important:** Accounts are limited to 100 redactions per site.

Creating custom redactions

When you have a sensitive field that is not filtered out by default, you can create a custom field redaction:

1. From the **Rules** menu, select **Redactions**. The Redactions page appears.
2. Click the **Add redaction** button. The add redaction menu page appears.
3. In the **Field name** field, enter the name of the field to be redacted.
4. From the **Field type** menu, select the type of field to be redacted. Options include Request parameter, Request header, or Response header.
5. Click the **Create redaction** button.

Editing custom redactions

To edit a custom redaction, complete the following steps:

1. From the **Rules** menu, select **Redactions**. The Redactions page appears.
2. Click **View** to the right of the custom redaction you want to edit. The details page for the redaction appears.
3. Click **Edit redaction**. The Edit form appears.
4. Change the **Field name** and **Field type** as needed.
5. Click the **Update redaction** button.

Deleting custom redactions

1. From the **Site Rules** menu, select **Redactions**. The redactions menu page appears.
2. Click the **View** link to the right of the custom redaction you want to delete. The view redaction menu page appears.
3. Click the **Remove redaction** button. The delete redaction menu page appears.
4. Click the **Delete** button to delete the redaction.

Transparency

To allow for easy verification of what the agent sends to the backend, Signal Sciences provides a way to view all agent to backend communication.

Verifying in the console

To verify our agents are correctly filtering and sanitizing requests, we provide a raw log of data that's sent from our agents:

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. Click on **Agents**. The agents page appears.
4. Click on the **Agent ID**. The agent details menu page appears.
5. Click the **Requests** tab. A list of all requests processed by the agent appears.
6. Review the requests and verify that data is correctly redacted.

Verifying with the agent

You can also verify directly from the agent itself by setting the `debug-log-uploads` agent configuration option. For example, if you want to log all agent uploads in formatted JSON, add the following line to your agent configuration file (by default at `/etc/sigsci/agent.conf`):

```
debug-log-uploads = 2
```

Additional information about agent configuration options can be found here.

Upgrading the Apache Module

> Check the Apache Changelog to see what's new in the Apache Module.

Our Module package is distributed in our package repositories, if you haven't already, configure our repository on your system.

```
sudo apt-get update
```

```
sudo apt-get install sigsci-module-apache
```

2. Restart Apache. After upgrading the module you'll need to restart your Apache service.

Upgrading the Apache module on Red Hat/CentOS systems

1. Upgrade the Apache module package

   RHEL 6/CentOS 6

   ```
   sudo yum update
   ```

   Apache 2.2:

   ```
   sudo yum install sigsci-module-apache
   ```

   Apache 2.4:

   ```
   sudo yum install sigsci-module-apache24
   ```

   RHEL 7/CentOS 7

   ```
   sudo yum update
   ```

   ```
   sudo yum install sigsci-module-apache
   ```

2. Restart Apache. After upgrading the module you'll need to restart your Apache service.

Generic Webhooks

Our generic webhooks integration allows you to subscribe to notifications for certain activity on Signal Sciences.

Adding a webhook

1. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
2. Click **Add site integration**. The add site integration menu page appears.
3. Select the **Generic Webhook** integration. The Generic Webhook integration setup page appears.
4. In the **Webhook URL** field, enter a URL to receive the notifications at.
5. Select if you want to be alerted regarding **All activity** or **Specific activity**.
   - If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
6. Click **Create site integration**.

Notifications format

Notifications are sent with the following format:

```
{
  "created": "2022-12-09T10:43:54-08:00",
  "type": "flag",
  "payload": ...,
  "link":"dashboard link to event"
}
```

X-SigSci-Signature Header

All requests sent from the generic webhook integration contain a header called `X-SigSci-Signature`. The value is an HMAC-SHA256 hex digest hashed using a secret key generated when the generic webhook was created.

Verification is done by creating an HMAC-SHA256 hex digest of the generic webhook payload using the signing key and comparing the result to the value of the `X-SigSci-Signature` header.

X-SigSci-Signature Header Verification Example Code

The examples show header verification code for `X-SigSci-Signature`.

Go

```
package main

import (
        "crypto/hmac"
        "crypto/sha256"
        "encoding/hex"
        "fmt"
)

// CheckMAC reports whether messageMAC is a valid HMAC tag for message.
func CheckMAC(message, messageMAC, key []byte) bool {
        mac := hmac.New(sha256.New, key)
        mac.Write(message)
        expectedMAC := mac.Sum(nil)

        return hmac.Equal(messageMAC, expectedMAC)
}

func main() {
        key := []byte("[insert signing key here]")

        h := "[insert X-SigSci-Signature value here]"

        json := []byte(`[insert JSON payload here]`)

        hash, err := hex.DecodeString(h)
        if err != nil {
                log.Fatal("ERROR: ", err)
        }

        ok := CheckMAC(json, hash, key)

        fmt.Println(ok)
}
```

Python

```
import hashlib
import hmac

def checkHMAC(message, messageMAC, key):
    mac = hmac.new(key, message, digestmod=hashlib.sha256).hexdigest()

    return mac == messageMAC

key = '[insert signing key here]'

h = '[insert X-SigSci-Signature value here]'

json = '[insert JSON payload here]'

ok = checkHMAC(json, h, key)
```

Ruby

```
require 'openssl'
require "base64"

key = '[insert signing key here]'
h = '[insert X-SigSci-Signature value here]'
json = '[insert JSON payload here]'

hash  = OpenSSL::HMAC.hexdigest('sha256', key, json)

puts hash == h
```

Bash

```
#!/bin/bash

function check_hmac {
  json="$1"
  messageMAC="$2"
  key="$3"

  result=$(echo -n "$json" | openssl dgst -sha256 -hmac "$key")
  if [ "$result" == "$messageMAC" ]
  then
          return 0
  else
          return 1
  fi
}

key='[insert key here]'
h='[insert X-SigSci-Signature value here]'
json='[insert JSON payload here]'

check_hmac "$json" $h $key
```

Activity types

Activity type	Description	Payload
`siteDisplayNameChanged`	The display name of a site was changed	
`siteNameChanged`	The short name of a site was changed	
`loggingModeChanged`	The agent mode (`Blocking`, `Not Blocking`, `Off`) was changed	Get site by name
`agentAnonModeChanged`	The agent IP anonymization mode was changed	Get site by name
`flag`	An IP address was flagged	Get event by ID
`expireFlag`	An IP address flag was manually expired	List events
`createCustomRedaction`	A custom redaction was created	Create a custom redaction
`removeCustomRedaction`	A custom redaction was removed	Remove a custom redaction
`updateCustomRedaction`	A custom redaction was updated	Update a custom redaction
`customTagCreated`	A custom signal was created	
`customTagUpdated`	A custom signal was updated	
`customTagDeleted`	A custom signal was removed	
`customAlertCreated`	A custom alert was created	Create a custom alert
`customAlertUpdated`	A custom alert was updated	Update a custom alert
`customAlertDeleted`	A custom alert was removed	Remove a custom alert
`detectionCreated`	A templated rule was created	

`detectionDeleted`	A templated rule was removed	
`listCreated`	A list was created	Create a list
`listUpdated`	A list was updated	Update a list
`listDeleted`	A list was removed	Remove a list
`ruleCreated`	A request rule was created	
`ruleUpdated`	A request rule was updated	
`ruleDeleted`	A request rule was deleted	
`customDashboardCreated`	A custom dashboard was created	
`customDashboardUpdated`	A custom dashboard was updated	
`customDashboardReset`	A custom dashboard was reset	
`customDashboardDeleted`	A custom dashboard was removed	
`customDashboardWidgetCreated`	A custom dashboard card was created	
`customDashboardWidgetUpdated`	A custom dashboard card was updated	
`customDashboardWidgetDeleted`	A custom dashboard card was removed	
`agentAlert`	An agent alert was triggered	

Enabling and disabling two-factor authentication

We support two-factor authentication (2FA) via apps that support both HOTP (RFC-4226) and TOTP (RFC-6238). This includes Duo Security and Google Authenticator for both iPhone and Android.

> **Note:** Two-factor authentication settings are set at the user-level for a particular corp. This means that a user only needs to configure two-factor authentication once to access all the sites to which they belong.

Enabling two-factor authentication

1. From the **My Profile** menu, select **Account Settings**. The account settings menu page appears.
2. Select **Enable**. The two-factor authentication setup window appears.
3. Scan the QR code with your authenticator app or click **Enter code manually instead** and enter the code manually into your authenticator app.
4. Click the **Continue** button.
5. Enter the verification code from your authenticator app.
6. Click the **Verify** button.

Disabling two-factor authentication

1. From the **My Profile** menu, select **Account Settings**. The account settings menu page appears.
2. Click the **Disable** button to disable two-factor authentication.

About the Corp Rules menu

The Corp Rules menu is located on the right side of the corp navigation bar. From the Corp Rules menu, you can access the following pages:

- Corp Rules
- Corp Lists
- Corp Signals

Before you begin

Be sure you know how to access the web interface controls.

About the Corp Rules page

Selecting **Corp Rules** from the Corp Rules menu displays the Corp Rules page. From this page, you can manage your corp rules. Corp rules block, allow, and tag requests and exclude system signals for arbitrary sets of conditions.

The Corp Rules page lists existing corp rules and provides controls to filter the list of rules. Specifically, you can filter the list by:

- **Scope:** allows you to filter the list by whether rules apply to select sites or all sites in your corp.
- **Type:** allows you to filter the list by the type of rule. Rule types include request rules and signal exclusion rules.

request, allow request, add signal, and exclude signal.

From the Corp Rules page, you can also create corp rules by clicking the **Add corp rule** button.

About the Corp Lists page

Selecting **Corp Lists** from the Corp Rules menu displays the Corp Lists page. From this page, you can manage your corp lists. Corp lists are sets of custom data used in corp rules, such as a list of countries a corp doesn't do business with.

The Corp Lists page contains a table listing existing corp lists. Each row has a **View** link which takes you to a detailed view of a corp list and additional controls that enable you to edit or delete the corp list.

From the Corp Lists page, you can also create a corp list by clicking the **Add corp list** button.

About the Corp Signals page

Selecting **Corp Signals** from the Corp Rules menu displays the Corp Signals page. From this page, you can manage your corp signals. Corp signals are tags that describe requests.

The Corp Signals page contains a table listing existing corp signals. Each row has a **View** link which takes you to a detailed view of a corp signal and additional controls that enable you to edit or delete the corp signal.

From the Corp Signals page, you can also create a corp signals by clicking the **Add corp signals** button.

What's next

Dig deeper into details about all areas of the web interface controls.

Defining rule conditions

When creating rules, you define a set of conditions that outline the circumstances under which requests should be allowed, blocked, rate limited, or tagged.

About rule conditions

A rule condition is made up of a field, an operator, and a value.

Fields

A rule conditions is based on a specific field.

When creating a request rule or an advanced rate limiting rule, you can base a rule's conditions on the following request fields:

Field	Type	Properties
Agent name	String	Text or wildcard
Country	Enum	ISO countries
Domain	String	Text or wildcard
IP address	IP	Text or wildcard, supports CIDR notation
Method	Enum	`GET, POST, PUT, PATCH, DELETE, HEAD, TRACE`
Path	String	Text or wildcard
POST parameter	Multiple	`Name (string), Value (string)`
Query parameter	Multiple	`Name (string), Value (string)`
Request cookie	Multiple	`Name (string), Value (string)`
Request header	Multiple	`Name (string), Value (string), Value (IP)`
Response code	String	Text or wildcard
Response header	Multiple	`Name (string), Value (string)`
Scheme	Enum	`http, https`
Signal	Multiple	`Type (signal), Parameter name (string), Parameter value (string)`
User agent	String	Text or wildcard

When creating a signal exclusion rule, you can base the rule conditions on the same request fields and fields specific to the particular signal that is being excluded:

Parameter value String Text or wildcard

Operators

When creating rules, operators are used to specify the logic of your rule when matching conditions. For example, the equals operator is used to check if a value in the request matches the value in the rule condition exactly, such as when attempting to match a specific IP address or path.

Operator	Description	Example Match
Equals	Checks if the request value matches the rule condition value exactly.	`203.0.113.169` Equals `203.0.113.169`
Does not equal	Checks if the request value does not match the rule condition value exactly.	`203.0.113.169` Does not equal `192.0.2.191`
Contains	Checks if the rule condition value being checked is contained within the request value; for example, to check if a substring is found within a larger string.	`thisisanexamplestring` Contains `example`
Does not contain	Checks if the rule condition value being checked is not contained within the request value; for example, to check if a substring is not found within a larger string.	`thisisanexamplestring` Does not contain `elephant`
Like (wildcard)	Allows the use of wildcard characters in matching; checks if the request value matches the rule condition value.	`bats` Like (wildcard) `[bcr]ats`
Not like (wildcard)	Allows the use of wildcard characters in matching; checks if the request value does not match the rule condition value.	`bats` Not like (wildcard) `[hps]ats`
Matches (regexp)	Allows the use of regular expressions in matching; checks if the request value matches the rule condition value. Platform availability: Professional and Premier.	`bats` Matches (regexp) `(b|c|r)ats`
Does not match (regexp)	Allows the use of regular expressions in matching; checks if the request value does not match the rule condition value. Platform availability: Professional and Premier.	`bats` Does not match (regexp) `(h|p|s)ats`
Is in list	Checks if the request value matches any of the values in a specific list. Platform availability: Professional and Premier.	`203.0.113.169` Is in list `Known IP Addresses`
Is not in list	Checks if the request value does not match any of the values in a specific list. Platform availability: Professional and Premier.	`192.0.2.191` Is not in list `Known IP Addresses`

Exists where	Allows for name-value pair subconditions in matching; checks if the subconditions exist in the request value.	**Request value:**
		`Content-Type: application/xml`
		Condition:
		`Request Header` Exists where
		Subconditions:
		`Name` Equals `Content-Type`
		And
		`Value (string)` Equals `application/xml`

Does not exist where	Allows for name-value pair subconditions in matching; checks if the subconditions do not exist in the request value.	**Request value:**
		`Content-Type: application/xml`
		Condition:
		`Request Header` Does not exist where
		Subconditions:
		`Name` Equals `Content-Type`
		And
		`Value (string)` Equals `application/json`

Wildcards

If you need to match a literal `*`, `?`, `[`, or `]` character, escape them with the `\` character. For example: `\*`.

The `Like` (wildcard) operator requires a full string match. If you're trying to match part of a string, you may need to include the `*` wildcard at the beginning or end to include the rest of the string for correct matching.

Regular expressions are not supported with the `Like` (wildcard) operator. If you want to use regular expressions, you must use the `Matches` (regexp) operator.

Field value case sensitivity

All fields in rules are case sensitive with the exception of header names.

For example, if you create a rule that looks for a header named `X-Custom-Header`, it will match on requests with headers named `X-Custom-Header` and `x-custom-header` because header names aren't case sensitive. However, if the rule looks for the value `Example-Value`, it will only match on `Example-Value` and not `example-value` because all other rule fields—such as header values in this example—are case sensitive.

> **Note:** When constructing a regular expression pattern (regexp) that matches on a header name, POST parameter name, or query parameter name, write the pattern in all lowercase or prefix the pattern with a case insensitive regex matching mode of `(?i)`.

Path syntax best practices

When basing a rule condition on the Path field, keep these best practices in mind:

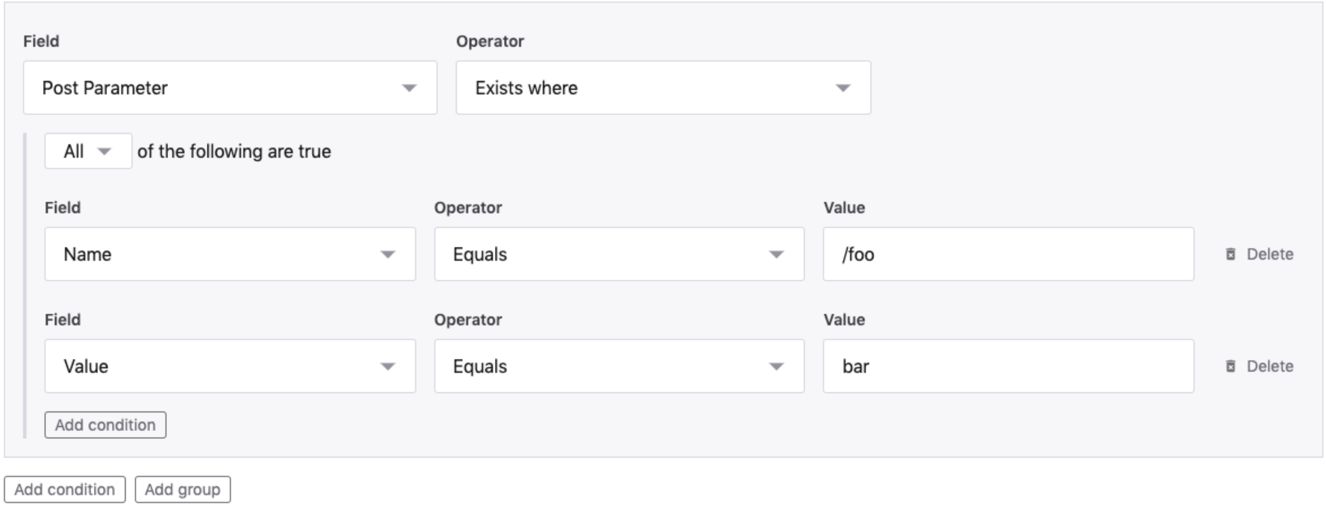
- **Always use leading slashes**. For a URL like `https://example.com/some-path`, the correct path syntax to use would be `/some-path`.
- **Use relative paths instead of absolute URLs.** For example, if the absolute URL to the login page on your site is `https://example.com/login`, then `/login` is the correct path syntax to enter when configuring your login signals.
- **Take care when using trailing characters in your paths.** Since our path syntax uses exact matching, trailing characters can sometimes return zero matches. Consider an example where the path to your login page is `https://example.com/login/`:
  - `/login/` will return a match.
  - `/login` with not return a match.

Post Parameter field

When creating rules that inspect the JSON body of POST requests, Post Parameter names require a leading `/`. For example, if the JSON payload is:

```
{
  "foo": "bar"
}
```

Then the name of the Post Parameter will need to be `/foo` in the rule.



Field: Post Parameter
Operator: Exists where

All of the following are true

Field: Name
Operator: Equals
Value: /foo

Field: Value
Operator: Equals
Value: bar

The leading / on of Post Parameter name facilitates nested values. For example, `/foo/bar` for a payload such as:

```
{
  "foo": {
    "bar": ["value1", "value2"]
  }
}
```

Country field

The Country field allows you to specify conditions that match against a particular country to block or allow traffic. Geolocation can be combined with other conditions like path or domain.

Where does the geodata come from?

We license MaxMind's Geolite2 data and distribute it within our agent. This data is updated periodically and included with newer agent releases as well as dynamically updated similar to rule updates.

How often is geodata updated?

We update our geodata and release an agent monthly (typically the second week of the month). At the same time as the agent release, the new geodata is deployed to our cloud tagging so that the latest country information is present. This will be a minor agent increment. This data is also dynamically updated similar to rule updates and these agents will download and cache the updated geolocation data.

What happens if my agent is out-of-date?

If your agent is out-of-date, then an IP may be blocked or allowed based on outdated geo information. Or requests may display in the console that would have been blocked with newer geodata. The country displayed in the console will reflect the latest available geodata.

How do dynamic geolocation data updates work?

The geolocation data is packaged up for the agent to download whenever there is an update. This data is cached locally on the agent machine. The cache location is under the shared-cache-dir directory which defaults to `{$TMPDIR|%TMP%|%TEMP%}/sigsci-agent.cache/`). The geolocation data is only downloaded if it does not exist locally or the data is not up-to-date.

Requirements for this functionality:

- The filesystem where this cache directory resides must be:
  - Writeable by the user running the agent
  - Have at least 5MB of free space
  - While auto-detection of the cache directory normally works fine, you may need to configure shared-cache-dir on some systems where a TEMP space is not defined (e.g., where `$TMPDIR` or `%TMP%` or `%TEMP%` environment vars are not set properly)
- The network must be capable of:
  - Downloading from the base download-url (this is the same base URL as normal rule updates)
  - Downloading the data (currently about 2MB) within the timeout limit (currently 60 seconds)

If the dynamic geolocation data cannot be downloaded, then the agent will default to the geolocation data packaged with the agent.

Custom agent response codes allow you to specify the HTTP status code that is returned when a request to your web application is blocked. By default, all block actions return the 406 custom agent response code. You can change this default behavior by:

- updating the site default blocking response code from 406 to an alternative response code. Blocking actions use the site default blocking response code unless a different response code is specified in a rule.
- creating request rules that have a block action and that will return a specified response code.
- creating advanced rate limit rules that have a block action and that will return a specified response code.

Custom agent response codes can facilitate additional actions at the edge depending on the rule triggered. For example, a specific custom agent response code can be used to tell your CDN to redirect the request to a CAPTCHA. The Fastly CDN supports custom agent response codes in VCL to redirect requests to other pages (e.g., custom error pages).

Limitations

When working with custom agent response codes, keep the following things in mind:

- The Essential platform does not support custom agent response codes.
- Supported custom agent response codes are 301, 302, and 400-599.
- Each site may have up to 5 unique response codes at any time.
- There is no limit to the total number of rules that use custom agent response codes.
- Custom agent response codes require a minimum agent and module version. When an unsupported module version is told to block a request due to a rule that uses a custom agent response code, that request will **not be blocked**.

Response code precedence

Blocking actions will use the site default blocking response code unless a different response code is specified in a rule. Examples of this rule are as follows:

- When a templated rule blocks a request, the site default blocking response code is returned.
- When a rule with the site default blocking response code and a rule with a custom agent response code both block a request, the custom agent response code is returned.

When rules with different custom agent response codes block the same request, the custom agent response code created first takes precedence over other relevant custom agent response codes. For example, let's say that your site has the following rules:

Rule	Condition	Action	Date created
E	IP Address (Client) equals `192.0.2.0`	Block and respond with 500	2022-12-01
D	IP Address (Client) equals `192.0.2.0`	Block and respond with 400	2022-10-01
C	IP Address (Client) equals `192.0.2.0`	Block and respond with 404	2022-08-01
B	Path equals `/example/path`	Block and respond with 400	2022-06-01
A	Path equals `/example/redirect`	Block and respond with 301	2022-04-01

In this example, a client with an IP address of `192.0.2.0` makes a request to the `/custom-limits` page of your web application. As the request meets the conditions of rules C, D, and E, the request is blocked. While rule C was created before rules D and E, the 400 response code from rule D is returned because it is the oldest relevant response code. Specifically, the 400 response code was first added to rule B on June 1st and the 404 and 500 response codes were created on August 1st and December 1st respectively.

Selecting custom agent response codes

Because custom agent response codes can be returned to upstream systems, ensure you understand the behavior of your upstream systems. Specifically, keep the following things in mind when selecting a custom agent response code:

- Some CDNs automatically cache certain response codes. For example, the Fastly CDN automatically caches 301, 302, 404, and 410 responses.
- Using a 401 response code may result in a username and password prompt to the client browser.
- Using response codes such as 400 or 403 may result in an artificial increase of measured "bad request" or "forbidden" requests.
- Response codes in the 5xx range are generally associated with server connections or application errors.

Minimum version support

The following agent and module versions support custom agent response codes:

Name	Minimum version
Agent	4.10+

Cloud Foundry	Any
Envoy	Any
Golang	1.8.0+
HAProxy	1.2.0+
Heroku	Any
IBM Cloud	Any
IIS	2.2.0+
Java	2.1.1+
.Net	1.6.0+
.Net Core	1.3.0+
NGINX	1.4.0+
NGINX C Binary	1.0.44+
Node.js	1.6.1+

Unsupported agents and modules handle requests that should be blocked by rules with custom agent response codes in the following ways:

Agent	Module	Result
Supported	Supported	Blocked with custom agent response code
Supported	Unsupported	Not blocked
Unsupported	Supported	Blocked with default response code of 406
Unsupported	Unsupported	Not blocked
Supported (Reverse Proxy)	N/A	Blocked with custom agent response code
Unsupported (Reverse Proxy)	N/A	Blocked with default response code of 406

Using redirect custom agent response codes

With redirect custom agent response codes (i.e., 301 and 302), you can specify the absolute or relative URL of the redirect location.

The redirect URL can pass one instance of the `{{REQUESTID}}` variable (e.g., `https://www.example.com/blocked/?reqid={{REQUESTID}}`). When used, this variable is replaced with the ID of the relevant request before the client is sent to the redirect location.

Using lists in rules

About Lists

Lists can be used to create and maintain sets of data for use when creating rules. Lists allow you to easily reuse the same sets of data across multiple rules. Lists can be created on individual sites (Site Lists) as well as the corp as a whole (Corp Lists) to be easily used in multiple sites.

For example, you could create a list of prohibited countries that you don't do business with. You could then use this list in any rules that involve those countries, such as rules to track registration or login attempts originating from those countries. If a prohibited country changes, simply update the list instead of updating every rule that uses it.

Lists can consist of the following types of data:

- Countries
- IP addresses
- Strings
- Wildcards

Limitations and caveats

- Lists support CIDR notation for IP address ranges.
- Lists are limited to 25 per corp plus 25 per site.
- Lists can contain a maximum of 5000 items.

Creating a List

Create both Corp and Site lists using these steps.

Corp Lists

1. From the **Corp Rules** menu, select **Corp Lists**. The corp lists menu page appears.

4. In the **Name** field, enter the name of the list.
5. Optionally, in the **Description (optional)** field, enter a description for the list.
6. In the **Entries** field, enter the items that will comprise the list. Each entry must be on its own line.
7. Click **Create corp list**.

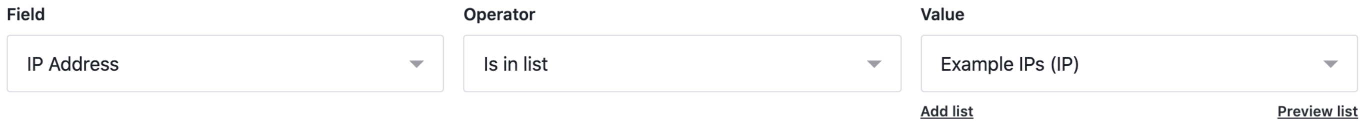
> **Note:** Only Owners can create, edit, and delete Corp Lists. This is because Corp Lists have the ability to manipulate traffic across every site and other user types can only manage Rules and Lists for sites they have access to.

Site Lists

1. From the **Site Rules** menu, select **Site Lists**. The site lists menu page appears.
2. Click **New list**. The new list menu page appears.
3. From the **Type** menu, select the type of data the list will contain.
4. In the **Name** field, enter the name of the list.
5. Optionally, in the **Description (optional)** field, enter a description for the list.
6. In the **Entries** field, enter the items that will comprise the list. Each entry must be on its own line.
7. Click **Create site list**.

Using a List

When creating a rule, select **Is in list** or **Is not in list** for the operator, then select the list from the value dropdown menu.



Field	Operator	Value
IP Address	Is in list	Example IPs (IP)

For more information about creating rules, see Rules.

Linking Fastly accounts

You can link your Fastly and Signal Sciences accounts, allowing you to log in to either account using your Fastly credentials. Once logged in you can freely switch between the Signal Sciences and Fastly consoles.

> **IMPORTANT:** Be sure you understand the limitations and considerations of linking before you connect your accounts.

Linking your Fastly and Signal Sciences accounts only affects authentication when logging into the Signal Sciences console. Other settings such as user roles and API access tokens are not affected.

Limitations and considerations

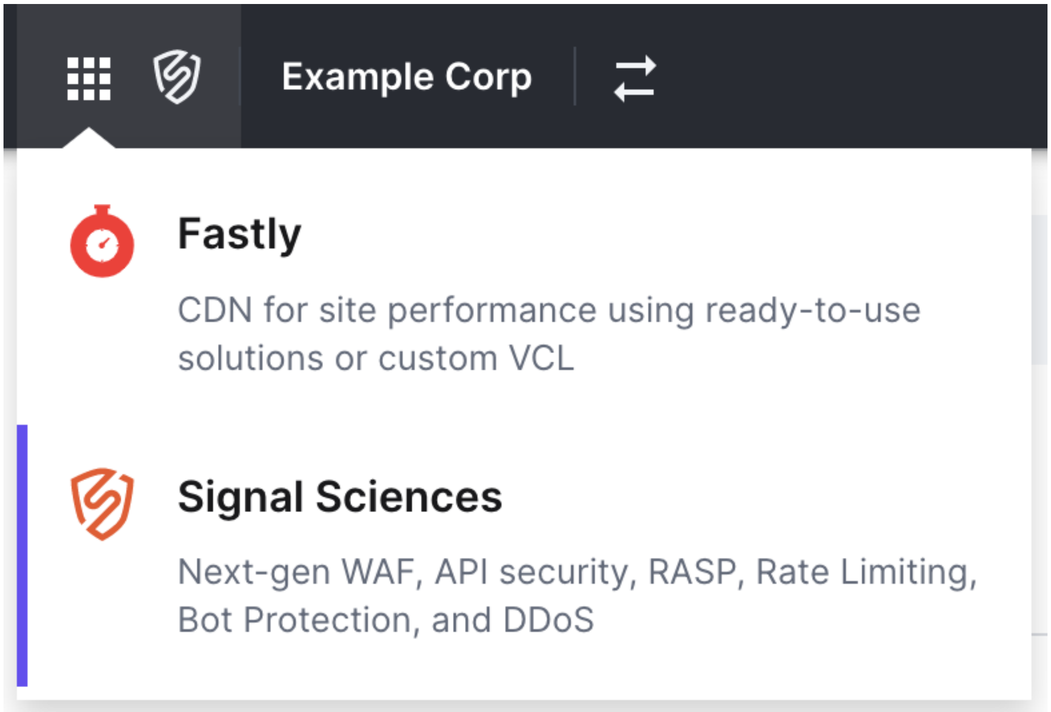
Before linking your accounts, keep in mind the following:

- **Account linking is not reversible.** You can't unlink your Signal Sciences and Fastly accounts once they have been linked.
- **Linked accounts require you to use your Fastly account credentials to log in.** After linking your accounts, you will only be able to log in to the Signal Sciences console using your Fastly account credentials.
- **SAML authentication is not supported.** Linked accounts authenticate using your Fastly email address and password, rather than through your identity provider.
- **Accounts using the bypass SSO feature can't be linked.** Signal Sciences accounts set to bypass SSO can not be linked to a Fastly account.
- **2FA is supported.** 2FA is supported for linked accounts but it must be enabled on both your Fastly and Signal Sciences accounts before you will be able to link them.

How to link your Fastly and Signal Sciences accounts

1. Log in to the Signal Sciences console.
2. From the **Profile** menu, select **Account Settings**. The account settings management page appears.
3. Under the **Link Fastly account** header, click **Link account**. The link account page appears.
4. Click **Start Verification**. The Fastly account login page appears.
5. Enter your Fastly account login credentials.
6. Click **SIGN IN**. The account link confirmation page appears.
7. Click **Link Fastly account**. A confirmation appears stating the account has been successfully linked.
8. Click **Account settings** to return to the account settings management page, or click **View dashboard** to return to the Corp Overview page.

console or the Fastly app's web interface.


Example Corp
Fastly
CDN for site performance using ready-to-use solutions or custom VCL
Signal Sciences
Next-gen WAF, API security, RASP, Rate Limiting, Bot Protection, and DDoS

> **TIP:** You can also access the Signal Sciences console directly from the **Overview** and **Next-Gen WAF** tabs on the **Secure** page in the Fastly web interface.

Working with advanced rate limiting rules

Advanced rate limiting rules put a cap on how often an individual client can send requests that meet set conditions before all or some requests from that same client are blocked or logged.

Limitations and considerations

When working with advanced rate limiting rules, keep the following things in mind:

- Advanced rate limiting is only included with the Premier platform and certain packaged offerings. It is not included as part of the Professional or Essential platforms.
- Each site is limited to a maximum of 15 rate limit rules.
- A given signal can only be used as the threshold signal for a single rate limit rule. A signal can't be used as the threshold signal in more than one rate limit rule.

How advanced rate limiting rules work

When the web application you're protecting starts receiving requests that meet the conditions of a rate limit rule, the requests are tagged with the a user-selected signal called the *threshold signal*. Threshold signals are tallied and counted towards the threshold for that rule. When the threshold signal count for a single client exceeds the rule's threshold, that client is rate limited.

How the client is rate limited depends on the Match type and Action type fields for that rate limit rule. The **Match type** field defines which requests from the client should be blocked or logged after the threshold has been passed. Field options include:

- **Rule conditions:** rate limit requests from the client that match the rate limit rule's conditions. See an example use case of this option.
- **Other signal:** rate limit requests from the client that are tagged with a user-selected signal called the *action signal*. When the action signal is not an attack or anomaly signal, you need a request rule that tags requests with the selected signal. See an example use case of this option.
- **All requests:** rate limit all requests from the client.

The **Action type** field defines whether requests that meet the Match type condition should be blocked or logged. When the **Action type** field is set to **Block**, subsequent requests that meet the Match type are blocked and tagged with these signals:

- the threshold signal defined in the rule

When the **Action type** field is set to **Log**, subsequent requests that meet the Match type are logged and tagged with these signals:

- the threshold signal defined in the rule
- `Rate Limit` system signal

Data storage

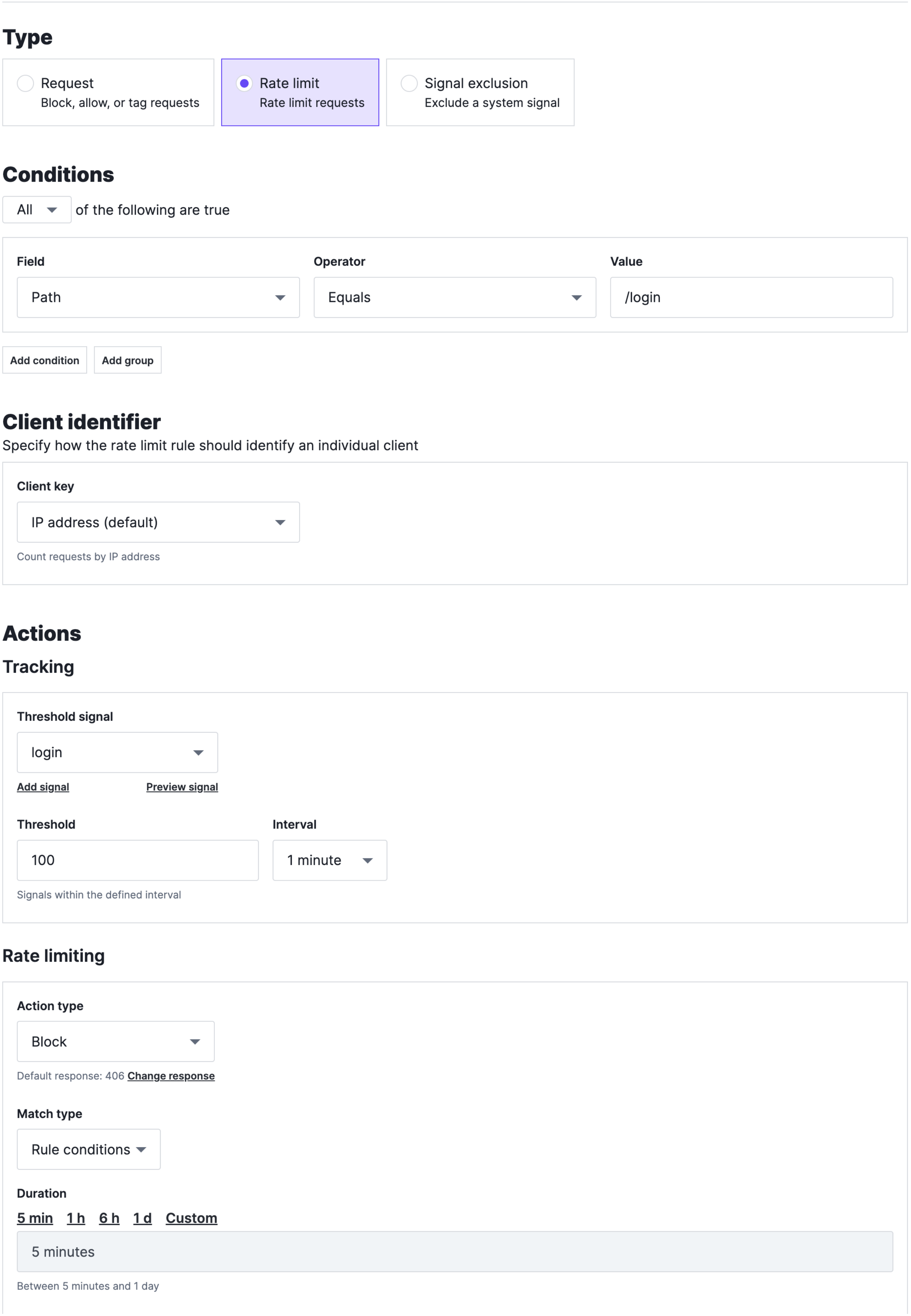
We store signal data based on the storage category of the signal.

Signal	Signal type	Storage category
Threshold signal	Informational	Time series only
`Rate Limit`	Anomaly	Sampled
`Blocked Request`	Anomaly	Sampled

Creating advanced rate limiting rules

To create an advanced rate limiting rule, start by navigating to the Add form for rules and selecting a type of rule:

1. On the **Site Rules** page, click the **Add site rule** button. The Add form appears.



Type

○ Request
Block, allow, or tag requests

● Rate limit
Rate limit requests

○ Signal exclusion
Exclude a system signal

Conditions

All of the following are true

Field	Operator	Value
Path	Equals	/login

Client identifier

Specify how the rate limit rule should identify an individual client

Client key: IP address (default)

Count requests by IP address

Actions

Tracking

Threshold signal: login

Threshold: 100

Interval: 1 minute

Signals within the defined interval

Rate limiting

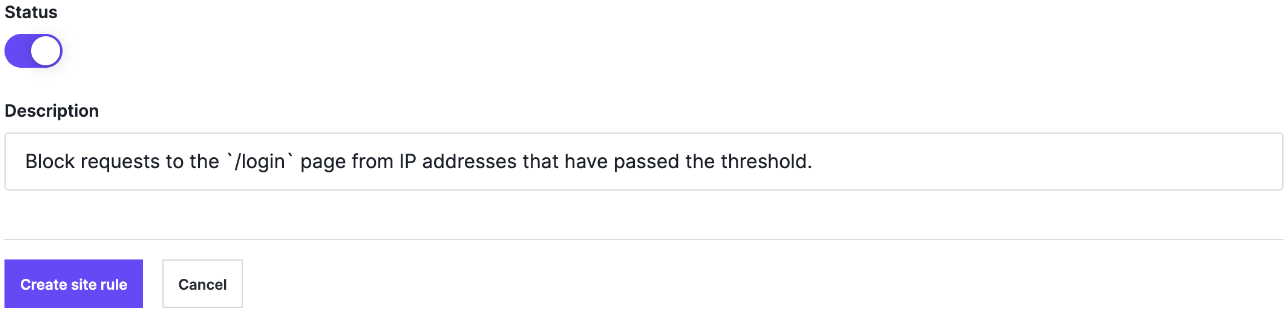
Action type: Block

Default response: 406 Change response

Match type: Rule conditions

Duration: 5 minutes

Between 5 minutes and 1 day



Status

Description

Block requests to the `/login` page from IP addresses that have passed the threshold.

2. In the **Type** section, select **Rate limit**. The controls for adding rate limit rules appear in the form.

Next, define the logic that the rule should use to identify requests that count towards the threshold. In the **Conditions** section:

1. Fill out these fields to create a condition:
   - From the **Field** menu, select the request field that the condition is based on.
   - From the **Operator** menu, select an operator to specify how the selected field and value relate.
   - In the **Value** field, enter a value for the specified field.
2. Optionally click the **Add condition** button to add another condition or the **Add group** button to create a group of conditions.
3. Decide whether a request must meet one or all conditions in order to count towards the threshold:
   - Select **Any** from the conditions menu to specify that a request must meet only one of the conditions you’ve created.
   - Leave **All** selected in the conditions menu to specify that a request must meet every condition you’ve created.

Once you’ve done that, specify how the rate limit rule should identify an individual client. In the **Client identifier** section:

1. From the **Client key** menu, select how the rate limit rule should identify a client. Depending on the Client key option you selected, additional client identifier fields may appear.
2. Optionally fill out any additional fields that appeared in the Client identifier section.

Next, specify a signal that should be applied to requests that meet the rule’s condition set and define the threshold. In the **Actions** section, fill out the **Tracking** subsection as follows:

1. From the **Threshold signal** menu, select the signal that you want applied to requests that match the rule conditions. Requests tagged with the threshold signal are tallied and counted towards the threshold of the rule.
2. In the **Threshold** field, enter the number of requests that must be detected before a client is rate limited.
3. From the **Interval** menu, select the period of time requests must be detected during to pass the threshold.

Next, define how a client that exceeds the threshold should be rate limited. In the **Actions** section, fill out the **Rate limiting** subsection as follows:

1. From the **Action type** menu, select the action that should be taken when the threshold has been exceed and requests meet the conditions of the match type. Action types include **Block** and **Allow**. When you select **Block**, the Change response link appears.
2. Optionally click the **Change response** link to specify a custom response code to return when the rule blocks a request and fill out the related fields as follows:
   - In the **Response code (optional)** field, enter a custom response code. Supported custom response codes are 301, 302, and 400-599.
   - If you entered `301` or `302` in the **Response code (optional)** field then, in the **Redirect URL (optional)** field, enter the absolute or relative URL of the redirect location. See Using redirect custom response codes.
3. From the **Match type** menu, select the conditions that determine what should be rate limited once the threshold is exceeded. Options include:
   - **Rule conditions:** rate limit requests from the client that match the rule’s conditions. See an example use case of this option.
   - **Other signal:** rate limit requests from the client that are tagged with the selected **Action signal**. When the action signal is not an attack or anomaly signal, you need a request rule that tags requests with the selected signal. See an example use case of this option.
   - **All requests:** rate limit all requests from the client.
4. From the **Duration** menu, select the amount of time the client should be rate limited.

Finally, add a description of the rule and save the rule:

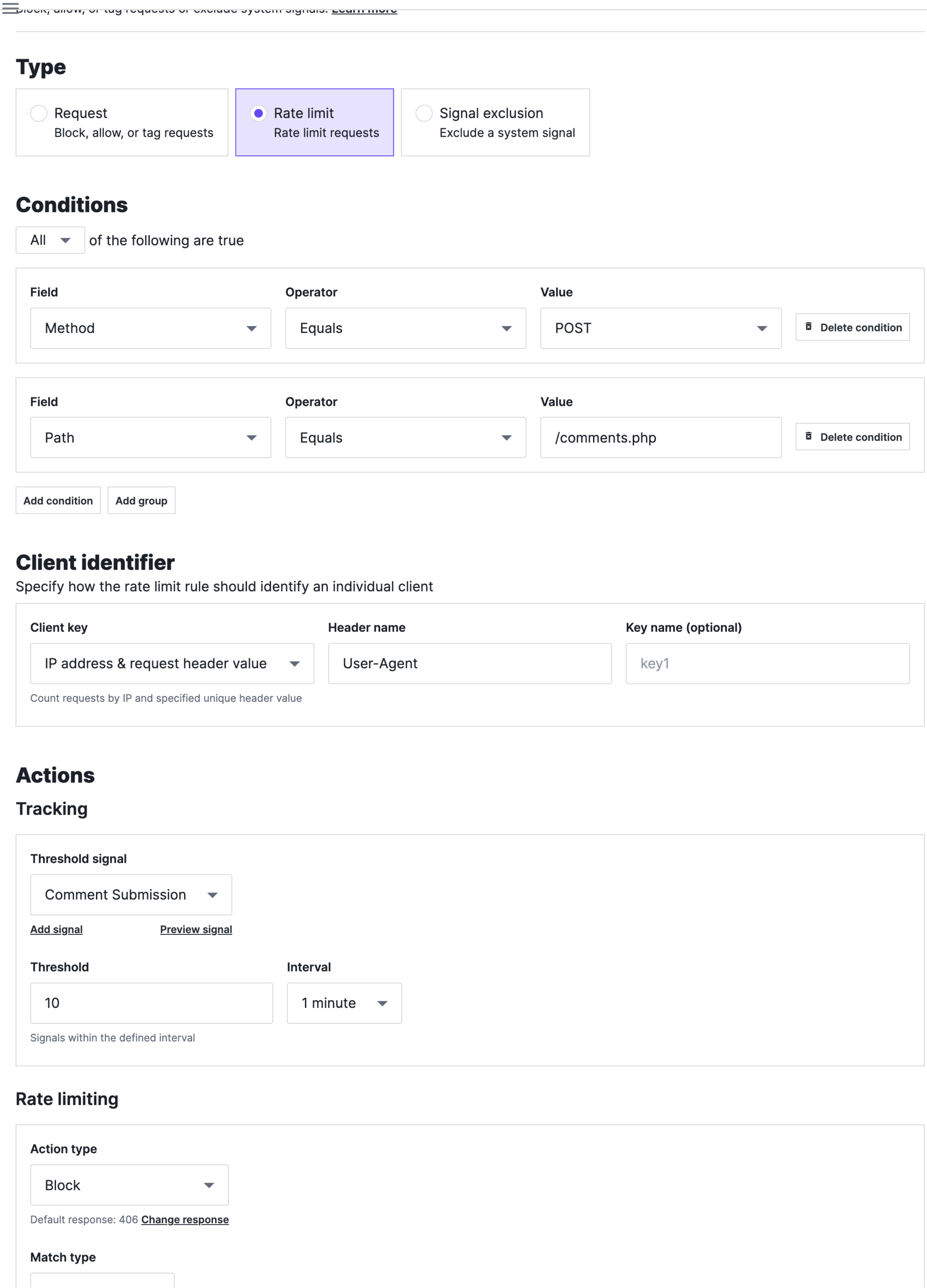
1. Fill out the **Details** section as follows:
   - Leave the **Status** switch enabled.

Example rate limit rules

The following examples demonstrate how to configure rate limit rules for common use-cases. Be aware that values (e.g., paths and response codes) used in these examples may not be the same as those used by your particular web application.

Rate limit comment submissions

This example rule demonstrates how to rate limit users' ability to submit comments:



Block, allow, or tag requests or exclude system signals. Learn more

Type

○ Request
Block, allow, or tag requests

● Rate limit
Rate limit requests

○ Signal exclusion
Exclude a system signal

Conditions

All of the following are true

Field	Operator	Value	
Method	Equals	POST	Delete condition
Path	Equals	/comments.php	Delete condition

Client identifier

Specify how the rate limit rule should identify an individual client

Client key	Header name	Key name (optional)
IP address & request header value	User-Agent	key1

Count requests by IP and specified unique header value

Actions

Tracking

Threshold signal: Comment Submission

Threshold: 10

Interval: 1 minute

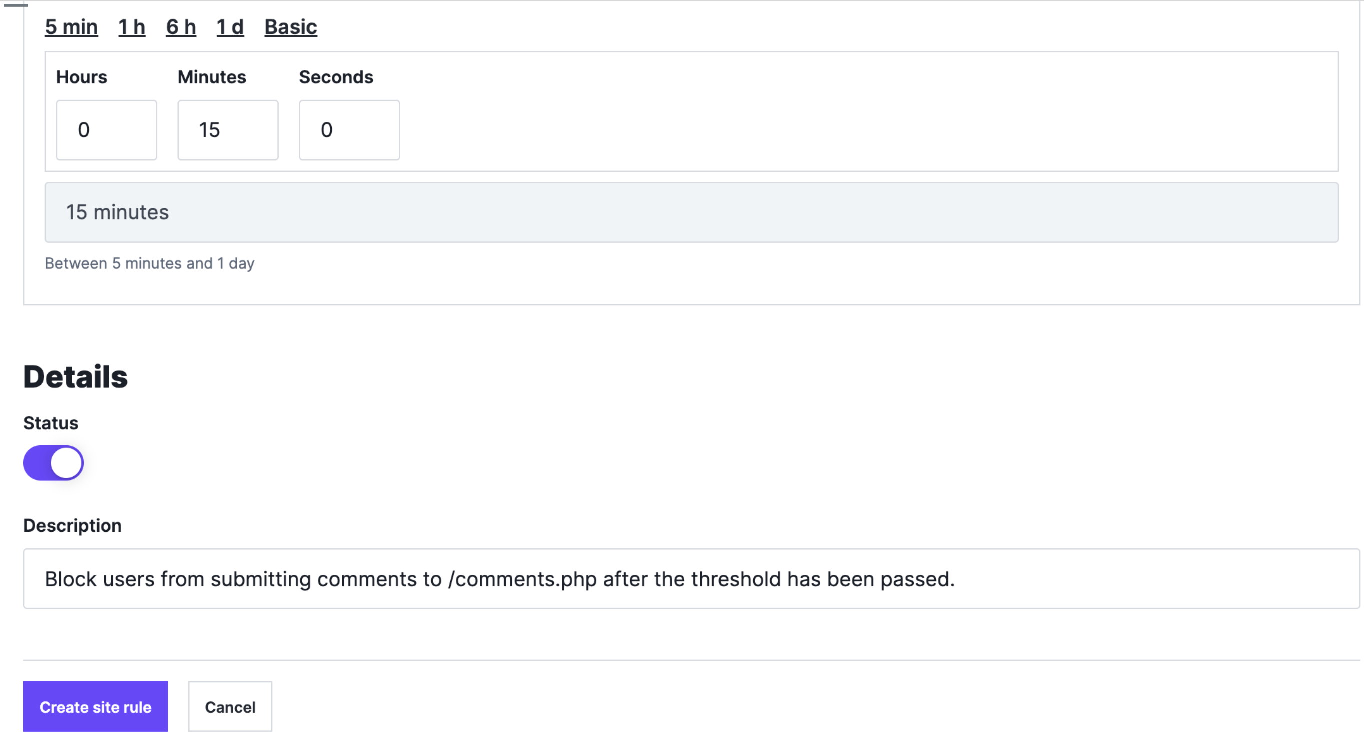
Signals within the defined interval

Rate limiting

Action type: Block

Default response: 406 Change response

Match type



5 min 1 h 6 h 1 d Basic

Hours	Minutes	Seconds
0	15	0

15 minutes

Between 5 minutes and 1 day

Details

Status

Description

Block users from submitting comments to /comments.php after the threshold has been passed.

Specifically, the rule looks for POST requests to the `/comments.php` file and tags them with the `Comment Submission` custom signal (the **Threshold signal**). Because the user may attempt to change their IP address to circumvent the rate limit, the rule uses both the IP address and the value of the `User-Agent` request header (the **Client identifier**) to track requests from this user.

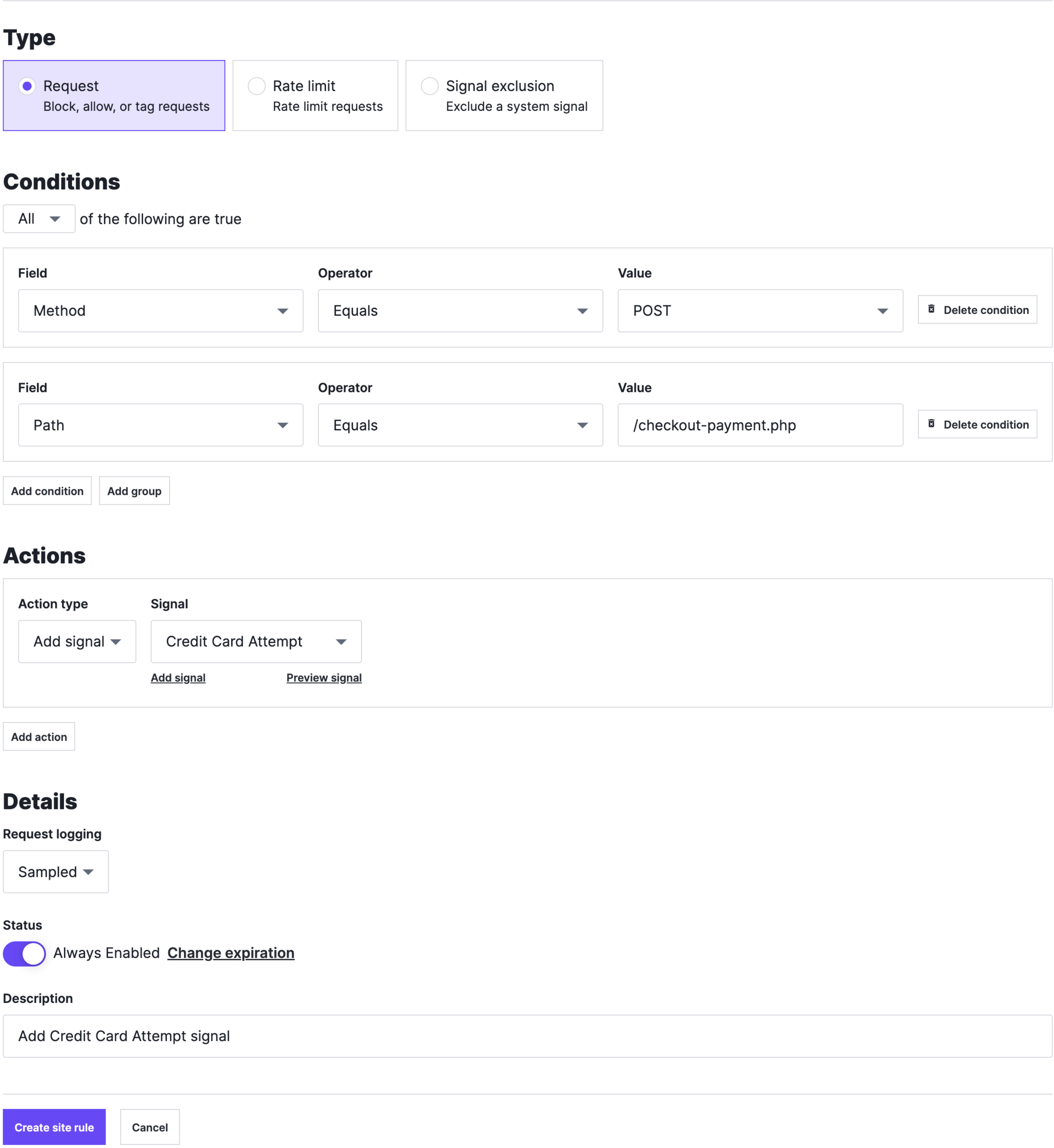
When 10 requests (the **Threshold**) tagged with the `Comment Submission` signal are detected from a unique IP address and `User-Agent` within 1 minute (the **Interval**), any subsequent requests with the `Comment Submission` signal from that IP address and `User-Agent` will be blocked (the **Action type**) for the next 15 minutes (the **Duration**).

Credit card validation attempts

This example demonstrates how to rate limit credit card validation attempts after too many failed attempts. This use-case requires two separate rules:

- a request rule to track credit card validation attempts.
- a rate limit rule to track credit card validation failures and rate limit the originating IP address.

The request rule looks for POST requests to the `/checkout-payment.php` file and tags them with the `Credit Card Attempt` custom signal.



Type

● Request
Block, allow, or tag requests

○ Rate limit
Rate limit requests

○ Signal exclusion
Exclude a system signal

Conditions

All of the following are true

Field	Operator	Value	
Method	Equals	POST	Delete condition
Path	Equals	/checkout-payment.php	Delete condition

Add condition Add group

Actions

Action type	Signal
Add signal	Credit Card Attempt

Add signal Preview signal

Add action

Details

Request logging

Sampled

Status

Always Enabled Change expiration

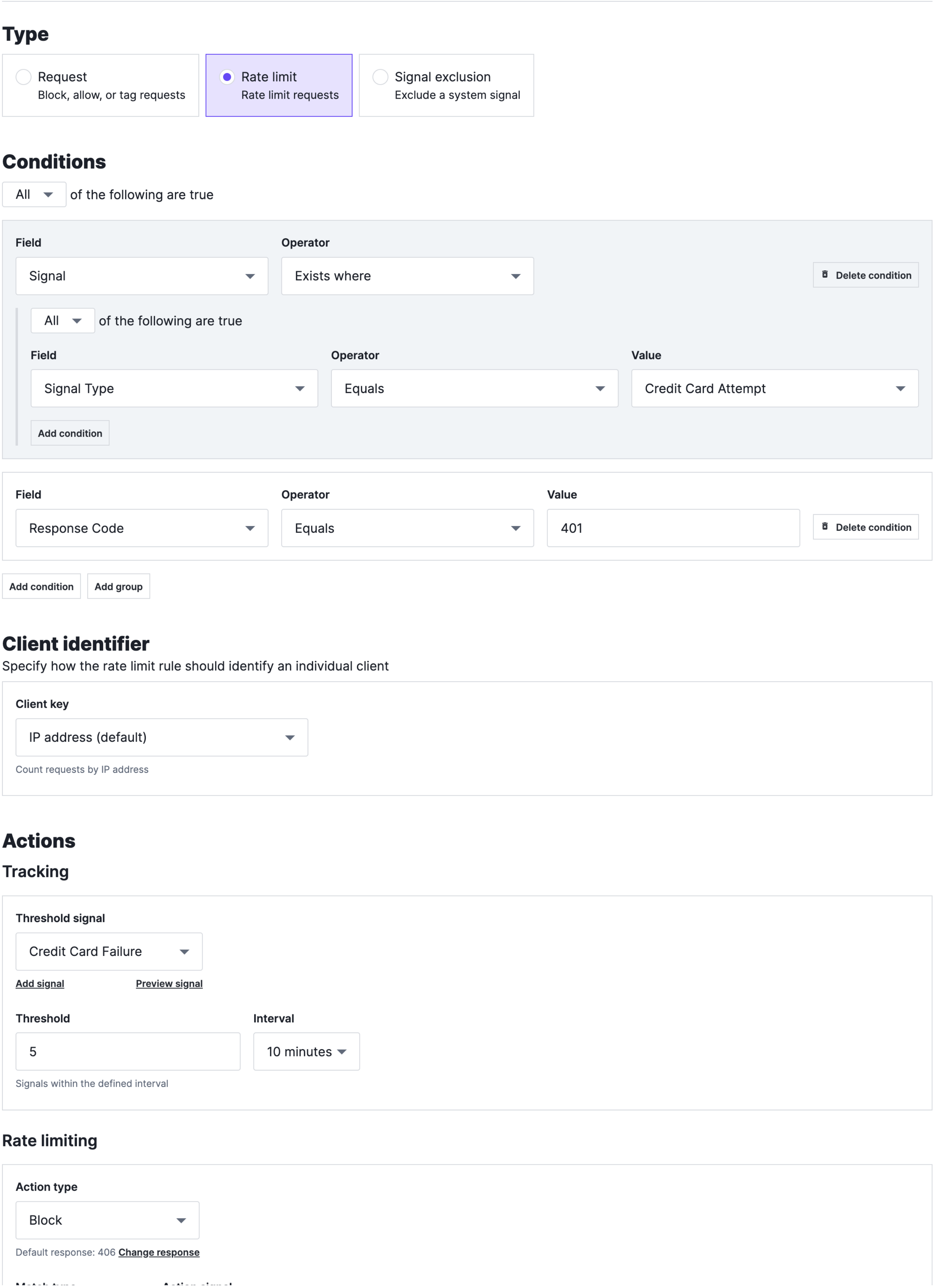
Description

Add Credit Card Attempt signal

Create site rule Cancel

The rate limit rule looks for requests tagged with the `Credit Card Attempt` custom signal, as well as if the request received a `401` response code indicating the credit card validation attempt was a failure. The rule applies a `Credit Card Failure` custom signal (the **Threshold signal**) to these requests.

When 5 requests (the **Threshold**) tagged with the `Credit Card Failure` signal are detected from a signal IP within 10 minutes (the **Interval**), any subsequent requests tagged with the `Credit Card Attempt` signal (the **Action signal**) from that IP will be blocked (the **Action type**) for the next hour (the **Duration**).



Type

○ Request
Block, allow, or tag requests

● Rate limit
Rate limit requests

○ Signal exclusion
Exclude a system signal

Conditions

All of the following are true

Field: Signal
Operator: Exists where

All of the following are true

Field: Signal Type
Operator: Equals
Value: Credit Card Attempt

Field: Response Code
Operator: Equals
Value: 401

Client identifier

Specify how the rate limit rule should identify an individual client

Client key: IP address (default)
Count requests by IP address

Actions

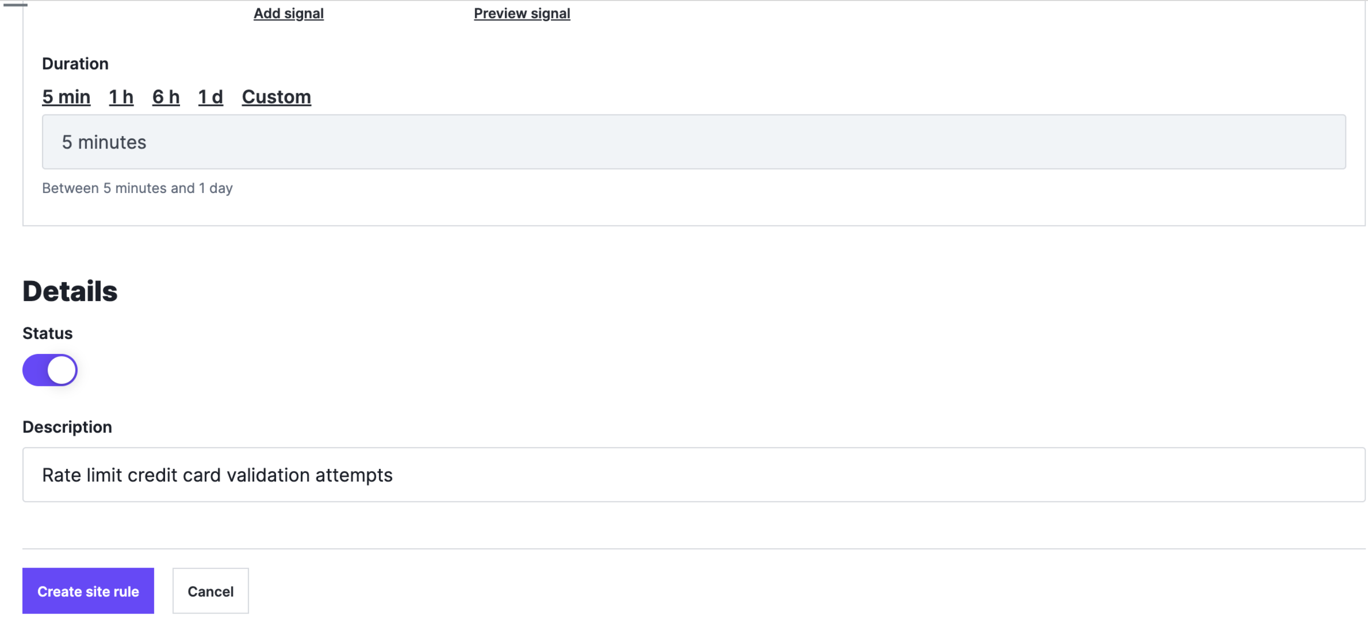
Tracking

Threshold signal: Credit Card Failure

Threshold: 5
Interval: 10 minutes
Signals within the defined interval

Rate limiting

Action type: Block
Default response: 406 Change response



Add signal Preview signal

Duration

5 min 1 h 6 h 1 d Custom

5 minutes

Between 5 minutes and 1 day

Details

Status

Description

Rate limit credit card validation attempts

Glossary

Term	Definition
Action type	Whether requests are logged or blocked.
Client	The source from where requests originate.
Client identifier	The parts of requests used to identify an individual client.
Duration	How long a client remains rate limited.
Interval	The period of time requests must be detected during to pass the threshold.
Match type	Which requests from the client should be blocked or logged after the threshold has been passed.
Threshold	How many requests must be detected before a client is rate limited.
Threshold signal	The signal that requests are tagged with when they meet the rate lime rule's condition set. Threshold signals are tallied and counted towards the threshold for that rule. When the threshold signal count for a single client exceeds the rule's threshold, that client is rate limited.

Upgrading the IIS Module

Check the IIS Module Changelog to see what's new in the IIS module.

Upgrading the IIS Module

The process for upgrading the IIS module is the same as installing the IIS Module with the latest release.

1. **Upgrade the IIS Module via MSI Package (recommended)**

   Download the latest IIS module MSI: IIS Module MSI

   Follow the MSI install instructions.

2. **Upgrade the IIS Module via MSI from a previous ZIP install (recommended if running ZIP install)**

   Download the latest IIS module MSI: IIS Module MSI

   Follow the ZIP to MSI upgrading instructions.

3. **Upgrade the IIS Module via ZIP Archive**

   Download the latest IIS module ZIP archive: IIS Module MSI

   Follow the ZIP install instructions.

pages:

- Sites
- Corp Users
- User Authentication
- API Access Tokens
- Corp Integrations
- Corp Audit Log
- Cloud WAF Certificates
- Could WAF Instances

Before you begin

Be sure you know how to access the web interface controls.

About the Sites page

Selecting **Sites** from the Corp Manage menu displays the Sites page. From this page, you can manage your sites. A site is a single web application, bundle of web applications, API, or microservice that Signal Sciences can protect from attacks.

The Sites page lists existing sites and provides controls to filter the list. Specifically, you can filter the list by:

- **Site name:** allows you to filter sites by name.
- **Agent mode:** allows you to filter sites by agent mode. Agent mode options include blocking, not blocking, and off.

When you click the name of a site, the Site Settings page appears. From the Site Settings page, you can update the name of the site, the agent configurations for the site, and the users who have access to the site.

From the Sites page, you can also add a site by clicking the **Add site** button.

About the Corp Users page

Selecting **Corp Users** from the Corp Manage menu displays the Corp Users page. From this page, you can manage your users. Users are the people who manage, edit, or observe activity in your corp.

The Corp Users page lists existing users and provides controls to filter the list. Specifically, you can filter the list by:

- **Site:** allows you to filter users by site.
- **User:** allows you to filter users by user name, email, or role.
- **Role:** allows you to filter users by user role. User roles include owner, admin, user, and observer.
- **Status:** allows you to filter users by whether they are active or pending.
- **2FA:** allows you to filter users by whether they have two factor authentication enabled or disabled.

When you click the name of a user, a details page for the user appears. From this page, you can update the user's information or delete the user.

From the Corp Users page, you can also add users by clicking the **Add corp user** button.

About the User Authentication page

Selecting **User Authentication** from the Corp Manage menu displays the User Authentication page. From this page, you can:

- set up a user authentication method. Methods include password-based authentication, SAML Single Sign-On, and Google Apps Single Sign-On.
- configure Okta to be your identity provider (IdP).
- specify the duration of a validated session for a user.
- select whether to allow or restrict all users from creating and using API access tokens. When token creation and usage is restricted, you can enable specific users to create and use API Access tokens.
- specify when API access tokens expire.
- opt out of specific features if you are enrolled in Fastly Security Labs. Fastly Security Labs is a program that grants your corp access to in-development beta features.

  NOTE: Fastly Security Labs is only included with the Professional and Premier platforms. It is not included as part of the Essential platform.

About the API Access Tokens page

columns:

- **Created by:** the user name of the person who created the token.
- **Token Name:** the friendly name of the token.
- **Logged IP:** the IP address of the request.
- **User Agent:** the user agent of the request.
- **Timestamp:** the date the token was used.
- **Status:** the status of the token. Status options include active, expired, and disabled by owner.
- **Expires:** the date the token expires.

From the API Access Tokens page, you can also delete tokens.

About the Corp Integrations page

Selecting **Corp Integrations** from the Corp Manage menu displays the Corp Integrations page. Corp integrations notify you about activity within your corp, including changes to users, sites, and settings. There are three types of corp integrations:

- Mailing Lists
- Microsoft Teams
- Slack

From the Corp Integrations page, you can:

- manage existing corp integrations by clicking on the name of an existing integration.
- add a new corp integration by clicking the **Add corp integration** button.

About the Corp Audit Log page

Selecting **Corp Audit Log** from the Corp Manage menu displays the Corp Audit Log page. This page lists corp activity from the last 30 days. You can filter the list by activity type. Types include user activity (e.g., a user created a new access token) and corp configuration (e.g., a user created a new site).

The Corp Audit Log page lists connected corp integrations in a side bar. You can click on an integration to view details about that integration or click **Manage corp integrations** to go to the Corp Integrations page.

About the Certificates page

Selecting **Cloud WAF Certificates** from the Corp Manage menu displays the Certificates page. From the Certificates page, you can:

- view a summary table that lists the TLS/SSL certificates for your sites.
- use a search bar to filter the list.
- access a detailed view of a certificate by clicking the **View** link to the right of the certificate. Additional controls on the detailed view page enable you to edit or delete the certificate.
- create a certificate by clicking **Add certificate**.

About the Cloud WAF Instances page

Selecting **Cloud WAF Instances** from the Corp Manage menu displays the Cloud WAF Instances page. From this page, you can use the **Add Cloud WAF Instance** button to create an instance, the Instances tab to manage existing instances, and the Routes tab to manage existing routes.

From the Instances tab, you can:

- view a summary table that lists up to 20 Cloud WAF instances running on your corp.
- use a search bar to filter the instances.
- access a detailed view of an instance by clicking the **View** link to the right of an instance. Additional controls on the detailed view page enable you to edit, delete, or re-deploy the instance.

From the Routes tab, you can:

- view a summary table that lists your routes.
- use a search bar to filter the routes.
- access a detailed view of the instance associated with a route by clicking the **View** link to the right of a route. Additional controls on the detailed view page enable you to edit, delete, or re-deploy the instance.

HashiCorp Vault

Specifically, the plugin allows:

- Vault to store the Agent Access Keys and Agent Secret Keys for your sites.
- the Vault agent to pull the keys from Vault when needed and give the keys to the deployed Signal Sciences agent.
- Vault to rotate or replace site keys. When Vault replaces keys, the Vault agent updates the configuration file for the relevant Signal Sciences agent and restarts the Signal Sciences agent.
- authenticated applications, services, and machines to read site keys that are stored in Vault.

Important: This information is part of a beta release. For additional details, read our product and feature lifecycle descriptions.

Limitations and considerations

Before setting up the plugin and managing site keys in Vault, keep the following in mind:

- To use the Signal Sciences plugin for HashiCorp Vault, Vault must already be installed and configured to load external plugins.
- The Signal Sciences agent is restarted as part of the key rotation process. Due to the agent's brief downtime during key rotation, we recommend rotating the keys during a maintenance window.

Set up plugin

To set up the plugin for the first time, follow these steps:

1. Using the curl command line tool, copy the plugin binary to the external plugins directory:

   ```
   curl -O https://dl.signalsciences.net/vault-plugin-sigsci/latest/vault-plugin-sigsci.tar.gz
   tar xzvf vault-plugin-sigsci.tar.gz
   vault plugin register -sha256=$(sha256sum vault-plugin-sigsci|cut -c-64) secret vault-plugin-sigsci
   ```

2. Using the command line, enable the plugin:

   ```
   vault secrets enable -path=sigsci vault-plugin-sigsci
   ```

   Vault mounts the plugin at path `/sigsci`.

3. Create a user for the plugin. Assign the user the **User** role. An invitation email is sent to the email address you supplied for the plugin user.

4. From the plugin user's email account, open the invitation email and click **Accept invite**. The account creation form appears.

5. Fill out the account creation form:

   - Leave the **Email address** field as is.
   - In the **Name** field, enter `vault-user`.
   - In the **Password** field, enter a password for the account.
   - In the **Confirm password** field, enter the password again.
6. Click **Create account**.

7. Create an API access token for the plugin user. Signal Sciences cloud API credentials are required for reading and managing agent site keys.

8. Using the command line, copy the API access token to `token.txt` file:

   ```
   vault write -f /sigsci/role/vault-user corp=<corp-id> email=<email-id> token=@token.txt
   ```

   Replace `<corp-id>` with the ID of your corp and `<email-id>` with the plugin user's email address.

9. Using the command line, copy site keys for a single site or all sites to vault:

   ```
   vault write -f /sigsci/creds/vault-user/sites/<site-name>
   ```

   Replace `<site-name>` with the name of the site.

   or

   ```
   vault write -f /sigsci/creds/vault-user/sites/
   ```

10. Install and configure the Vault agent using the following template:

```
 }
```

The Vault agent automates the rendering of the Signal Sciences agent configuration template when the site keys are rotated.

Example content of the configuration template `/etc/signalsciens/agent.ctmpl`:

```
{{ with secret "sigsci/creds/vault-user/sites/<site-name>" }}
accesskeyid={{ .Data.accessKey }}
secretkey={{ .Data.secretKey }}
{{ end }}
```

11. Using the command line, create a systemd service to restart the agent:

```
 sudo tee -a /etc/systemd/system/sigsci-agent-restart.service <<END
[Unit]
Description="signalsciences agent restarter"

[Service]
Type=OneShot
ExecStart=/usr/bin/systemctl restart sigsci-agent.service

[Install]
WantedBy=multi-user.target
END
```

12. Using the command line, create a configuration file watcher:

```
 sudo tee -a /etc/systemd/system/sigsci-agent-restart.path <<END
[Path]
PathChanged=/etc/signalsciens/agent.conf

[Install]
WantedBy=multi-user.target
END
```

13. Using the command line, start and enable the configuration file watcher:

```
 systemctl enable --now sigsci-agent-restart.service
```

Rotate site keys

To rotate the keys for a site, replace the keys in Vault, restart the Signal Sciences agent, and then delete the non-primary keys in Vault:

1. Using the command line, rotate a site key in Vault:

```
 vault write -f /sigsci/rotate/sites/<site-name>
```

Replace `<site-name>` with the name of the relevant site.

2. Using the command line, delete the non-primary keys in Vault:

```
vault delete /sigsci/rotate/sites/<site-name>
```

Replace `<site-name>` with the name of the relevant site.

Manage plugin roles and keys

Once the plugin is set up, you can use the command line to perform these actions:

Action	Command
List roles	`vault read /sigsci/role/`
Read role details	`vault read /sigsci/role/vault-user`
Delete role	`vault delete /sigsci/role/vault-user`
Copy keys for one site to Vault	`vault write -f /sigsci/creds/vault-user/sites/<site-name>`

Rotate keys for a site	`vault write -f /sigsci/rotate/sites/<site-name>`
List keys for all sites	`vault read /sigsci/creds/vault-user/sites/`
Read keys for one site	`vault read /sigsci/creds/vault-user/sites/<site-name>`
Delete the non-primary keys for a site from Vault	`vault delete /sigsci/rotate/sites/<site-name>`
Delete the keys for a site from Vault	`vault delete /sigsci/creds/vault-user/sites/<site-name>`

JIRA

Our JIRA issue integration creates an issue when IP addresses are flagged on Signal Sciences.

Adding a JIRA issue integration

JIRA issue integrations are configured per project.

1. Create a new user in JIRA for the integration to use.
2. Create an API token for that user.
3. Log in to the Signal Sciences console.
4. From the **Sites** menu, select a site if you have more than one site.
5. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
6. Click **Add site integration**. The add site integration menu page appears.
7. Select the **Jira Issue** integration. The JIRA integration setup page appears.
8. In the **Host** field, enter the URL of your JIRA instance.
9. In the **Username** field, enter the username you created in JIRA.
10. In the **API Token** field, enter the API token you created in JIRA.
11. In the **Project Key** field, enter the key of the JIRA project to create new issues in.
12. In the **Issue Type** field, enter the type of issue that should be created.
13. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
14. Click **Create site integration**.

Activity types

Activity type	Description
`flag`	An IP address was flagged
`agentAlert`	An agent alert was triggered

Working with request rules

Request rules allow you to define arbitrary conditions and block, allow, or tag requests indefinitely or for a specific period of time. For example, you could make a rule to block all requests with specific headers, requests to certain paths, or requests originating from specific IP addresses.

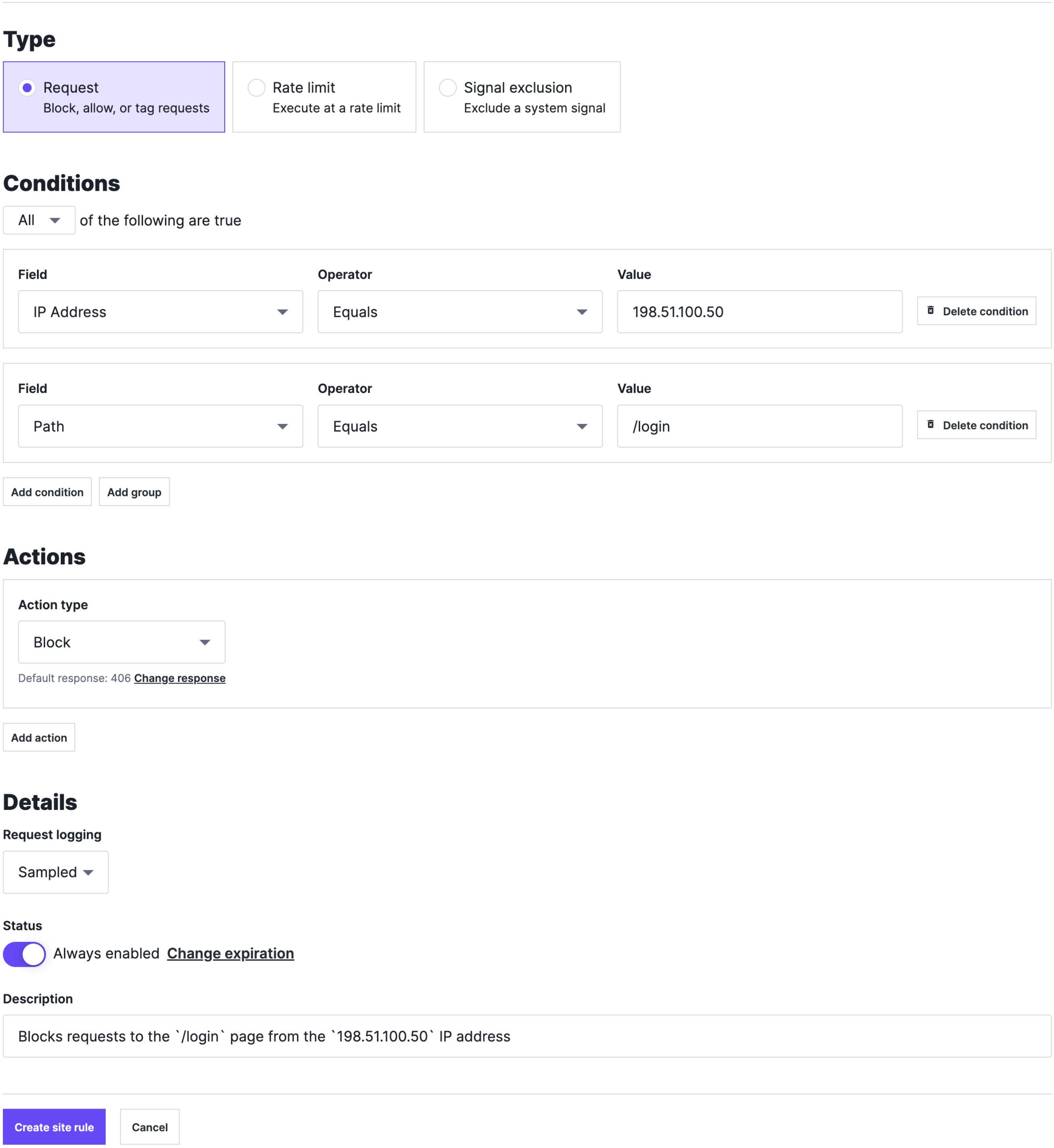
Limitations and considerations

Request rules are limited to 1000 per corp plus 1000 per site.

Creating request rules

To create a request rule, follow these steps:

1. On the **Site Rules** page, click the **Add site rule** button. The Add form appears.



Type

● Request
Block, allow, or tag requests

○ Rate limit
Execute at a rate limit

○ Signal exclusion
Exclude a system signal

Conditions

All of the following are true

Field	Operator	Value	
IP Address	Equals	198.51.100.50	Delete condition
Path	Equals	/login	Delete condition

Add condition Add group

Actions

Action type: Block

Default response: 406 Change response

Add action

Details

Request logging: Sampled

Status: Always enabled Change expiration

Description: Blocks requests to the `/login` page from the `198.51.100.50` IP address

Create site rule Cancel

2. In the Type section, select **Request**.

3. Fill out the fields in the Conditions section as follows:

   - From the **Field** menu, select the request field that the condition is based on.
   - In the **Value** field, enter a value for the specified field.
   - From the **Operator** menu, select an operator to specify how the selected field and value relate.
   - Optionally click the **Add condition** button to add another condition or the **Add group** button to create a group of conditions.
   - Select **All** to specify that a request must meet every condition or **Any** to specify that a request must meet only one condition.

4. Fill out the fields in the Actions section as follows:

   - From the **Action type** menu, select the action that should be taken when a request meets the rule's conditions. Action types include Block, Allow, and Add signal.

  - If you entered `301` or `302` in the **Response code (optional)** field then, in the **Redirect URL (optional)** field, enter the absolute or relative URL of the redirect location. See Using redirect custom response codes.
  - Optionally click the **Add action** button to add another action.
5. Fill out the fields in the **Details** section as follows:
  - From the **Request logging** menu, select **Sampled** to store the logs for requests that match the rule's criteria and **None** to not store the logs. When you select **None**, the time series graphs will still include data from requests that match the rule's criteria. See Request data storage for more information.
  - Leave the **Status** switch enabled.
  - Click the **Change expiration** link and select from the menu when the rule should be disabled.
  - In the **Description** field, enter a description of the rule.
6. Click the **Create site rule** button. The request rule is created, and the Site Rules page appears.

Mailing List

Our mailing list integration allows you to receive email notifications for certain activity on Signal Sciences.

Adding a mailing list integration

Corp integration

> **Note:** Only Owners can create, edit, and delete corp integrations.

1. Log in to the Signal Sciences console.
2. From the **Corp Manage** menu, select **Corp Integrations**. The corp integrations menu page appears.
3. Click **Add corp integration**. The add corp integration menu page appears.
4. Select the **Mailing List** integration. The mailing list integration setup page appears.
5. In the **Email address** field, enter the email address or alias to send alerts to.
6. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
7. Click **Create corp integration**.

Site integration

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
4. Click **Add site integration**. The add site integration menu page appears.
5. Select the **Mailing List** integration. The mailing list integration setup page appears.
6. In the **Email address** field, enter the email address or alias to send alerts to.
7. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
8. Click **Create site integration**.

Activity types

Corp

Activity type	Description
`releaseCreated`	New release notifications
`featureAnnouncement`	New feature announcements
`corpUpdated`	Account timeout setting updated
`newSite`	A new site was created
`deleteSite`	A site was deleted
`enableSSO`	SSO was enabled for the corp
`disableSSO`	SSO was disabled for the corp
`corpUserInvited`	A user was invited
`corpUserReinvited`	A user was reinvited
`listCreated`	A list was created
`listUpdated`	A list was updated

`customTagCreated`	A custom signal created
`customTagDeleted`	A custom signal updated
`customTagUpdated`	A custom signal removed
`userAddedToCorp`	A user was added to the corp
`userMultiFactorAuthEnabled`	A user enabled 2FA
`userMultiFactorAuthDisabled`	A user disabled 2FA
`userMultiFactorAuthUpdated`	A user updated 2FA secret
`userRegistered`	A user was registered
`userRemovedCorp`	A user was removed from the corp
`userUpdated`	A user was updated
`userUndeliverable`	A user’s email address bounced
`userUpdatePassword`	A user updated their password
`accessTokenCreated`	An API Access Token was created
`accessTokenDeleted`	An API Access Token was deleted

Site

Activity type	Description
`siteDisplayNameChanged`	The display name of a site was changed
`siteNameChanged`	The short name of a site was changed
`loggingModeChanged`	The agent mode (“Blocking”, “Not Blocking”, “Off”) was changed
`agentAnonModeChanged`	The agent IP anonymization mode was changed
`flag`	An IP address was flagged
`expireFlag`	An IP address flag was manually expired
`createCustomRedaction`	A custom redaction was created
`removeCustomRedaction`	A custom redaction was removed
`updateCustomRedaction`	A custom redaction was updated
`customTagCreated`	A custom signal was created
`customTagUpdated`	A custom signal was updated
`customTagDeleted`	A custom signal was removed
`customAlertCreated`	A custom alert was created
`customAlertUpdated`	A custom alert was updated
`customAlertDeleted`	A custom alert was removed
`detectionCreated`	A templated rule was created
`detectionUpdated`	A templated rule was updated
`detectionDeleted`	A templated rule was removed
`listCreated`	A list was created
`listUpdated`	A list was updated
`listDeleted`	A list was removed
`ruleCreated`	A request rule was created
`ruleUpdated`	A request rule was updated
`ruleDeleted`	A request rule was deleted
`customDashboardCreated`	A custom dashboard was created
`customDashboardUpdated`	A custom dashboard was updated
`customDashboardReset`	A custom dashboard was reset
`customDashboardDeleted`	A custom dashboard was removed
`customDashboardWidgetCreated`	A custom dashboard card was created
`customDashboardWidgetUpdated`	A custom dashboard card was updated
`customDashboardWidgetDeleted`	A custom dashboard card was removed

`weeklyDigest`	Weekly digest sent

Managing users

If you have an owner or admin role, you can manage the users in your corp.

Managing users as an owner

Owners can view and manage all users on the corp by going to the **Corp Manage** menu and selecting **Corp Users**. This page lists all the users in the corp, along with their roles, site memberships, and whether they have 2FA enabled, as well as the list of pending invited users.

Adding users

To add a new user, complete the following steps:

1. Click **Add corp user**. The add corp user menu page appears.
2. In the **Email** field, enter the user's email address.
3. In the **Role** section, select which role the user should have.
4. In the **Site memberships** section, select which sites the user should be a member of. A user must belong to at least one site.
5. Click **Invite user**.

When the user is invited, they'll receive an email to register an account. They must click the **Accept invite** button at which point they'll be prompted to set their account password. After creating their account, they will then have access to all the sites they're a member of. The invitation is valid for 3 days. If the invitation is expired, resend the invite by clicking the pending user's row and clicking the **Resend Invite** button from the User Edit page.

Editing users

To edit a user, complete the following steps:

1. In the list of users, click on the user.
2. Click **Edit corp user**. The edit corp user page appears.
3. Edit the **Role** and **Site memberships** sections as needed.
4. Click **Update user**.

Deleting users

To delete a user, complete the following steps:

1. In the list of users, click on the user.
2. Click **Remove corp user**. The remove corp user page appears.
3. Click **Delete corp user**.

Disabling 2FA for a user

To disable two-factor authentication (2FA) for a user, complete the following steps:

1. In the list of users, click on the user.
2. Click **Edit corp user**. The edit corp user page appears.
3. Click **Disable 2FA**. A confirmation window appears.
4. Click **Yes, disable**.

The user will then be able to sign into their account without needing to authenticate through 2FA.

Auditing two-factor authentication

In the filters to the left of the list of users, select **Enabled** in the **2FA** section. This filters the list of users to only contain users who have two-factor authentication enabled.

We don't currently support 2FA enforcement.

Single sign-on

Check out Setting up single sign-on for more information about enabling Single Sign-On.

Bypassing SSO

Select **Allow this user to bypass Single Sign-On (SSO)** to set the user to bypass SSO.

API access tokens

Check out Using Our API for information about personal API access tokens.

Managing users as an admin

Admins have limited user management abilities for any sites they are a member of.

Invite new users to a site

To invite new users to a site, complete the following steps:

1. From the **Manage** menu, select **Site Settings**. The site settings menu page appears.
2. Click **Users**. The users tab appears.
3. From the **Manage site users** menu, select **Invite new user**. The user invitation menu page appears.
4. In the **Email** field, enter the user's email address.
5. In the **Role** section, select which role the user should have.
6. Click **Invite site user**.

When the user is invited, they'll receive an email to register an account. They must click the **Accept invite** button at which point they'll be prompted to set their account password. After creating their account, they will then have access to all the sites they're a member of. The invitation is valid for 3 days. If the invitation is expired, resend the invite by clicking the pending user's row and clicking the **Resend Invite** button from the User Edit page.

Assign existing users to a site

To assign existing users to a site, complete the following steps:

1. From the **Manage** menu, select **Site Settings**. The site settings menu page appears.
2. Click **Users**. The users tab appears.
3. From the **Manage site users** menu, select **Assign existing users**. The assign users menu page appears.
4. From the menu, select a user to add to the site.
5. Click **Assign to site**.

Remove users from a site

To remove users from a site, complete the following steps:

1. From the **Manage** menu, select **Site Settings**. The site settings menu page appears.
2. Click **Users**. The users tab appears.
3. In the list of users, click on the user.
4. Click **Remove site user**. The remove user confirmation page appears.
5. Click **Remove user**.

All users must belong to at least one site. If this is the only site the user is a member of, you will not be able to remove the user. Instead, an Owner user will need to delete the user entirely.

Console timeout

The default duration for a validated session is 30 days. To set a custom duration your corp:

1. Log in to the Signal Sciences console.
2. From the **Corp Manage** menu, select **User Authentication**. The User Authentication page appears.
3. Under **Account Timeout**, click on a pre-set duration or click **Custom** to specify a custom duration. If selecting **Custom**, enter the custom duration in the **Days**, **Hours**, **Minutes**, and **Seconds** fields.
4. Click **Update Timeout** to save the new timeout duration.

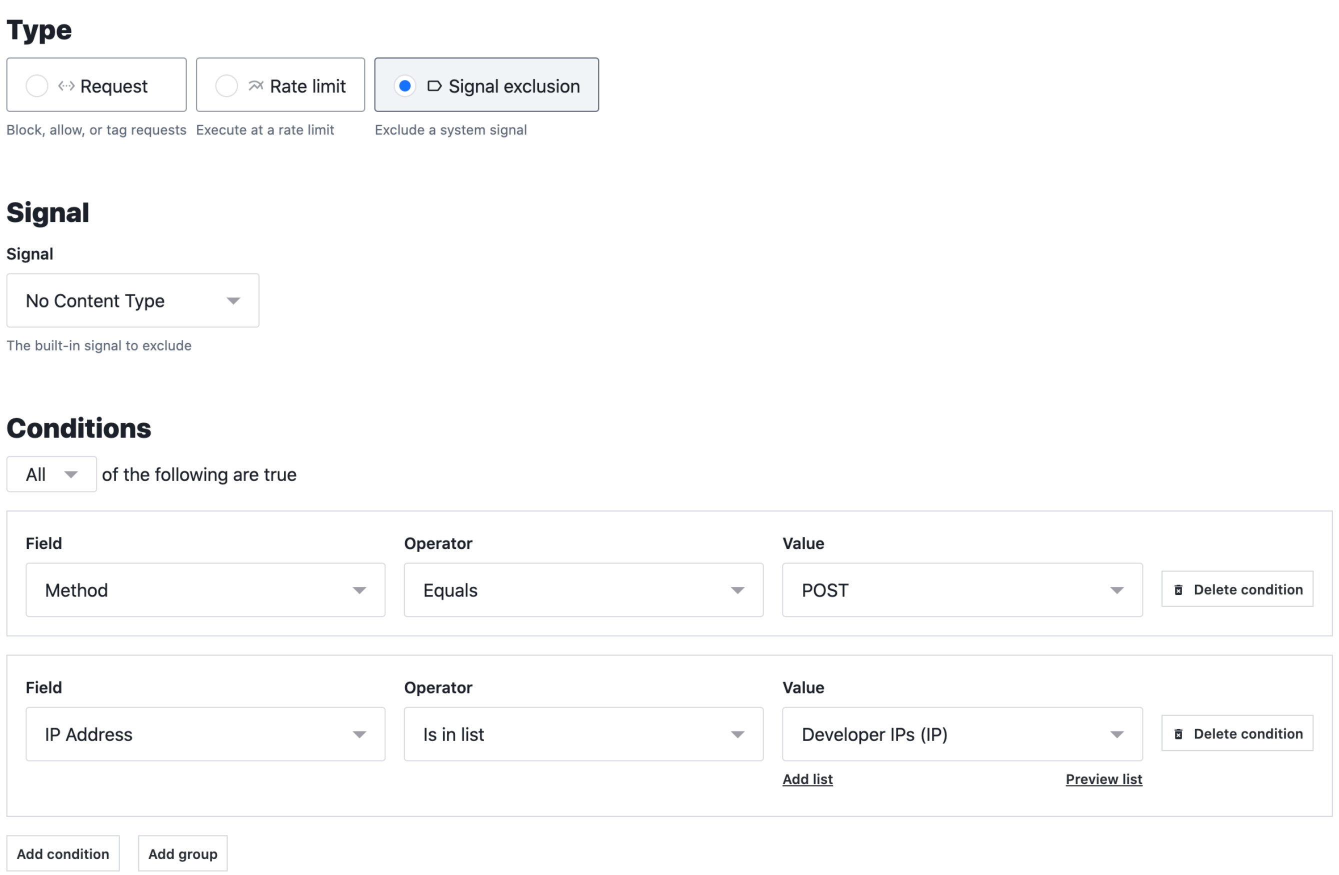
Working with signal exclusion rules

Signal exclusion rules allow you to define arbitrary conditions to exclude a specific system signal.

The below example signal exclusion rule prevents `POST` requests originating from a list of known internal IP addresses from being tagged with the `NO-CONTENT-TYPE` signal.

- The **Signal** is set to **No Content Type**.

selected for the **Value.**



Type

○ Request — Block, allow, or tag requests
○ Rate limit — Execute at a rate limit
● Signal exclusion — Exclude a system signal

Signal

Signal: No Content Type

The built-in signal to exclude

Conditions

All of the following are true

Field	Operator	Value
Method	Equals	POST
IP Address	Is in list	Developer IPs (IP)

Limitations and considerations

Signal exclusion rules are limited to 1000 per corp plus 1000 per site and count against the total number of request rule limits for corps and sites.

Monitoring account activity with audit logs

Activity across your corp and sites over the last 30 days is tracked and available to review in the audit logs. There are two different audit logs available: the Corp Audit Log for corp-level activity and the Site Audit Log for site-level activity. These logs can also be filtered by type of activity to more easily identify specific events.

Email notifications and integrations with third-party applications can be set up to automatically notify you of activity within your corp and sites. For additional information, see Integrations.

Corp Audit Log

The Corp Audit Log tracks activity related to your corp itself, such as the creation of new users and sites.

You can view the Corp Audit Log by going to the **Corp Manage** menu and selecting **Corp Audit Log**.

Activity types

The corp activity types that are logged include:

Activity type	Description
User invited	A new user was invited to the corp
User re-invited	The invitation email was re-sent to an invited user
User updated	A user was edited, including changes to user role
User password updated	A user updated their password
User added to site	A user was added to one or more sites
User removed from site	A user was removed from one or more sites

undeliverable	
User removed from corp	A user was deleted
User SSO exemption changed	A user's ability to bypass Single Sign-On (SSO) was changed
Corp integration created	A new corp-level integration was created
Corp integration updated	A corp-level integration was updated
Corp integration removed	A corp-level integration was removed
Corp integration tested	A corp-level integration was tested
Two-factor authentication enabled	A user enabled two-factor authentication (2FA)
Two-factor authentication updated	A user updated their two-factor authentication (2FA) secret
Two-factor authentication disabled	A user disabled two-factor authentication (2FA)
SSO enabled	Single Sign-On (SSO) was enabled for the corp
SSO disabled	Single Sign-On (SSO) was disabled for the corp
Site created	A new site was created
Site deleted	A site was deleted
User authentication setting updated	A user authentication setting was changed, including the account timeout setting, API access token creation permission and expiration settings, and restrictions of which IP addresses can access the console
API access token created	An API Access Token was created
API access token deleted	An API Access Token was deleted
SAML request certificate created	A new SAML request certificate was created
CloudWAF corp SSL certificate uploaded	An SSL certificate for CloudWAF was uploaded to the corp
CloudWAF corp SSL certificate deleted	An SSL certificate for CloudWAF was deleted from the corp
CloudWAF instance created	A new CloudWAF instance was created
CloudWAF instance updated	A CloudWAF instance was updated
CloudWAF instance deleted	A CloudWAF instance was deleted
CloudWAF certificate about to expire	A CloudWAF certificate is about to expire. Includes certificate ID and expiration date

Site Audit Log

The Site Audit Log tracks activity related to your individual sites. This includes activity such as flagged IPs, the creation of new rules, and site configuration changes.

You can view the Site Audit Log by going to the **Manage** menu and selecting **Site Audit Log**.

Activity types

The site activity types that are logged include:

Activity type	Description
Site display name changed	The display name of a site was changed
Site short name changed	The short name of a site was changed
Agent mode changed	The agent mode ("Blocking", "Not Blocking", "Off") was changed
Agent IP anonymization mode changed	The agent IP anonymization mode was changed
Client IP Header changed	A header used to determine the client IP address was changed
IP flagged	An IP address was flagged
IP flag expired	An IP address flag was manually expired
New agent online	A new agent was detected
Site integration created	A new site-level integration was created

Site integration removed	A site-level integration was removed
Site integration tested	A site-level integration was tested
Agent key created	A new agent key was created
Agent key deleted	An agent key was deleted
Primary agent key changed	The primary agent key was changed
Custom redaction created	A custom redaction was created
Custom redaction updated	A custom redaction was updated
Custom redaction removed	A custom redaction was removed
Header link created	A header link was created
Header link updated	A header link was updated
Header link removed	A header link was removed
Rule created	A rule was created
Rule updated	A rule was updated
Rule deleted	A rule was deleted
Templated rule created	A templated rule was created
Templated rule updated	A templated rule was updated
Templated rule removed	A templated rule was removed
List created	A list was created
List updated	A list was updated
List deleted	A list was removed
Custom signal created	A custom signal was created
Custom signal updated	A custom signal was updated
Custom signal removed	A custom signal was removed
Custom alert created	A custom alert was created
Custom alert updated	A custom alert was updated
Custom alert removed	A custom alert was removed
Rate limited IP expired	A rate limited IP was manually expired
Rate limited IPs bulk expired	All rate limited IPs were manually expired
Custom dashboard created	A custom dashboard was created
Custom dashboard updated	A custom dashboard was updated
Custom dashboard reset	A custom dashboard was reset
Custom dashboard deleted	A custom dashboard was removed
Custom dashboard card created	A custom dashboard card was created
Custom dashboard card updated	A custom dashboard card was updated
Custom dashboard card deleted	A custom dashboard card was removed
Default dashboard updated	The default dashboard was changed
Agent alert	An agent alert was triggered
Weekly digest sent	The weekly digest was sent
Monitor URL enabled	The monitor view URL for a dashboard was enabled
Monitor URL disabled	The monitor view URL for a dashboard was disabled
Monitor URL created	The monitor view URL for a dashboard was updated
Monitor URL invalidated	The previous monitor view URL for a dashboard was disabled
CloudWAF SSL certificate uploaded	An SSL certificate for CloudWAF was uploaded to the site
CloudWAF SSL certificate deleted	An SSL certificate for CloudWAF was deleted from the site
CloudWAF config updated	The CloudWAF configuration was updated

Working with templated rules

Templated rules are partially pre-constructed rules that can help you protect against Common Vulnerabilities and Exposures (CVE) and gain visibility into registrations, logins, and API requests. For example, you can enable the GraphQL API Query templated rule to track GraphQL API requests.

- The Templated Rules page is only included with the Premier and Professional platforms. It is not included as part of the Essential platform.
- The Signals page is only included with the Essential platform. It is not included as part of the Premier and Professional platforms.
- Depending on the type of templated rule, the Essential platform includes a different level of support:

Type	Support
API protection rules	Some API protection signals are not supported.
ATO protection rules	All ATO protection signals are not supported.
Virtual patching rules	Virtual patching rules are only supported in BLOCK mode. Threshold blocking is not supported.

- To use the GraphQL API Query templated rule, your agents must be on version 4.33.0 or above.

Types of templated rules

There are three types of templated rules:

- **API protection rules:** tags requests made to your API, allowing you to detect patterns such as repeated API requests from an unexpected user agent. API Protection signals are informational, so only certain requests tagged with these signals will appear in the requests page of the console. See Storage categories for additional details.
- **ATO protection rules:** enable you to quickly create rules to identify account takeover (ATO) attacks, such as failed password reset attempts. With the exception of the Login and Registration groups of signals, ATO Protection signals are informational, so only certain requests tagged with these signals will appear in the requests page of the console. See Storage categories for additional details.
- **Virtual patching rules** block or log requests matching specific vulnerabilities. These can be configured to send an alert after a threshold of matching requests.

Enabling and editing templated rules

1. From the **Site Rules** menu, select **Templated Rules**. The templated rules menu page appears.
2. Click **View** to the right of the rule you want to enable or edit. The page for that templated rule appears. This page features a graph, Event list, and list of requests tagged with the signal associated with this rule.
3. Click **Configure** in the upper-right corner to enable or edit the rule. The rule builder page appears. The rule builder will feature pre-built rule conditions designed for the templated rule you selected.
4. In the **Value** fields, enter values specific to your application, such as paths, response codes, and headers. It is possible to add, edit, and remove conditions in the rule as necessary for your application.
5. Click **Update Site Rule**.

Enabling threshold blocking

When configuring Failed Logins or Failed Registrations, you have the additional option to block either subsequent Login Attempts or Registration Attempts respectively.

The duration for the block is customizable. Either the site default (normally 1 day), 10 minutes, 1 hour, 6 hours, or 24 hours.

Working with virtual patching rules

Virtual patching rules are partially pre-constructed rules that allow you to block, tag, and log requests that match CVEs. The rules can be configured to send an alert when a threshold of matching requests is reached.

New virtual patching rules are announced through an optional email subscription. You can subscribe to virtual patching announcements in your account settings.

Working with virtual patching rules from the Templated Rules page

For Premier and Professional platforms, you can view, enable, and edit virtual patching rules from the Templated Rules page.

Working with virtual patching rules from the Signals page

For Essential platform, you can view, enable, and edit virtual patching rules from the Signals page.

View virtual patching rules from the Signals page

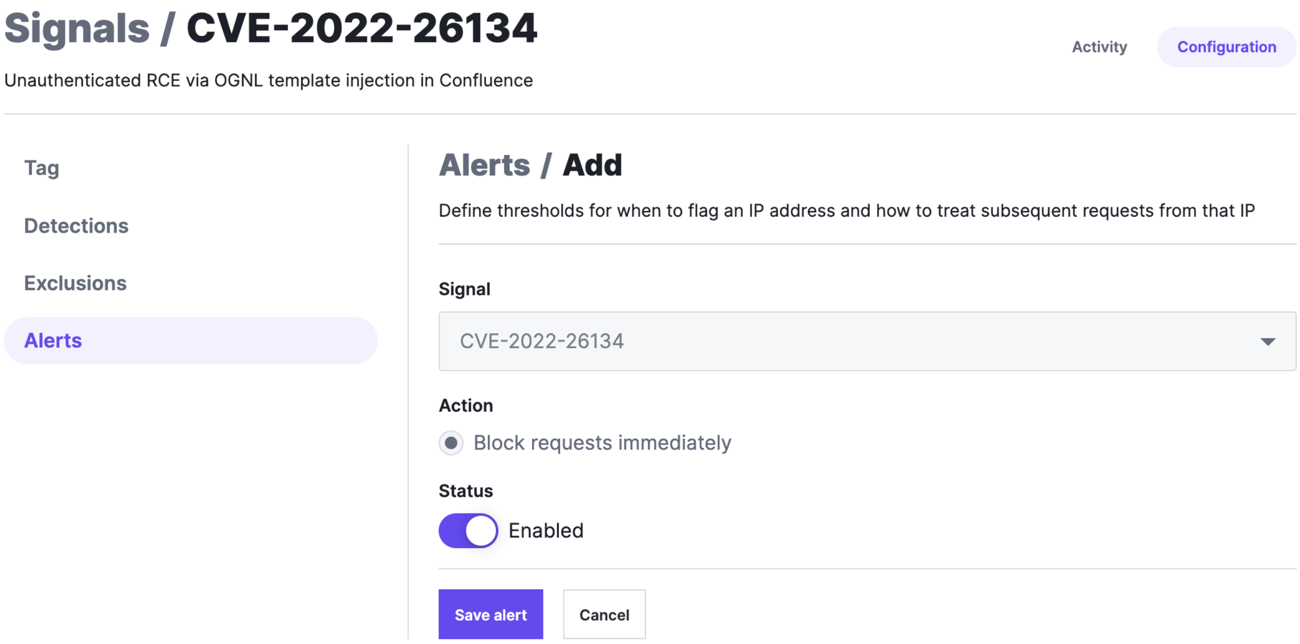
To view virtual patching rules, follow these steps:

1. Log in to the Signal Sciences console.

Enable virtual patching rules from the Signals page

To enable a virtual patching rule, follow these steps:

1. On the Signals page, click **View** in the row of the virtual patching rule that you want to enable. An activity overview of the selected rule appears.
2. Click the **Configuration** tab. Configuration options for the signal appear.
3. Click the **Alerts** tab. The Alerts tab appears.
4. Click **Add alert**. The Add form appears.



Signals / CVE-2022-26134

Unauthenticated RCE via OGNL template injection in Confluence

Alerts / Add

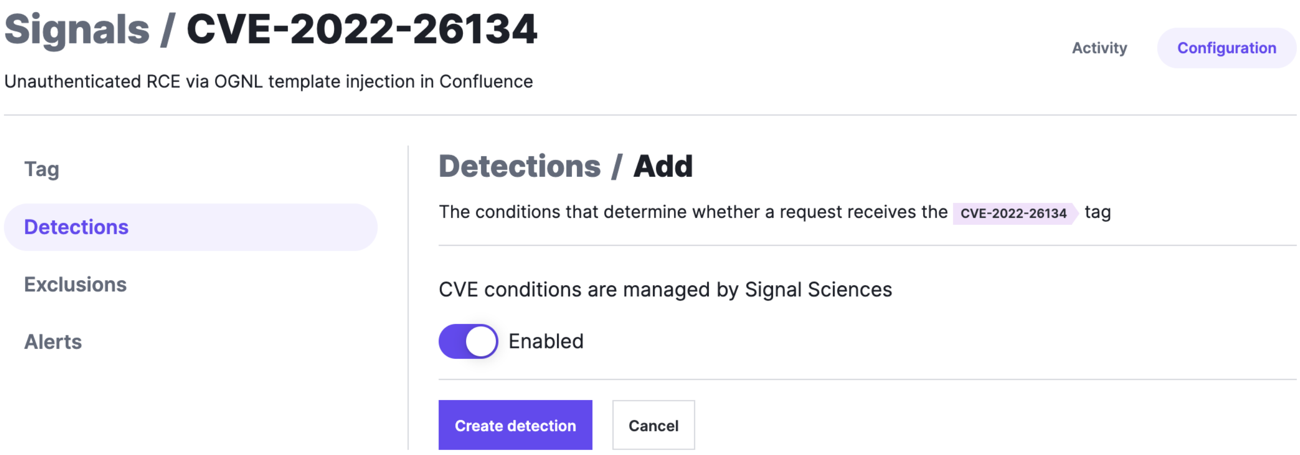
Define thresholds for when to flag an IP address and how to treat subsequent requests from that IP

Signal: CVE-2022-26134

Action: ● Block requests immediately

Status: Enabled

5. Fill out the alert configuration fields as follows:
   - In the **Signal** area, verify that the virtual patching rule that you want to enable is selected.
   - In the **Action** area, select **Block requests immediately**.
   - In the **Status** area, set the switch to **Enabled**.
6. Click the **Save alert** button. The virtual patching rule is enabled.
7. Click the **Detections** tab. The Detections configuration tab appears.
8. Click **Add detection**. The Add form appears.



Signals / CVE-2022-26134

Unauthenticated RCE via OGNL template injection in Confluence

Detections / Add

The conditions that determine whether a request receives the CVE-2022-26134 tag

CVE conditions are managed by Signal Sciences

Enabled

9. Verify the switch is set to **Enabled**.
10. Click the **Create detection** button. Requests that match the virtual patching rule are assigned the tag associated with the rule.

Our Teams integration allows you to be notified when certain activity occurs on Signal Sciences.

Adding Teams integration

You can add Teams integration for both Corps and Sites.

Corp integration

> **Note:** Only Owners can create, edit, and delete corp integrations.

1. Add a custom incoming webhook in Microsoft Teams
2. Copy the **Webhook URL** of the new webhook.
3. Log in to the Signal Sciences console.
4. From the **Corp Manage** menu, select **Corp Integrations**. The corp integrations menu page appears.
5. Click **Add corp integration**. The add corp integration menu page appears.
6. Select the **Microsoft Teams** integration. The Microsoft Teams integration setup page appears.
7. In the **Webhook URL** field, enter the **Webhook URL** created in Slack.
8. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
9. Click **Create corp integration**.

Site integration

1. Add a custom incoming webhook in Microsoft Teams
2. Copy the **Webhook URL** of the new webhook.
3. Log in to the Signal Sciences console.
4. From the **Sites** menu, select a site if you have more than one site.
5. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
6. Click **Add site integration**. The add site integration menu page appears.
7. Select the **Microsoft Teams** integration. The Microsoft Teams integration setup page appears.
8. In the **Webhook URL** field, enter the **Webhook URL** created in Slack.
9. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
10. Click **Create site integration**.

Activity types

Corp and Site integrations have the following separate activity types.

Corp

Activity type	Description
`releaseCreated`	New release notifications
`featureAnnouncement`	New feature announcements
`corpUpdated`	Account timeout setting updated
`newSite`	A new site was created
`deleteSite`	A site was deleted
`enableSSO`	SSO was enabled for the corp
`disableSSO`	SSO was disabled for the corp
`corpUserInvited`	A user was invited
`corpUserReinvited`	A user was reinvited
`listCreated`	A list was created
`listUpdated`	A list was updated
`listDeleted`	A list was removed
`customTagCreated`	A custom signal created
`customTagDeleted`	A custom signal updated
`customTagUpdated`	A custom signal removed
`userAddedToCorp`	A user was added to the corp
`userMultiFactorAuthEnabled`	A user enabled 2FA

`userMultiFactorAuthUpdated`	A user updated 2FA secret
`userRegistered`	A user was registered
`userRemovedCorp`	A user was removed from the corp
`userUpdated`	A user was updated
`userUndeliverable`	A user’s email address bounced
`userUpdatePassword`	A user updated their password
`accessTokenCreated`	An API Access Token was created
`accessTokenDeleted`	An API Access Token was deleted

Site

Activity type	Description
`siteDisplayNameChanged`	The display name of a site was changed
`siteNameChanged`	The short name of a site was changed
`loggingModeChanged`	The agent mode (“Blocking”, “Not Blocking”, “Off”) was changed
`agentAnonModeChanged`	The agent IP anonymization mode was changed
`flag`	An IP address was flagged
`expireFlag`	An IP address flag was manually expired
`createCustomRedaction`	A custom redaction was created
`removeCustomRedaction`	A custom redaction was removed
`updateCustomRedaction`	A custom redaction was updated
`customTagCreated`	A custom signal was created
`customTagUpdated`	A custom signal was updated
`customTagDeleted`	A custom signal was removed
`customAlertCreated`	A custom alert was created
`customAlertUpdated`	A custom alert was updated
`customAlertDeleted`	A custom alert was removed
`detectionCreated`	A templated rule was created
`detectionUpdated`	A templated rule was updated
`detectionDeleted`	A templated rule was removed
`listCreated`	A list was created
`listUpdated`	A list was updated
`listDeleted`	A list was removed
`ruleCreated`	A request rule was created
`ruleUpdated`	A request rule was updated
`ruleDeleted`	A request rule was deleted
`customDashboardCreated`	A custom dashboard was created
`customDashboardUpdated`	A custom dashboard was updated
`customDashboardReset`	A custom dashboard was reset
`customDashboardDeleted`	A custom dashboard was removed
`customDashboardWidgetCreated`	A custom dashboard card was created
`customDashboardWidgetUpdated`	A custom dashboard card was updated
`customDashboardWidgetDeleted`	A custom dashboard card was removed
`agentAlert`	An agent alert was triggered

OpsGenie

Our OpsGenie issue integration creates an alert when IP addresses are flagged on Signal Sciences.

Adding a OpsGenie integration

3. Log in to the Signal Sciences console.
4. From the **Sites** menu, select a site if you have more than one site.
5. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
6. Click **Add site integration**. The add site integration menu page appears.
7. Select the **OpsGenie Alert** integration. The OpsGenie Alert integration setup page appears.
8. In the **API Key** field, enter the **API Key** created in OpsGenie.
9. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
10. Click **Create site integration**.

Activity types

Activity type	Description
`flag`	An IP address was flagged
`agentAlert`	An agent alert was triggered

Setting up single sign-on (SSO)

Single sign-on (SSO) is a means of allowing your users to authenticate against a single identity provider to access your corp. We support both SAML 2.0 and Google Apps SSO (OAuth 2.0).

Enabling single sign-on

Single sign-on can be enabled by Owners on the **User Authentication** page in the **Corp Manage** menu. In the **Authentication** section, click either **Switch to SAML** or **Switch to Google Apps**.

Enabling SAML single sign-on

After clicking **Switch to SAML**, you'll be required to specify the SAML 2.0 Endpoint and x.509 public certificate from the app configured in your identity provider.

If you use Okta or OneLogin, you should be able to search for the Signal Sciences application. Otherwise, configure an application with the following settings:

- **Recipient/Consumer URL:** `https://dashboard.signalsciences.net/saml`
- **Audience URI (SP Entity ID):** `https://dashboard.signalsciences.net/`
- **Consumer URL Validator:** `^https:\/\/dashboard\.signalsciences\.net\/saml$`

A few things to note if you're self-configuring:

- We require a signed SAML response, but don't care about individually-signed assertions. They won't hurt anything, but they will be ignored. Ensure your overall response is signed.
- You must allow SP (Service Provider) initiated logins to complete the handshake that sets up SAML (see below). Once that's complete, you will be able to use IdP (Identity Provider) initiated logins.
- We do not currently publish metadata.

  Note: If using PingFederate as your SSO provider, you will need to deselect the **Require authn requests to be signed when received via the post or redirect bindings** and **Always sign the SAML assertion** settings under the Signature Policy settings.

Enabling Google Apps single sign-on

Google Apps single sign-on uses OAuth 2.0 to authenticate. After clicking **Switch to Google Apps**, you'll be redirected to Google to authenticate. The domain of the email you authenticate against will be used as the SSO domain for the corp.

After you've authenticated, you'll be redirected back to Signal Sciences. You will be shown the domain you selected and be required to enter your password to confirm. If you chose the wrong domain, change the domain by clicking **Switch domains**.

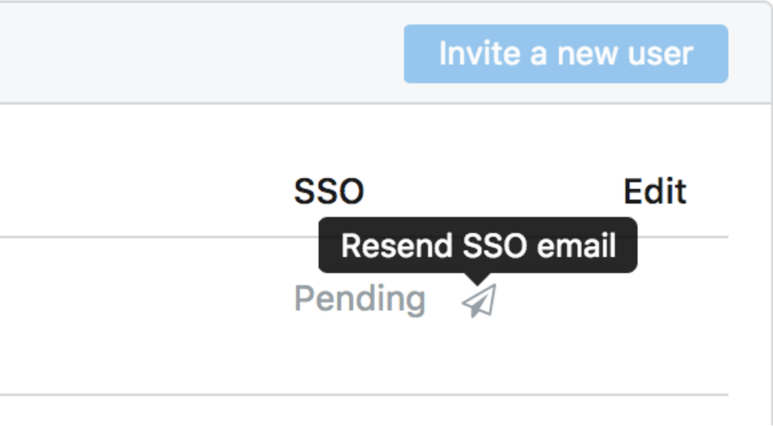
What if the email from my identity provider doesn't match the email in my Signal Sciences account?

If the email from your identity provider doesn't match the email in your Signal Sciences account, you will be alerted that your Signal Sciences email will be changed to your identity provider's email when you enable SSO.

If the email you choose doesn't match the email in your Signal Sciences account and conflicts with an email already in the system, you will be shown an error message and be required to choose another email.

After enabling single sign-on

If the SSO binding link expires, resend it by clicking the **Resend SSO email** button next to the **Pending** SSO status in the **Users** panel on the User Management page.



To enforce SSO, all other users will have their active sessions expired.

What do existing users see when I enable single sign-on?

Existing users will receive an email telling them that they need to set up single sign-on to authenticate against Signal Sciences. Once they successfully configure SSO, they will receive an email confirming the change.

If they attempt to sign in before following the SSO link in their email, they will receive an error message telling them that SSO has been enabled for their corp and to follow the link in their email.

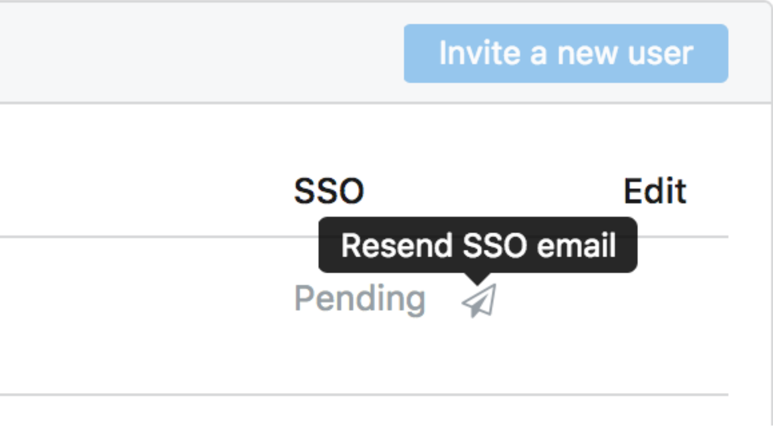
What if an existing user authenticates with an email address in their identity provider that doesn't match the email in their Signal Sciences account?

If the email they authenticate with in their identity provider doesn't match the email in their Signal Sciences account, they will be alerted that their Signal Sciences email will be changed to the email address of the identity provider when they finish authenticating their account.

If the email they choose doesn't match the email in their Signal Sciences account and conflicts with an email already in the system, they will be shown an error message and be required to choose another email.

What if an existing user didn't receive the SSO email?

If the existing user didn't receive the email or the SSO link expires, resend it by clicking the **Resend SSO email** button next to the **Pending** SSO status next to the user's name in the **Users** panel on the User Management page.



What do new users see when I enable single sign-on?

When new users accept an invitation, they'll be prompted to authenticate via the identity provider associated with the corp.

How does sign-in work?

When users visit the Signal Sciences sign-in page, they'll need to enter in their email.

If they authenticate with an email that is different from the email they entered, they will receive an error message.

What happens if I have two-factor auth enabled?

When single sign-on is enabled, all passwords and 2FA tokens are deleted. 2FA is not enforced and we recommend you configure two-factor auth with your identity provider.

How do I disable single sign-on?

Owners can disable single sign-on for all users on the corp. After disabling single sign-on, all other users in your corp will have their active sessions expired. They will receive an email with a link to set a new password, informing them SSO has been disabled. All users will need to set new passwords to log back into the Signal Sciences console.

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. From the **Corp Manage** menu, select **User Authentication**. The user authentication menu page appears.
4. To the right of **Signal Sciences built-in authentication**, click **Switch to built-in auth**. The set password page appears.
5. You are required to set a new password for your user before disabling single sign-on to prevent you from being locked out of the Signal Sciences console. In the **Password** field, enter your new password.
6. Click **Continue**.

Can I set specific users to bypass single sign-on?

If your corp has single sign-on enabled, an Owner user can set a user to bypass SSO, which allows them to log in to the Signal Sciences console via username and password without needing to authenticate through your SSO provider.

1. Log in to the Signal Sciences console.
2. From the **Corp Manage** menu, select **Corp Users**. The Corp User management page appears.
3. Click on the user you want to bypass SSO. The view user page appears.
4. Click **Edit corp user**. The edit user page appears.
5. Under **Authentication**, select **Allow this user to bypass Single Sign-On (SSO)**.
6. Click **Update user**.

Do you support automatic provisioning, or deprovisioning?

We don’t support automatic provisioning / deprovisioning at this time. If this is something you’re interested in, reach out to us with your use case.

What is a single sign-off endpoint (SAML Logout Endpoint)?

If your corp’s IT department determines you need to use a custom logout URL to handle logout redirects and cookie updates, it is possible to supply an optional logout endpoint. There are no parameters necessary, the browser will do a GET request and follow any sign-out/redirects supplied by your IT department.

About the My Profile Menu

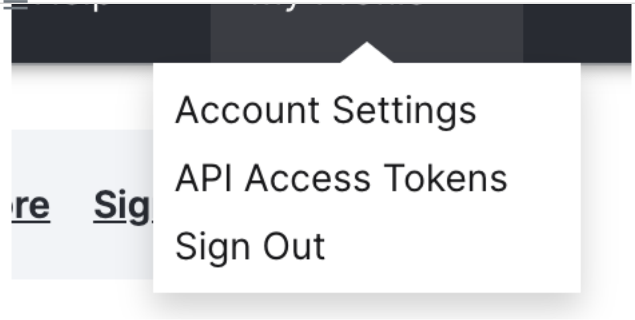
The My Profile menu provides you with access to your personal profile information and settings as determined by the role you have been assigned. The My Profile menu also provides you access to your personal API access tokens for using the Signal Sciences API and a way to log out of the web interface.

Before you begin

Be sure you know how to access the web interface controls before learning about each of the pages you’ll encounter there.

About the My Profile menu

The My Profile menu appears at the far right of the corp navigation bar:



About the Account Settings

Selecting **Account Settings** from the My Profile menu displays the name and email address tied to your account as well as specific settings. The settings displayed will vary depending on your user role and package. From the Account Settings, you can:

- enable and disable two-factor authentication (2FA)
- subscribe to alerts for your corps and sites
- change your password
- link your Fastly and Signal Sciences accounts

Subscribing to alerts

You can elect to be notified via email about certain corp and site integration activity on Signal Sciences. Select the checkbox next to the subscription, and then click the **Update subscriptions** button to start receiving email notifications. For a more extensive list of alerts to subscribe to, check out our mailing list integration.

Changing your password

To change your password from the Account Settings:

1. In the **Current password** field, enter your existing password.
2. In the **New password** field, enter the new password.
3. In the **Confirm password** field, enter the new password a second time.
4. Click **Change Password** to save the changes.

About the API Access Tokens

Selecting **API Access Tokens** from the My Profile menu displays the personal API access tokens associated with your account.

What’s next

Dig deeper into details about all areas of the web interface controls.

PagerDuty

Our PagerDuty issue integration creates an incident when IP addresses are flagged on Signal Sciences.

Adding a PagerDuty integration

PagerDuty issue integrations are configured per project.

1. Create a new service in PagerDuty selecting **Use Our API Directly** from the **Integration Type** menu.
2. Copy the newly created **Service API Key**.
3. Log in to the Signal Sciences console.
4. From the **Sites** menu, select a site if you have more than one site.
5. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
6. Click **Add site integration**. The add site integration menu page appears.
7. Select the **PagerDuty Trigger** integration. The PagerDuty integration setup page appears.
8. In the **Service API Key** field, enter the **Service API Key** created in PagerDuty.
9. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
10. Click **Create site integration**.

Activity types

`agentAlert`	An agent alert was triggered

Using user roles and permissions

Every user in your corp is assigned a role. Roles are groups of permissions that afford users the ability to view and control a variety of things in your corp.

- **Owners** have access to all corp features, can edit settings on every site, and can make changes to user accounts.
- **Admins** have limited access to corp features, access to specific sites and site-level settings, and can invite new users to specific sites.
- **Users** have access to specific sites and site-level settings.
- **Observers** have access to specific sites.

Corp management permission

The corp management permissions for each role are as follows:

Permission	Owner	Admin	User	Observer
View corp-wide data and reports	Access	Limited access	Limited access	Limited access
Edit corp-wide security policies	Access	No access	No access	No access
Create or edit Corp Rules	Access	No access	No access	No access
View Corp Rules	Access	Access	Access	Access
Create or edit Corp Lists	Access	No access	No access	No access
Create or edit Corp Signals	Access	No access	No access	No access
View corp integrations	Access	Access	Access	Access
Edit corp integrations	Access	No access	No access	No access
View corp audit logs	Access	Access	Access	Access

User management permissions

The user management permissions for each role are as follows:

Permission	Owner	Admin	User	Observer
View users	All sites	Specific sites	Specific sites	Specific sites
Invite or remove other users	All sites	Specific sites	No sites	No sites
Allow users to create API Access Tokens	Access	No access	No access	No access

Site management permissions

The site management permissions for each role are as follows:

Permission	Owner	Admin	User	Observer
Create or delete sites	Access	No access	No access	No access
View site-level data and reports	All sites	Specific sites	Specific sites	Specific sites
Edit site blocking mode	All sites	Specific sites	Specific sites	No sites
Edit site IP anonymization policy	All sites	Specific sites	Specific sites	No sites
Edit site default blocking response code	All sites	Specific sites	Specific sites	No sites
View associated users	All sites	Specific sites	Specific sites	No sites
Edit site Display Name and Short Name	All sites	Specific sites	Specific sites	No sites

Site configuration permissions

The site configuration permissions for each role are as follows:

Permission	Owner	Admin	User	Observer
Change Blocking Mode	All sites	Specific sites	Specific sites	No sites
Create or edit rules	All sites	Specific sites	Specific sites	No sites
View rules	All sites	Specific sites	Specific sites	Specific sites
Create or edit signals	All sites	Specific sites	Specific sites	No sites
View signals	All sites	Specific sites	Specific sites	Specific sites
Create or edit lists	All sites	Specific sites	Specific sites	No sites
View lists	All sites	Specific sites	Specific sites	Specific sites

View redactions	All sites	Specific sites	Specific sites	Specific sites
Create or edit integrations	All sites	Specific sites	Specific sites	No sites
View integrations	All sites	Specific sites	Specific sites	Specific sites
Create agent keys	All sites	Specific sites	Specific sites	No sites
View agent keys	All sites	Specific sites	Specific sites	No sites
View site audit logs	Access	Access	Access	Access

Personal account management permissions

The personal account management permissions for each role are as follows:

Permission	Owner	Admin	User	Observer
Edit account profile information	Access	Access	Access	Access
Create, edit, view support tickets	Access	Access	Access	Access
Create API Access Token	Limited access	Limited access	Limited access	Limited access

About the Site Overview page

The Site Overview page allows you to view metrics for a site via system-generated and custom dashboards.

Before you begin

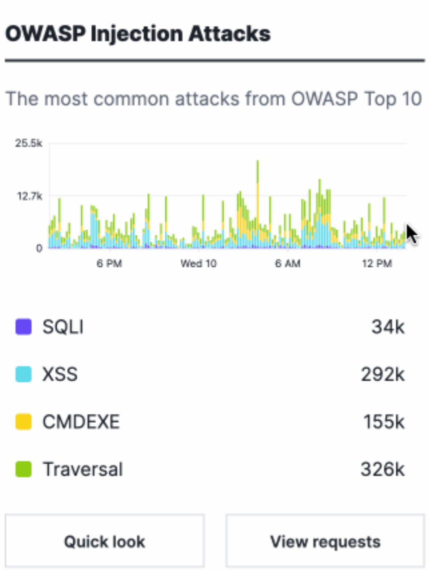
Be sure you know how to access the web interface controls.

About the Site Overview page

The Site Overview page allows you to control:

- the specific system-generated or custom dashboard for which to display metrics
- the time frame over which to display data

Different metrics appear depending on the dashboard you've selected. Hovering over any part of a graph displays a timestamp indictor that updates itself as you move your cursor.



OWASP Injection Attacks

The most common attacks from OWASP Top 10

🟪 SQLI	34k
🟦 XSS	292k
🟨 CMDEXE	155k
🟩 Traversal	326k

At the bottom of each card are buttons that provide more details about the data in each graph.

Clicking on the **Quick Look** button displays a summary view of the data in the graph.

Clicking on the **View Requests** button takes you to the Requests page with data from the graph applied in the search filter. The Requests page shows individual requests that contain attack or anomaly data.

Overview dashboard

The Overview dashboard provides a high-level, system-generated overview of metrics related to your site. It includes the following cards:

- **What's new:** a list of the latest Signal Sciences feature announcements.
- **Scanners:** a graph displaying the number of commercial and open source scanning tools over time.
- **Traffic Source Anomalies:** a graph displaying the number of requests from unusual or suspicious sources over time.
- **Events:** a list of IPs that were flagged for exceeding thresholds. Click **View all events** to open the Events page.
- **Request Anomalies:** a graph displaying the number of anomalous behaviors within request headers over time.
- **Response Anomalies:** a graph displaying the number of client and server error codes over time.
- **Suspicious IPs:** a list of IPs that are approaching thresholds. Once the threshold is met or exceeded, the IP address will be flagged and added to the Events list. If the agent mode is set to blocking, then all malicious requests from flagged IPs are blocked (without blocking legitimate traffic).
- **Authentication:** a graph displaying the number of attempts to log in to application endpoints over time.
- **Top Attacks:** a list of the top URLs containing attack signals.

API Protection dashboard

The API Protection dashboard provides system-generated data about API protection signals. It includes the following cards:

- **Enumeration:** a graph displaying the number of attempts to access enumerated resources over time.
- **Request anomalies:** a graph displaying the number of anomalous behaviors within request headers over time.
- **Injection attacks:** a graph displaying the number of OWASP attacks associated with API abuse over time.
- **Serialization anomalies:** a graph displaying the number of request errors over time. The errors may indicate autonomous clients.
- **Request violations:** a graph displaying the number of requests violating common controls over time.
- **Traffic source anomalies:** a graph displaying the number of requests from unusual or suspicious sources over time.

ATO Protection dashboard

The ATO Protection dashboard provides system-generated data about account takeover (ATO) signals. It includes the following cards:

- **Login:** a graph displaying the number of attempts to log in to application endpoints over time.
- **Password reset:** a graph displaying the number of attempts to reset passwords over time.
- **Account creation:** a graph displaying the number of attempts to create accounts over time.
- **Account changes:** a graph displaying the number of changes to sensitive account information over time.
- **Anomalies:** a graph displaying the number of requests from unusual or suspicious sources over time.
- **Gift card validation:** a graph displaying the number of attempts to validate gift card details over time.
- **Credit card validation:** a graph displaying the number of attempts to validate credit card details over time.
- **Spam:** a graph displaying the number of requests to application messaging features over time.

What's next

Learn how to work with custom dashboards on the Site Overview page.

Pivotal Tracker

The PivotalTracker integration allows you to create a story anytime an event triggers.

Adding a PivotalTracker integration

PivotalTracker alerts integrations are configured per project.

1. In PivotalTracker, locate your **API token**.
2. Access your Pivotal Tracker project settings and, under **Access**, locate your **Project ID**.
3. Log in to the Signal Sciences console.
4. From the **Sites** menu, select a site if you have more than one site.
5. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
6. Click **Add site integration**. The add site integration menu page appears.
7. Select the **PivotalTracker Story** integration. The PivotalTracker story integration setup page appears.
8. In the **API Token** field, enter the **API token** found in PivotalTracker.
9. In the **Project ID** field, enter the **Project ID** found in PivotalTracker.
10. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
11. Click **Create site integration**.

Activity types

Activity type	Description
`flag`	An IP address was flagged

About the Requests page

The Requests page features a summary table, which lists individual requests that have been tagged with signals and that fit into either the all or sampled data storage category. The summary table includes general request data (e.g., path, response code, response size) and identifies the specific attacks and anomalies that a request was tagged with.

Before you begin

Be sure you know how to access the web interface controls.

About the Requests page

Selecting **Requests** from the site navigation bar displays the Requests page.

From the Requests page, you can:

- use menus to filter the table by when the requests were made and by the signals and HTTP response codes that the requests are tagged with.
- enter queries into a search bar to filter the table. Clicking the **Show search examples** link reveals example search queries with valid syntax.
- click on signals and linked data in the table to filter the table's contents. For example, clicking on a source IP will constrain the results to all requests made by that IP.
- view full details for an individual request by clicking the **View request detail** link. The request details page lists all of the metadata captured about the request, including request and response headers, and all the signals we've identified. This page can help you further debug a particular attack or anomaly.

  > **NOTE:** You may need additional context to fully investigate an attack or anomaly. To do this, we recommend using a header link to add a link to your internal systems on the request details page via a linking identifier (e.g., an X-Request-Id response header).

- download full details for the first 1,000 requests by clicking the **Download as** menu and selecting **JSON** or **CSV**. To download additional requests, use the **Next** button to navigate to a subsequent results page and click **Download as** again.

Agent alerts

You can set up agent alerts to inform you when thresholds are reached. To set up agent alerting, click on the **Manage Alerts** button at the top of the Agents page.

The alerting system uses our integrations to communicate. You must first have at least one integration configured to set up an agent alert. There are two types of alerts:

- **Average RPS:** Will alert whenever the average number of requests per second (RPS) for all agents across all sites reaches a specified threshold. We offer an out-of-the-box alert (disabled by default) if the average number of requests per second (RPS) for all agents falls below 10. If you are a high RPS customer, this alert could let you know of a possible issue.
- **Online Agent Count:** Will alert whenever the number of online agents reaches a specified threshold. We offer an out-of-the-box alert (disabled by default) when the agent count falls to zero, which could be indicative of a problem.

  Note: You likely do not need both alerts enabled. Most customers find it useful to have one, but not both, enabled. Which alerts are useful to you will be specific to your setup.

You can edit and create multiple alerts. Currently, we offer alerting based on average agent RPS across all sites and online agent count. You can customize these alerts to specify values, boolean operators (such as `less than` or `equal to`), and a length of time after which to send the alert.

Slack

Our Slack message integration allows you to be notified when certain activity occurs on Signal Sciences.

Adding a Slack message integration

You can add Slack message integration for both Corps and Sites.

1. In Slack, enable incoming webhooks if you have not already.
2. Create a new webhook.
3. Copy the **Webhook URL** of the new webhook.
4. Log in to the Signal Sciences console.
5. From the **Corp Manage** menu, select **Corp Integrations**. The corp integrations menu page appears.
6. Click **Add corp integration**. The add corp integration menu page appears.
7. Select the **Slack Message** integration. The Slack message integration setup page appears.
8. In the **Webhook URL** field, enter the **Webhook URL** created in Slack.
9. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
10. Click **Create corp integration**.

Site integration

1. In Slack, enable incoming webhooks if you have not already.
2. Create a new webhook.
3. Copy the **Webhook URL** of the new webhook.
4. Log in to the Signal Sciences console.
5. From the **Sites** menu, select a site if you have more than one site.
6. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
7. Click **Add site integration**. The add site integration menu page appears.
8. Select the **Slack Message** integration. The Slack message integration setup page appears.
9. In the **Webhook URL** field, enter the **Webhook URL** created in Slack.
10. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
11. Click **Create site integration**.

Activity types

Corp and Site integrations have the following separate activity types.

Corp

Activity type	Description
`releaseCreated`	New release notifications
`featureAnnouncement`	New feature announcements
`corpUpdated`	Account timeout setting updated
`newSite`	A new site was created
`deleteSite`	A site was deleted
`enableSSO`	SSO was enabled for the corp
`disableSSO`	SSO was disabled for the corp
`corpUserInvited`	A user was invited
`corpUserReinvited`	A user was reinvited
`listCreated`	A list was created
`listUpdated`	A list was updated
`listDeleted`	A list was removed
`customTagCreated`	A custom signal created
`customTagDeleted`	A custom signal removed
`customTagUpdated`	A custom signal updated
`userMultiFactorAuthEnabled`	A user enabled 2FA
`userMultiFactorAuthDisabled`	A user disabled 2FA
`userMultiFactorAuthUpdated`	A user updated 2FA secret
`userRegistered`	A user was registered
`userRemovedCorp`	A user was removed from the corp
`userUpdated`	A user was updated

`userUpdatePassword`	A user updated their password
`accessTokenCreated`	An API Access Token was created
`accessTokenDeleted`	An API Access Token was deleted

Site

Activity type	Description
`siteDisplayNameChanged`	The display name of a site was changed
`siteNameChanged`	The short name of a site was changed
`loggingModeChanged`	The agent mode (“Blocking”, “Not Blocking”, “Off”) was changed
`agentAnonModeChanged`	The agent IP anonymization mode was changed
`flag`	An IP address was flagged
`expireFlag`	An IP address flag was manually expired
`createCustomRedaction`	A custom redaction was created
`removeCustomRedaction`	A custom redaction was removed
`updateCustomRedaction`	A custom redaction was updated
`customTagCreated`	A custom signal was created
`customTagUpdated`	A custom signal was updated
`customTagDeleted`	A custom signal was removed
`customAlertCreated`	A custom alert was created
`customAlertUpdated`	A custom alert was updated
`customAlertDeleted`	A custom alert was removed
`detectionCreated`	A templated rule was created
`detectionUpdated`	A templated rule was updated
`detectionDeleted`	A templated rule was removed
`listCreated`	A list was created
`listUpdated`	A list was updated
`listDeleted`	A list was removed
`ruleCreated`	A request rule was created
`ruleUpdated`	A request rule was updated
`ruleDeleted`	A request rule was deleted
`customDashboardCreated`	A custom dashboard was created
`customDashboardUpdated`	A custom dashboard was updated
`customDashboardReset`	A custom dashboard was reset
`customDashboardDeleted`	A custom dashboard was removed
`customDashboardWidgetCreated`	A custom dashboard card was created
`customDashboardWidgetUpdated`	A custom dashboard card was updated
`customDashboardWidgetDeleted`	A custom dashboard card was removed
`agentAlert`	An agent alert was triggered

Sumo Logic

The generic webhook integration enables you to export notifications for certain activity on Signal Sciences directly to Sumo Logic.

Integrating with Sumo Logic

1. Create a new hosted collector in Sumo Logic.

2. Add an HTTP Logs and Metrics Source to the new hosted collector.

   - Copy the **HTTP Source Address** for later use when setting up the generic webhook integration.

3. Log in to the Signal Sciences console.

6. Click **Add site integration**. The add site integration menu page appears.
7. Select the **Generic Webhook** integration. The Generic Webhook integration setup page appears.
8. In the **Webhook URL** field, enter a URL to receive the notifications at.
9. Select if you want to be alerted regarding **All activity** or **Specific activity**.
    - If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
10. Click **Create site integration**.

Activity types

Activity type	Description	Payload
`siteDisplayNameChanged`	The display name of a site was changed	
`siteNameChanged`	The short name of a site was changed	
`loggingModeChanged`	The agent mode (`Blocking`, `Not Blocking`, `Off`) was changed	Get site by name
`agentAnonModeChanged`	The agent IP anonymization mode was changed	Get site by name
`flag`	An IP address was flagged	Get event by ID
`expireFlag`	An IP address flag was manually expired	List events
`createCustomRedaction`	A custom redaction was created	Create a custom redaction
`removeCustomRedaction`	A custom redaction was removed	Remove a custom redaction
`updateCustomRedaction`	A custom redaction was updated	Update a custom redaction
`customTagCreated`	A custom signal was created	
`customTagUpdated`	A custom signal was updated	
`customTagDeleted`	A custom signal was removed	
`customAlertCreated`	A custom alert was created	Create a custom alert
`customAlertUpdated`	A custom alert was updated	Update a custom alert
`customAlertDeleted`	A custom alert was removed	Remove a custom alert
`detectionCreated`	A templated rule was created	
`detectionUpdated`	A templated rule was updated	
`detectionDeleted`	A templated rule was removed	
`listCreated`	A list was created	Create a list
`listUpdated`	A list was updated	Update a list
`listDeleted`	A list was removed	Remove a list
`ruleCreated`	A request rule was created	
`ruleUpdated`	A request rule was updated	
`ruleDeleted`	A request rule was deleted	
`customDashboardCreated`	A custom dashboard was created	
`customDashboardUpdated`	A custom dashboard was updated	
`customDashboardReset`	A custom dashboard was reset	
`customDashboardDeleted`	A custom dashboard was removed	
`customDashboardWidgetCreated`	A custom dashboard card was created	
`customDashboardWidgetUpdated`	A custom dashboard card was updated	
`customDashboardWidgetDeleted`	A custom dashboard card was removed	
`agentAlert`	An agent alert was triggered	

About the Agents page

The Agents page provides an at-a-glance view of the agents a site uses and high-level stats for those agents. Agents are small daemon processes which provide the interface between your web server and our analysis platform. They determine how requests should be handled (e.g., block, allow, tag).

About the Agents page

To navigate to the Agents page, click **Agents** in the site navigation bar.

The Agents page surfaces relevant information and data about your agents via two tabs:

- **Overview:** a tab featuring agent information (e.g., status, version, module, operating system, and server).
- **Metrics:** a tab highlighting key agent stats (e.g., current requests, connections total, connections open, and connections dropped).

  **Note: Once an agent has been offline for 3 days, it will disappear from the Agents page automatically.

From the Agents page, you can:

- click the **Manage alerts** button to set up and manage agent alerts. Agent alerts use configured integrations to inform you when thresholds for agents are reached.
- click the **View agent keys** button to access the agent keys window. From the agent keys window, you can copy the Agent Access key and Agent Secret key for the site. You need these keys to start and update agents.
- use a single set of filters to narrow the list of agents on both the Overview and Metrics tabs.
- use a search bar on the Overview or Metrics tab to narrow the list of agents on the active tab.

About the Signals page

The Signals page lists the system signals that are available to a site. A signal is a descriptive tag about a request.

Before you begin

Be sure you know how to access the web interface controls.

Limitations and considerations

The Signals page is only included with the Essential platform. Professional and Premier platform users can monitor signal activity via the Signals Dashboard page.

About the Signals page

Selecting **Signals** from the site navigation bar displays the Requests page.

From this page, you can:

- view the system signals for a site. Each signal contains a brief description, a status (e.g., enabled, disabled), and the number of requests that have been tagged with the signal.
- use filters to narrow down the signals list. You can filter by whether the signal is an attack or anomaly and by category (e.g., OWASP attack, scanner, CVE).
- enter text into a search bar to find a signal by tag name or description.
- click the **View** link for a signal to access a detailed overview of the signal's activity and configuration.

VictorOps

The VictorOps integration allows you to send a notification to VictorOps anytime activity occurs. This includes IP flagging events in addition to agent mode changes and allowlisting/blocklisting additions and removals.

Adding a VictorOps integration

VictorOps alerts integrations are configured per project.

1. Log in to VictorOps.

2. From the **Settings** menu, select **Integrations**. The integrations menu page appears.

3. Under **Incoming Alerts**, select **REST Endpoint**.

4. Click **Enable Integration** if you have not already generated an API key.

5. Copy the **Post URL**.

Change `$routing_key` to your target group who should be notified of the alert. Failure to do so may result in missed notifications.

6. Log in to the Signal Sciences console.
7. From the **Sites** menu, select a site if you have more than one site.
8. From the **Manage** menu, select **Site Integrations**. The site integrations menu page appears.
9. Click **Add site integration**. The add site integration menu page appears.
10. Select the **VictorOps Alert** integration. The VictorOps alert integration setup page appears.
11. In the **Webhook URL** field, enter the **Post URL** copied from VictorOps.
12. Select if you want to be alerted regarding **All activity** or **Specific activity**. If you selected **Specific activity**, in the **Activity** menu choose which types of activity you want the integration to create alerts for.
13. Click **Create site integration**.

Activity types

Activity type	Description
`siteDisplayNameChanged`	The display name of a site was changed
`siteNameChanged`	The short name of a site was changed
`loggingModeChanged`	The agent mode (“Blocking”, “Not Blocking”, “Off”) was changed
`agentAnonModeChanged`	The agent IP anonymization mode was changed
`flag`	An IP address was flagged
`expireFlag`	An IP address flag was manually expired
`createCustomRedaction`	A custom redaction was created
`removeCustomRedaction`	A custom redaction was removed
`updateCustomRedaction`	A custom redaction was updated
`customTagCreated`	A custom signal was created
`customTagUpdated`	A custom signal was updated
`customTagDeleted`	A custom signal was removed
`customAlertCreated`	A custom alert was created
`customAlertUpdated`	A custom alert was updated
`customAlertDeleted`	A custom alert was removed
`detectionCreated`	A templated rule was created
`detectionUpdated`	A templated rule was updated
`detectionDeleted`	A templated rule was removed
`listCreated`	A list was created
`listUpdated`	A list was updated
`listDeleted`	A list was removed
`ruleCreated`	A request rule was created
`ruleUpdated`	A request rule was updated
`ruleDeleted`	A request rule was deleted
`customDashboardCreated`	A custom dashboard was created
`customDashboardUpdated`	A custom dashboard was updated
`customDashboardReset`	A custom dashboard was reset
`customDashboardDeleted`	A custom dashboard was removed
`customDashboardWidgetCreated`	A custom dashboard card was created
`customDashboardWidgetUpdated`	A custom dashboard card was updated
`customDashboardWidgetDeleted`	A custom dashboard card was removed
`agentAlert`	An agent alert was triggered

- Events
- Observed Sources
- Signals Dashboard

Before you begin

Be sure you know how to access the web interface controls.

About the Events page

Selecting **Events** from the Monitor menu displays the Events page. Events are actions that Signal Sciences takes as the result of regular threshold-based blocking, templated rules, and site alerts.

The Events page contains a historical record of all flagged IP addresses within the last 30 days. From the Events page, you can:

- filter events by a specific IP address, by status (Active or Expired), or by the signal the event was tagged with.
- view information about an event in the event view area. The event view area is comprised of three sections: details, timeline, and sample request.

Details section

The **Details** section contains detailed information about the event and associated IP address, including:

- **Status:** the status of the event, either Active or Expired.
- **Country:** the country where the request originated.
- **Signal:** the signal tagged to the request.
- **Action:** additional actions taken on the IP address while flagged.
- **Host:** the host where the request originated.
- **User agents:** the user agents seen from this IP address. This list may include web browsers, media players, and other plug-ins.

The **Details** section also provides controls for managing IP addresses that have been flagged. Specifically, you can:

- click the **Remove flag now** button to remove the IP address from the flag list.
- click the **Allow IP** button to create a request rule to allow the IP address.
- click the **Block IP** button to create a request rule to block the IP address.

Timeline section

The **Timeline** section contains a timeline illustrating the actions that occurred during the event. This includes:

- when the IP address was identified as suspicious.
- the number of requests received from the IP address before it was flagged.
- when the IP address was flagged.
- the number of requests that were blocked or logged.
- the current status of the IP address.

Sample request section

The **Sample request** section highlights a single request received during the event, including the request itself and the signals applied to it. Clicking the **View this request** link takes you to the request details page for that request. Clicking the **Edit rule** link in the Signals field will take you to the **View** page for the rule where you can edit the request rule.

About the Observed Sources page

Selecting **Observed Sources** from the Monitor menu displays the Observed Sources page. The Observed Sources page provides an overview of all IP addresses that have been or soon will be flagged on your site. The Observed Sources page contains three tabs: Suspicious IPs, Flagged IPs, and Rate Limited Sources.

Suspicious IPs tab

The **Suspicious IPs** tab shows IP addresses that had requests containing attack payloads of a concerning volume but that did not exceed the decision threshold of flagged IPs. Once the threshold is met or exceeded, an IP address will be flagged and added to the Flagged IPs list. The Suspicious IPs tab helps anticipate which IPs may soon be flagged.

The Suspicious IPs tab lists:

- the suspicious IP address

- the threshold being approached
- how long ago the IP address was added to the Suspicious IPs list
- if the IP was flagged by another Signal Sciences customer

Clicking on an IP address in the Suspicious IPs list will take you to the Requests page with a search for that IP address already applied.

Flagged IPs tab

The **Flagged IPs** tab shows all IP flagging events. IP addresses can be flagged through regular site alerts and templated rules.

The Flagged IPs tab lists:

- the flagged IP address
- the country of origin
- the signal the IP address was flagged on
- how long ago the IP address was flagged
- if the IP address is still currently flagged

Clicking on an IP address in the Flagged IPs list will take you to the Requests page with a search for that IP address already applied.

Rate Limited Sources tab

> **Note:** Rate Limit rules are only included with the Premier platform and certain packaged offerings. They are not included as part of the Professional or Essential platforms.

The **Rate Limited Sources** tab shows all sources that have been rate limited via the Advanced Rate Limiting feature. Rate limit rules are a type of rule that allow you to define arbitrary conditions and automatically begin to block or tag requests that pass a user-defined threshold.

The Rate Limited Sources tab lists:

- the source
- the signal the source was rate limited on
- when the source will stop being rate limited

The tab also provides controls for managing sources that have been rate limited, including:

- removing specific sources from the rate limited sources list.
- creating request rules to allow specific sources.
- creating request rules to block specific sources.

About the Signals Dashboard page

> **Note:** The Signals Dashboards page is only included with the Professional and Premier platforms. Essentials platform users can monitor signals for a site via the Signals page.

Selecting **Signals Dashboard** from the Monitor menu displays the Signals Dashboard page. A signal is a descriptive tag about a request.

From this page, you can:

- view charts that display time series data for signals.
- use filters to narrow down the charted signals. You can filter by corp signals, site signals, OWASP injection attacks, scanners, traffic source anomalies, request anomalies, response anomalies, and virtual patching.
- use the time menu to modify the time frame over which to display data.
- click the chart name to expand a chart and view related target and source details.
- hover your cursor over the information icon on a chart to reveal a description of the signal.

About the Rules menu

The Rules menu is located on the right side of the site navigation bar. From the Rules menu, you can access the following pages:

- Site Rules
- Templated Rules
- Site Lists
- Site Signals
- Site Alerts
- Redactions

About the Site Rules page

Selecting **Site Rules** from the Rules menu displays the Site Rules page. From this page, you can manage your site rules. With site rules, you can allow, block, rate limit, or tag requests for an arbitrary set of conditions.

The Site Rules page lists existing site rules and provides controls to filter the list of rules. Specifically:

- **Type:** allows you to filter the list by the type of rule. Rule types include request rules, signal exclusion rules, and rate limit rules.
- **Status:** allows you to filter the list by whether rules are enabled or disabled.
- **Action:** allows you to filter the list by the type of action that occurs when a request matches a rule's criteria.

Additional controls on the Site Rules page enable you to:

- create site rules by clicking the **Add site rule** button.
- access a detailed view of a site rule by clicking the **View** link to the right of the site rule. Additional controls on the detailed view page allow you to edit and remove the site rule.

About the Templated Rules page

Selecting **Templated Rules** from the Rules menu displays the Templated Rules page. The Templated Rules page lists all templated rules. Templated rules are pre-built rule configurations for registrations, logins, and virtual patches.

From this page, you can:

- view all templated rules.
- filter the templated rules by category (e.g., API, ATO, and CVE), by status, and by whether the rules are recommended or not.
- use a search bar to filter the templated rules by tag name and description.
- access metrics and additional controls for a templated rule by clicking the **View** link to the right of the templated rule. The detailed view page displays a graph, Event list, and list of requests that are tagged with the signal associated with the templated rule. Controls on this page allow to you enable or edit the templated rule.

About the Site Lists page

Selecting **Site Lists** from the Rules menu displays the Site Lists page. The Site Lists page displays your site's lists. Lists are sets of custom data (e.g., list of countries a site doesn't do business with) that are used in site rules.

From this page, you can:

- view your lists.
- create a site list by clicking the **Add site list** button.
- access a detailed view of a site list by clicking the **View** link to the right of the site list. Additional controls on the detailed view page enable you to edit and remove the site list.

About the Site Signals page

Selecting **Site Signals** from the Rules menu displays the Site Signals page. The Site Signals page lists your site's custom signals. A signal is a descriptive tag about a request.

From this page, you can:

- view a list of your custom signals.
- create a custom signal by clicking the **Add site signal** button.
- access a detailed view of a custom signal by clicking the **View** link to the right of the custom signal. Additional controls on the detailed view page enable you to edit and remove the custom signal.

About the Site Alerts page

Selecting **Site Alerts** from the Rules menu displays the Site Alerts page. The Site Alerts page lists your site alerts. Site alerts allow you to define thresholds for when to flag, block, or log an IP address.

From this page, you can:

- view a list of your site alerts.
- create a site alert by clicking the **Add site alert** button.
- access a detailed view of a site alert by clicking the name of the site alert. Additional controls on the detailed view page enable you to edit or remove the site alert.

system-defined redactions. Redactions remove sensitive data from requests before they reach the platform backend. With custom redactions, you can specify the fields to redact from requests.

From this page, you can:

- view a list of your custom redactions.
- create a custom redaction by clicking the **Add redaction** button.
- access a detailed view of a redaction by clicking the **View** link to the right of the redaction. Additional controls on the detailed view page enable you to edit or remove the redaction.

About the Manage menu

The Manage menu is located on the right site of the site navigation bar. From the Manage menu, you can access the following pages:

- Site Settings
- Header Links
- Site Integrations
- Site Audit Log

Before you begin

Be sure you know how to access the web interface controls.

About the Site Settings controls

Selecting **Site Settings** from the Manage menu displays details associated with your site. The Site Settings details include:

- **Name** settings, where you'll find details about the site display name used throughout the Signal Sciences interface and short name used in URLs.
- **Agent Configurations** controls, where you can control agent blocking mode behavior, IP anonymization settings, specify client IP headers, and set a site default blocking response code.
- **Users** controls where you can control user invitations and configure their roles.

About the Header Links controls

Selecting **Header Links** from the Manage menu displays the Header Links page, where you can connect request data from Signal Sciences with your own external data.

From the Header Links page, you can:

- manage existing header links by clicking the **View** button next to a header link.
- add a new header link by clicking the **Add header link** button.

About Site Integrations controls

Selecting **Site Integrations** from the Manage menu displays the Site Integrations page, where you can elect to receive notifications about specific activity within your site.

From the Site Integrations page, you can:

- manage existing corp integrations by clicking on the name of an existing integration.
- add a new site integration by clicking the **Add site integration** button.

About the Site Audit Log

Selecting **Site Audit Log** from the Manage menu displays the Site Audit Log page. This page lists site activity from the last 30 days. You can filter the list by activity type. Types include events and agent alerts and site configuration (e.g., a user created a new integration). You can also choose to exclude agents that are online.

The Site Audit Log page lists connected site integrations in a side bar. You can click on an integration to view details about that integration or click **Manage site integrations** to go to the Site Integrations page.

What's next

Dig deeper into details about all areas of the web interface controls.

either request or response headers to any system (e.g., Kibana, Splunk).

For example, an `X-Request-ID` request header or `X-User-ID` response header can be linked directly to one of your internal systems.

Creating header links

1. Log in to the Signal Sciences Console.
2. From the **Sites** menu, select a site if you have more than one site.
3. From the **Manage** menu, select **Header Links**. The header links menu page appears.
4. Click **Add header link**. The add header link menu page appears.
5. In the **Header name** field, enter the name of the header (e.g., `X-Request-ID`).
6. From the **Header type** menu, select whether the header is a **Request Header** or a **Response Header**.
7. In the **Link template** field, enter the link to your internal system with the value replaced with the string `{{value}}`.

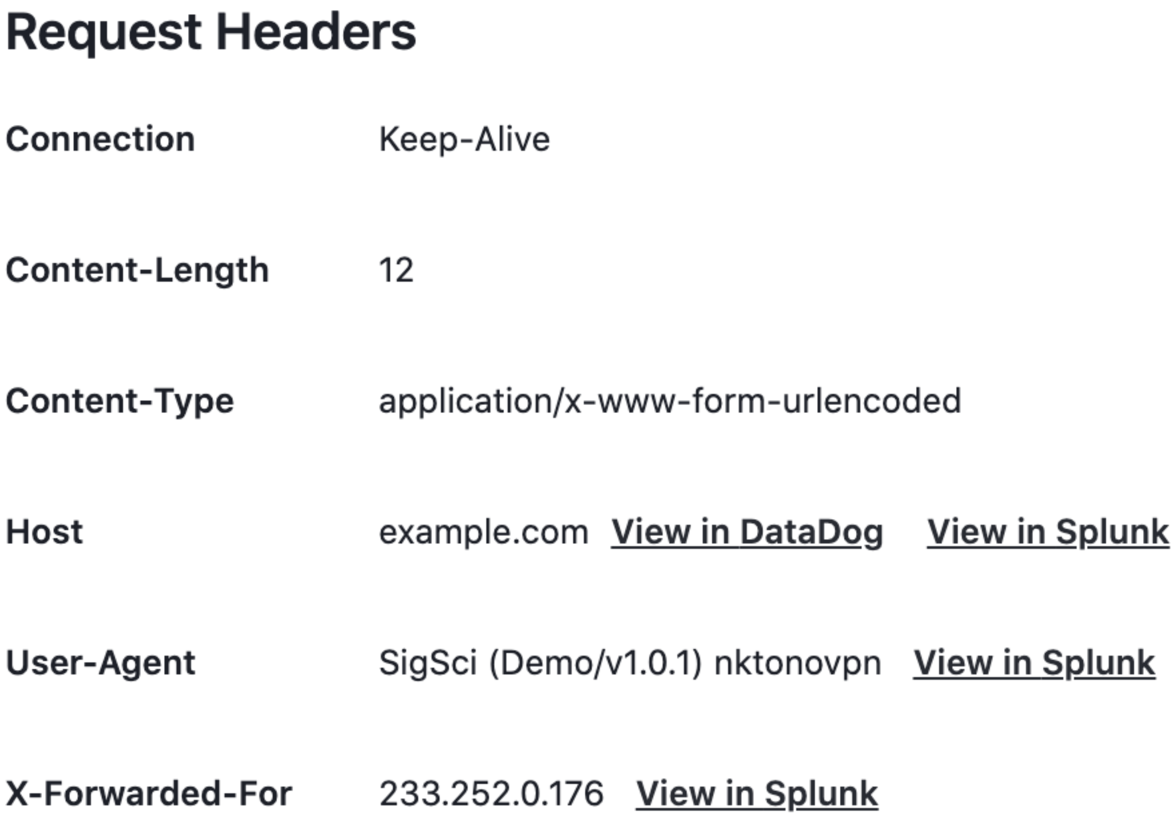
   For example, assume `https://internal-system.example.com/search?X-Forwarded-For&203.0.113.1/results` is the search URL for an internal system which displays all results that contained both the `X-Forwarded-For` header and the IP address `203.0.113.1`.

   To use this URL as the header link template URL for the `X-Forwarded-For` header, you would replace `203.0.113.1` with `{{value}}` in the URL. This makes the link generic and not specific to that single IP address. The header link template URL would then be `https://internal-system.example.com/search?X-Forwarded-For&{{value}}/results`.
8. In the **Display name** field, enter the name of the internal system. This name is used in the header links in the Signal Sciences console. For example, entering `Kibana` will title the link `View in Kibana`.

Using header links

To view the link in action, click **View request detail** on any request on the Requests page.

Underneath either **Request headers** or **Response headers**, next to the header you specified, you will see a header link (e.g., **View in Splunk**). Clicking this link will take you to that internal system with results for that specific header and value.



Request Headers

Connection	Keep-Alive
Content-Length	12
Content-Type	application/x-www-form-urlencoded
Host	example.com View in DataDog View in Splunk
User-Agent	SigSci (Demo/v1.0.1) nktonovpn View in Splunk
X-Forwarded-For	233.252.0.176 View in Splunk

Site alerts monitor and handle requests from IP addresses that have been tagged with specific signals. Specifically, when the number of requests from an IP address meets the signal count threshold for a site alert, the IP address is flagged and select, subsequent requests from the IP address are blocked or logged for a set period of time.

The Events page lists all IP addresses that were flagged in the past 30 days, and the Observed Sources page provides an overview of all IP addresses that have been or soon will be flagged on your site.

Types of site alerts

There are two types of site alerts:

- **System:** site alerts that we've defined to monitor and handle requests from IP addresses that contain attack signals.
- **Custom:** site alerts that you define to monitor and handle requests from IP addresses that contain specific signals.

About system site alerts

System site alerts target attackers' ability to use scripting and tooling. Specifically, they:

- monitor and flag IP addresses that exhibit repeat malicious behavior.
- handle requests from flagged IP addresses.

Flagging occurs when enough attacks are seen from a single IP address. More explicitly, we track the number of attack signals that are seen from a single IP address. When the number of attack signals associated with the IP address reaches one of our thresholds, we flag and blocklist that IP address.

Interval	Threshold	Frequency of check
1 minute	50	Every 20 seconds
10 minutes	350	Every 3 minutes
1 hour	1,800	Every 20 minutes

After an IP address has been flagged, subsequent requests that are from the flagged IP address and that are tagged with an attack signal are either blocked or logged depending on the Agent mode setting. Specifically, requests with an attack signal are blocked if the agent mode is `Blocked` and logged if the agent mode is `Not Blocking`.

By default, malicious traffic from the IP address is blocked or logged for 24 hours. You can change the default time that blocklisted IP addresses are blocked by updating the `blockDurationSeconds` field via our API.

Limitations and considerations

When working with system site alerts, keep the following things in mind:

- Requests that have only been tagged with anomaly and custom signals are not counted towards flagging thresholds.
- The thresholds for the system alerts are based on historical patterns that we've seen across all customers, but the default thresholds may not apply to every application.
- When an IP address is flagged by any Next-Gen WAF customer, we record that IP address as a known potential bad actor and make its status known across our whole network by tagging it with the Network Effect (`SigSci IP`) anomaly signal.

About custom site alerts

You can create custom site alerts to monitor and handle requests from IP addresses that contain specific signals. A custom site alert outlines:

- the criteria that must be met for an IP address to be flagged. For example, flag an IP address when there are 25 SQL Injection attack signals in 1 minute.
- how to handle requests from IP addresses that are flagged. You can either log subsequent requests or block subsequent requests containing attack signals from the IP address.
- how long to block or log subsequent requests from flagged IP addresses.

Limitations and considerations

When working with custom site alerts, keep the following things in mind:

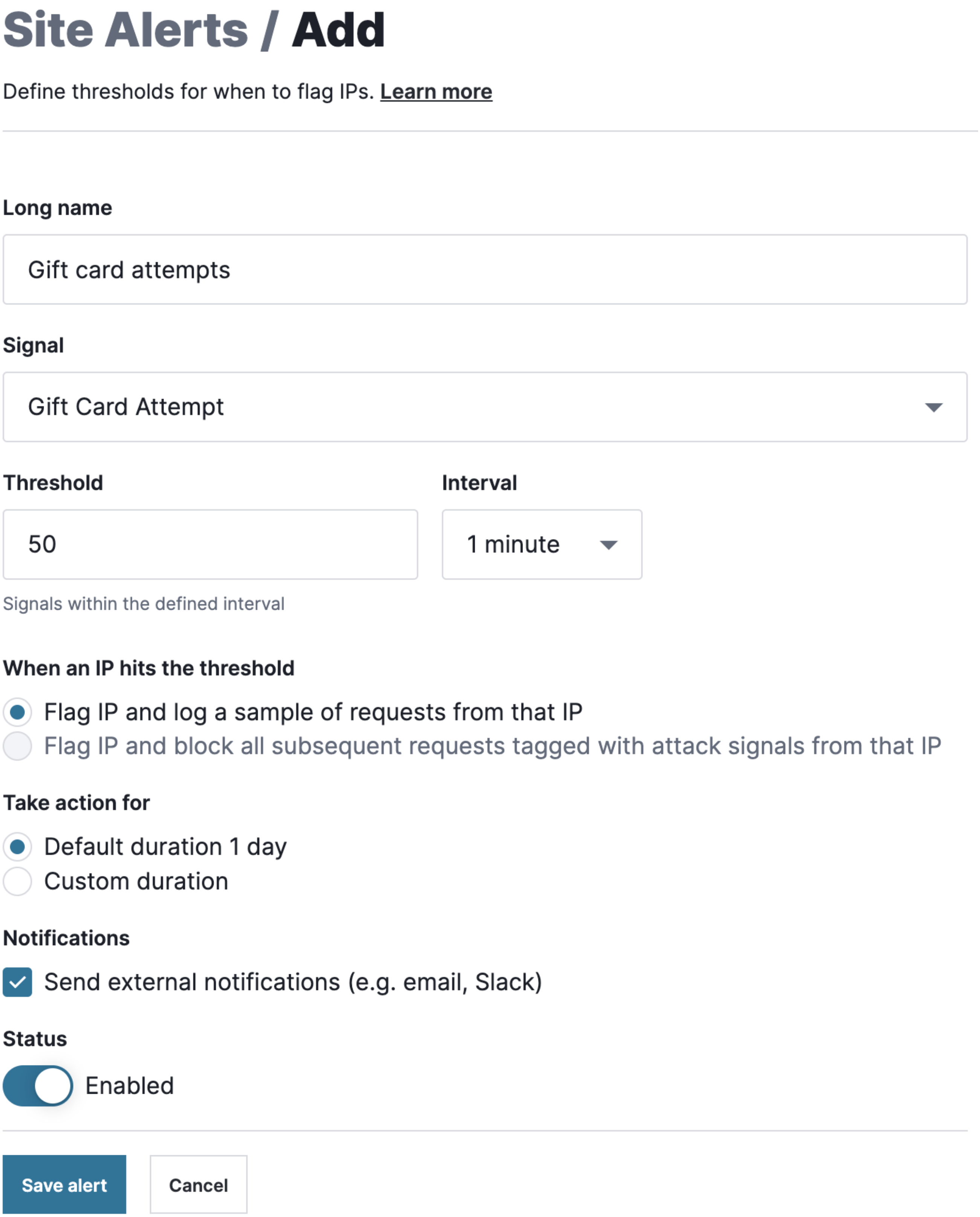
- Custom site alerts are only included with the Professional and Premier platforms. They are not included as part of the Essential platform.
- Accounts are limited to 50 custom site alerts per site.
- Users with an Observer role cannot configure custom site alerts.
- With the Premier platform, you can block all requests from IP addresses that have been flagged for events using request rules with the Site Flagged IP (`SITE-FLAGGED-IP`) anomaly signal.

1. Log in to the Signal Sciences console.

2. From the **Sites** menu, select a site.

3. From the **Rules** menu, select **Site Alerts**. The Site Alerts page appears.

4. Click the **Add site alert** button. The Add form appears.



Site Alerts / Add

Define thresholds for when to flag IPs. Learn more

Long name

Gift card attempts

Signal

Gift Card Attempt

Threshold

50

Interval

1 minute

Signals within the defined interval

When an IP hits the threshold

● Flag IP and log a sample of requests from that IP
○ Flag IP and block all subsequent requests tagged with attack signals from that IP

Take action for

● Default duration 1 day
○ Custom duration

Notifications

☑ Send external notifications (e.g. email, Slack)

Status

Enabled

  - From the **Signal** menu, select the signal that the site alert should track.
  - In the **Threshold** field, enter how many requests containing the signal should be detected before the IP address is flagged.
  - From the **Interval** menu, select the number of minutes during which signals from the IP address are counted to determine if the threshold has been met.
  - Under **When an IP hits the threshold**, select whether the alert should log subsequent requests or block subsequent requests containing attack signals from the IP address. If you selected a custom or anomaly signal as the **Signal**, then you will only be able to log subsequent requests from the IP.
  - Under **Take action for**, select how long the IP address should be flagged. By default, IP addresses are flagged for 24 hours. You can set a custom duration by selecting **Custom duration** and choosing a duration.
  - Leave the **Notifications** checkbox selected to send an external notification (e.g., email and Slack) when the site alert is triggered. Deselect the checkbox to not send any external notifications.
  - Click the **Status** switch to enable the site alert.
6. Click the **Save alert** button.

Site alert precedence

When multiple site alerts exist, the Signal Sciences agent uses the following logic to determine which site rules should take precedence:

- The site alert with the lowest threshold and smallest interval for a given action (i.e., block or log) will be checked first.
- Site alerts with a block action do not compete for precedence against site alerts with a log action.
- After a site alert with a block action flags an IP address, other site alerts with a block action can't flag that IP address until the existing flag is lifted.
- After a site alert with a log action flags an IP address, other site alerts with a log action can't flag that IP address until the existing flag is lifted.
- A site alert with a block action and a site alert with a log action can both flag the same IP address.

Preventing specific IP addresses from being flagged

To prevent an IP address from being flagged by site alerts, create a request rule with an allow action. For example, let's say you plan to scan your web application for vulnerabilities. To ensure the scanning IP address isn't flagged, you can create a request rule with an allow action.

Viewing agent details

You can access a detailed view of each agent from the Agents page. The agent details provide a summary of the status and performance of the agent.

Viewing the agent details

You can access the agent details from the **Agents** page.

To view agent details:

1. Log in to the Signal Sciences console.
2. From the **Sites** menu, select a site if you have more than one site.
3. Click **Agents** in the navigation bar near the top of the screen.
4. From the agents list, click the Agent ID of the agent that you want to view details for.

The following tabs appear in the agents detail:

- Status
- Requests
- Logs
- Charts

About the Charts tab

Signal Sciences provides a number of metrics to understand the performance impact on your infrastructure. You can verify the performance impact on your infrastructure directly in the console via the **Charts** tab.

The **Charts** tab contains graphs detailing the requests the agent has processed, any errors observed, memory usage, CPU percentage, decision times, and more. Many of these graphs report percentiles. For example, the Decision time (ms) graph uses percentiles to show the amount of time it takes the agent to process requests. When the 99th percentile line is at 10 ms, the agent took between 0 - 10 ms to process 99% of requests and more than 10 ms to process the remaining 1% of requests.

- **Total requests (req/sec):** the number of requests per second received for your site.
- **Connections (req/sec):** the total number of connections and the total number of dropped connections per second.
- **Connections open:** the total number of open connections handled by the agent.
- **CPU percentage:** the percentage of CPU used by the host and by the agent.
- **Runtime memory total, heap and stack (bytes):** the total memory being used by the agent.
- **Uptime (seconds):** the agent uptime, in seconds.
- **Decision time (ms):** the amount of time it took for the agent to process requests.
- **Agent queue time (ms):** the amount of time it took for the agent to begin processing requests where there is a backlog of requests.
- **Agent clock skew (seconds):** the difference in time between the agent's clock and the platform's clock.

To view advanced charts, click the **Show advanced charts** button. The following graphs appear.

- **Requests uploaded (req/sec):** the number of requests per second uploaded by the agent to the platform.
- **Requests after sampling (req/sec):** the number of incoming requests per second handed to the agent.
- **Request size, avg. (bytes):** the average size of web request in bytes. This is calculated by dividing the sum of read and written bytes by the number of requests.
- **RPC PreRequest (req/sec):** the number of incoming requests handed to the agent.
- **RPC UpdateRequest (req/sec):** the number of remote calls to agent to update details of request, post execution. Reported in requests per second.
- **RPC PostRequest (req/sec):** the number of remote calls to the agent to note requests that met one of the following conditions: the server returned an HTTP status of 400 or above, the size of the request was abnormally large, or the request took an abnormal amount of time to process.
- **Goroutines:** the number of goroutines running simultaneously on the agent
- **Garbage collections:** the number of time the garbage collector ran.
- **GC pause time (ms/sec):** the garbage collector pause time in milliseconds.
- **Host memory available (bytes):** the amount of memory available on the host machine in bytes.
- **Rule updates:** the number of times the agent updated its rules.
- **Communication failures:** the number of failed uploads and downloads per agent.
- **Agent latency time (ms):** the total amount of time that requests are in the queue and processed by the agent.
- **Agent post time (ms):** the amount of time since the agent's last data sync with the cloud engine.
- **RPC MissedUpdateRequest (req/sec):** the number of missed remote calls to the agent to update details of the request, post execution. Reported in requests per second.
- **Agent missed update time (ms):** the amount of time that has passed since the agent update timed out.

Data Flows

This document demonstrates various data flows between the Module and Agent. While MessagePack is the serialization protocol, the data is displayed here in JSON format for ease of reading.

Benign Post Request

Notice how in `HeadersIn` the `Cookie` value was redacted, and also that `TLSProtocol` and `TLSCipher` are filled in.

```
{
   "ModuleVersion": "sigsci-module-apache 0.214",
   "ServerVersion": "Apache/2.4.7 (Ubuntu) PHP/5.5.9-1ubuntu4.11 OpenSSL/1.0.1f",
   "ServerFlavor": "prefork",
   "ServerName": "soysauce.in",
   "Timestamp": 1438838135,
   "RemoteAddr": "198.51.100.209",
   "Method": "POST",
   "Scheme": "https",
   "URI": "/add-data"
   "Protocol": "HTTP/1.1",
   "TLSProtocol": "TLSv1.2",
   "TLSCipher": "ECDHE-RSA-AES128-SHA256",
   "HeadersIn": [
       [ "Host", "soysauce.in" ],
       [ "Accept", "*/*" ],
       [ "Accept-Language", "en-us" ],
       [ "Referer", "https://soysauce.in/" ],
       [ "Accept-Encoding", "gzip, deflate" ],
   ],
   "PostData": "foo=bar&company=something"
}
```

This request was completely benign, so all that is returned is a `200` response (allow the request to proceed).

```
{
    "WAFResponse": 200
}
```

And that is end of the request.

Benign request (with 404 error)

```
$ curl -v '127.0.0.1:8085/junk'
*   Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 8085 (#0)
> GET /junk HTTP/1.1
> User-Agent: curl/7.37.1
> Host: 127.0.0.1:8085
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content-Type: text/plain; charset=utf-8
< Date: Wed, 05 Aug 2015 18:38:24 GMT
< Content-Length: 19
<
```

would be converted into the following:

```
{
  "ModuleVersion": "sigsci-sdk-golang 1.0",
  "ServerVersion": "go1.4.2",
  "ServerFlavor": "",
  "ServerName": "127.0.0.1:8085",
  "Timestamp": 1438799904,
  "RemoteAddr": "127.0.0.1",
  "Method": "GET",
  "Scheme": "http",
  "URI": "/junk",
  "Protocol": "HTTP/1.1",
  "HeadersIn": [
      [ "User-Agent", "curl/7.37.1" ],
      [ "Accept", "*/*" ],
  ],
}
```

Response is just `200` or allow the response to pass through.

```
{
    "WAFResponse": 200
}
```

The server proceeds normally. If at the end of the request, we find that a error condition occurred or that it had an exceptionally large output or took an exceptionally long time to process, we would followup with a `PostRequest`. Notice how `ResponseCode`, `ResponseMillis`, `ResponseSize` and filled out as well as `HeadersOut`.

```
{
  "ModuleVersion": "sigsci-sdk-golang 1.0",
  "Timestamp": 1438799904,
  "RemoteAddr": "127.0.0.1",
  "Method": "GET",
  "Scheme": "http",
  "URI": "/junk",
  "Protocol": "HTTP/1.1",
  "WAFResponse": 200,
  "ResponseCode": 404,
  "ResponseMillis": 1,
  "ResponseSize": 19,
  "HeadersIn": [
      [ "User-Agent", "curl/7.37.1" ],
      [ "Accept",  "*/*" ],
  ],
  "HeadersOut": [
      [ "Content-Type",  "text/plain; charset=utf-8" ]
  ]
}
```

Blocked Request with SQLI and 406

Here are the raw HTTP headers:

```
$ curl -v '127.0.0.1:8085/junk?id=1+UNION+ALL+SELECT+1'
* Connected to 127.0.0.1 (127.0.0.1) port 8085 (#0)
> GET /junk?id=1+UNION+ALL+SELECT+1 HTTP/1.1
> User-Agent: curl/7.37.1
> Host: 127.0.0.1:8085
> Accept: */*
>
< HTTP/1.1 406 Not Acceptable
< Content-Type: text/plain; charset=utf-8
< Date: Wed, 05 Aug 2015 17:59:46 GMT
< Content-Length: 19
<
406 not acceptable
```

This translates to the following flow.

Server/Module sends the following to the agent:

```
{
  "ModuleVersion": "sigsci-sdk-golang 1.0",
  "ServerVersion": "go1.4.2",
  "ServerFlavor": "",
  "ServerName": "127.0.0.1:8085",
  "Timestamp": 1438796694,
  "RemoteAddr": "127.0.0.1",
  "Method": "GET",
  "Scheme": "http",
  "URI": "/junk?id=1+UNION+ALL+SELECT+1",
  "Protocol": "HTTP/1.1",
  "HeadersIn": [
      [ "Accept",  "*/*" ],
      [ "User-Agent",  "curl/7.37.1" ],
  ],
}
```

The Agent replies with the following. Notice the `RequestID` is filled in, along with an `X-SigSci-Tags` header describing was found (SQLi in this case).

```
  "RequestID": "55c24b96ca84c02201000001",
  "RequestHeaders": [
      [ "X-SigSci-Tags", "SQLI" ]
  ]
}
```

The request should be blocked, and at the end of the request, and `UpdateRequest` message.

```
{
  "RequestID": "55c24b96ca84c02201000001",
  "ResponseCode": 406,
  "ResponseMillis": 1,
  "ResponseSize": 19,
  "HeadersOut": [
      [ "Content-Type", "text/plain; charset=utf-8" ],
  ]
}
```

X-SigSci-* Request Headers

Starting with:

- Agent > 1.8.386
- NGINX Module > 1.0.0+343
- Apache Module > 207

`X-SigSci-` headers are added in the incoming request. The end user (your customers) can not see them. However you internal application can use these headers for various integrations.

Note your module may alter the case (e.g., `X-SigSci-AgentResponse` vs. `X-Sigsci-Agentresponse`) that what is listed here.

X-SigSci-AgentResponse

The agent will return `200` if the request should be allowed, and `406` if the request is blocked.

X-SigSci-RequestID

A request ID used for uniquely identifying this request. May not be present in all requests.

X-SigSci-Tags

A CSV list of signals associated with this request, for example:

- `SQLI`
- `XSS,NOUA`
- `TOR`
- `SITE.CUSTOM-SIGNAL`

This list includes custom signals added by rules. See system signals for a full list of default system signals.

Note that `IMPOSTOR` should not be used at the moment as an indicator of malicious intent. Anything that appears to be a mainstream search engine is tagged with this and the exact identification is done upstream. Improvements in how this is done will be forthcoming.

Agent

Agent Release Notes

4.43.0 2023-06-16

- Improved SQLI detection
- Added support for handling gRPC with gzip encoding
- Fixed v4.42.0 release issue where agent would incorrectly report its version number in some locations
- Updated base GeoIP data: June 14, 2023

- Updated base GeoIP data: June 2023

4.41.0 2023-05-11

- Improved SQLI detection
- Added x86_64 and arm64 support for Amazon Linux 2023
- Added x86_64 and arm64 support for Alpine Linux 3.18
- Upgraded to Golang 1.19.9
- Updated base GeoIP data: May 2023

4.40.0 2023-04-13

- Improved SQLI detection
- Improved RPM, DEB package upgrade scripts

4.39.1 2023-04-03

- Fixed resource leak when loading a new ruleset
- Fixed inspection regression causing internal errors for certain requests
- Improved SQLI detection
- Changed RPM package filenames to not specify `el` version in name

4.39.0 2023-03-15

- Improved detection of XSS
- Improved detection of SQLI
- Upgraded to Golang 1.19.7
- Updated base GeoIP data: March 2023

4.38.0 2023-02-15

- Improved detection of SQLI
- Improved detection of CMDEXE
- Updated base GeoIP data: February 2023

4.37.0 2023-01-12

- Upgraded to Golang 1.19.4
- Improved detection of SQLI
- Updated base GeoIP data: January 2023

4.36.1 2022-12-13

- Improved SQLI detection

4.36.0 2022-12-07

- Add support for Alpine 3.17
- Improved SQLI detection
- Updated base GeoIP data: December 2022

4.35.0 2022-11-09

- Added optional systemd based agent auto update for Debian, Ubuntu, and RHEL/CentOS
- Upgraded to Golang 1.19.3
- Improved GraphQL Parsing
- Improved CMDEXE detection
- Updated base GeoIP data: November 2022

4.34.0 2022-10-12

- Upgraded to Golang 1.19.2
- Improved SQLI detection
- Updated base geo IP data: October 2022

- Fixed HAProxy SPOA health check response
- Improved SQLI detection
- Improved CMDEXE detection
- Upgraded to Golang 1.18.6
- Updated base geo IP data: September 2022

4.32.1 2022-08-24

- Closed GraphQL related vulnerability
- Improved CMDEXE detection

4.32.0 2022-08-17

- Added x86_64 and arm64 support for Red Hat Enterprise Linux(RHEL9)/CentOS Stream 9
- Improved CODEINJECTION detection
- Improved CMDEXE detection
- Improved inspection of HTTP request bodies
- Updated base geo IP data: August 2022

4.31.0 2022-07-13

- Upgraded to Golang 1.18.4
- Improved SQLI detection
- Improved CMDEXE detection
- Updated base geo IP data: July 2022

4.30.0 2022-06-21

- Added Beta Support for AWS Lambda
- Added Fastly CDN endpoint for fetching updates, improving download performance
- Added x86_64 and arm64 support for Ubuntu 22.04
- Updated base geo IP data: June 2022
- Improved inspection for text/plain, CMDEXE, SQLI and Log4j

4.29.0 2022-05-11

- Updated base geo IP data: May 2022
- Improved inspection logic

4.28.0 2022-04-18

- Expanded GraphQL inspection to cover additional data types and anomalous behavior
- Improved XSS detection
- Enhanced inspection of `multipart/form-data`
- Updated base geo IP data: April 2022

4.27.0 2022-03-16

- Added arm64 Linux support and packages for Ubuntu, Debian, and CentOS
- Upgraded to Golang 1.17.8
- Updated base geo IP data: March 2022

4.26.0 2022-02-16

- Improved envoy v3 API compatibility
- Improved reporting of blocked WebSocket messages
- Improved reverse proxy WebSocket header forwarding
- Updated base geo IP data: February 2022

4.25.0 2022-01-19

- Improved reverse proxy `Content-Type` inspection
- Improved reverse proxy gRPC `User-Agent` forwarding

- Improved Content-Type normalization when determining which types to inspect

4.24.0 2021-11-17

- Updated base geo IP data: November 2021

4.23.0 2021-10-21

- Fixed an inconsistency in determining the client IP when `trust-proxy-headers` is disabled and `client-ip-header` was set to the default of using the `X-Forwarded-For` proxy header
- Improved GraphQL query parsing
- Updated base geo IP data: October 2021

4.22.0 2021-09-16

- Added `conn-max-per-host` reverse proxy configuration option to allow limiting the number of upstream connections
- Improved generation of agent cache directory when non path-safe characters are present in the system hostname
- Improved handling of abstract socket namespaces in `rpc-address`
- Upgraded to Golang 1.17.1
- Updated base geo IP data: September 2021

4.21.1 2021-08-16

- Corrected deadlock issue
- Added Debian 11 (bullseye) support (released 2021-09-01)

4.21.0 2021-08-16

- Added external data information to SIGUSR1 diagnostic logs
- Added an experimental `bypass-egress-proxy-for-upstreams` configuration option to more easily exclude reverse proxy upstream traffic from an egress proxy
- Improved external data error handling and metrics
- Standardized release notes
- Updated base geo IP data: August 2021

4.20.0 2021-07-22

- Added initial support for sigsci-module-envoy
- Added Alpine 3.13, 3.14 support
- Improved service lifecycle management, avoiding rare service restarts on agent startup
- Improved geo IP update logic to prevent downgrading to prior versions in specific cases
- Updated external data fetches to honor `download-config-version` option
- Updated base geo IP data: July 2021

4.19.1 2021-06-24

- Fixed permissions for the Unix RPC socket file under stricter `umask` settings

4.19.0 2021-06-23

- Improved handling of log locations when stdout or stderr is used
- Improved CMDEXE detection
- Added support for `application/graphql` content-type for reverse proxy mode
- Updated base geo IP data: June 2021

Apache

Apache Module Release Notes

1.9.2 2023-06-27

- Added additional module configuration for inspection

1.9.0 2022-01-18

- Improved `Content-Type` header inspection
- Added Debian 11 (bullseye) support
- Added Ubuntu 22.04 (jammy) support (2023-01-25)

1.8.5 2021-09-20

- Standardized release notes

1.8.4 2021-07-29

- Added support for `Content-type application/graphql`

1.8.3 2021-02-20

- Added cryptographic signatures to released RPM packages

1.8.2 2021-01-08

- Added Ubuntu 20.04 (Focal Fossa) support
- Removed support for Apache 2.2 32-bit LSB for CentOS 6 (EL6)

1.8.1 2020-07-13

- Added support for setting Location header if agent responds with `X-Sigsci-Redirect`

1.8.0 2020-06-10

- Added support for OPTIONS and CONNECT requests
- Deprecated alternative blocking response codes (`SigSciAltResponseCodes`). Allow any code received from agent, 300 and above as blocking.
- Improved socket error handling and logging

1.7.16 2020-03-06

- Improved handling of headers of larger size returned by agent
- Improved handling of reading from socket when data not ready

1.7.15 2020-03-02

- Added support for configurable agent response codes
- Fixed handling of inspection in Locations

1.7.14 2020-02-24

- Added support for agent response code 429
- Added support for Apache 2.2 32-bit LSB for CentOS 6 (EL6)

1.7.13 2020-02-10

- Fixed agent response parsing errors to get the response code

1.7.12 2020-02-04

- Added Debian 10 (buster) support
- Added CentOS 8 (EL8) support

1.7.11 2019-07-02

- Fixed double send of pre-request to agent

1.7.10 2019-05-07

- Added support for Apache 2.4 for Windows

1.7.8 2019-03-25

- Added `ServerName` field to agent messages

1.7.7 2019-02-15

- Fixed compiler error for CentOS 6 + Apache 2.4

1.7.6 2018-10-03

- Added ability to set `SigSciAgentPostLen` to `0` to turn off post body processing

1.7.5 2018-06-07

- Added ability to send request to agent despite missing TLS parameters

1.7.4 2018-05-23

- Improved error logging when building messages bound for the agent

1.7.3 2018-05-17

- Improved logging across all modules
- Enhanced logging of communication with the agent

1.7.2 2018-05-16

- Added config check for run-list creation
- Updated directive SigSciAgentInspection to be configured per directory and/or globally

1.7.1 2018-05-08

- Hardened apache module to ensure complete logging for errors

1.7.0 2018-05-01

- Added new global directives: `SigSciRunBeforeModulesList` and `SigSciRunAfterModulesList`

1.6.1 2018-04-06

- Standardized release notes
- Porting fixes for Ubuntu 18.04 (Bionic Beaver)
- Ubuntu 18.04 (Bionic Beaver) packaging

1.6.0 2018-1-30

- ISSUE-10307: Allow other modules to run before this one. i.e., `mod_auth_oidc`
-
  ```
      Improved performance and noise reduction per customer request
  ```
-
  ```
      Added new directive: `SigSciEnableFixups`
  ```
-
  ```
      Changed Directive names for all existing Directives to contain prefix `SigSci`
  ```

1.5.7 2018-01-24

- Added support for multipart/form-data post

1.5.6 2017-10-23

- Fixed module version gen script

1.5.5 2017-10-16

- No code changes

- Improved error logs
- Added debugging for specific customer issue

1.5.3 2017-09-11

- Standardized defaults across modules and document

1.5.2 2017-09-01

- Fixed module type

1.5.1 2017-07-24

- Added XML support and inspection
- Upgraded to latest `messagepack` library
- Added Alpine Linux support

1.5.0 2017-03-21

- Redacted

1.4.6 2016-12-02

- Added `.tar.gz` output packages
- Updated external package https://github.com/camgunz/cmp to reduce static analysis noise, no functional changes

1.4.5 2016-10-31

- Fixed error converting timeout from milliseconds to microseconds
- Fixed issue setting socket timeout when >= 1000ms

1.4.4 2016-10-27

- Added ability to allow post-bodies greater than 128k
- Increased default timeout time from 5ms to 100ms similar to NGINX

1.4.3 2016-09-15

- Added support for mod_remoteip over-rides of the client IP address

1.4.2 2016-08-31

- No change, rebuilt to correct version numbers

1.4.1 2016-08-11

- No change, rebuilt to support CentOS 6 + Apache 2.4

1.4.0 2016-07-13

- Switched to SemVer versions
- Added support for Ubuntu 16.04 (Xenial Xerus)

0.344 2016-07-12

- Removed module-level filtering to allow agent features
- Fixed minor packaging issues

0.340 2016-04-15

- Added support for Apache 2.4 on RHEL/CentOS 6

0.338 2016-04-10

- Added support for RHEL/CentOS 5

0.317 2016-02-26

- Originally HTTP methods that were inspected where explicitly listed (allowlisted, e.g. "GET", "POST"). The logic is now inverted to allow all methods not on an ignored list (blocklisted, e.g. "OPTIONS", "CONNECT"). This allows for the detection of invalid or malicious HTTP requests.
- Added backward compatibility support for using the agent RPCv1 protocol (e.g., with `-rpc-version=1`)
- Added the module base address to the startup message to aid debugging EX: SigSci Apache Module version 0.123 starting (base `7f08e4e86000`)
- Improved log messages when reading the request body
- Fixed a potential crash if a request times out

0.311 2016-02-03

- Fixed server crashes as seen in some configurations (so far only in the lab)
- Updated packaging
- Improved performance and memory
- Added support for inspecting HEAD requests

0.241 2015-08-24

- Fixed sending correct values of response code and bytes sent when Apache does certain forms of internal redirects
- Added a Hello World message on Apache start, indicating module is loaded and it's version number
- Improved work around Apache's state machine to capture more response headers

(Originally released as 239, but with minor improvements)

0.224 2015-08-11

HIGHLY RECOMMENDED

- Fixed incorrect handling of (rare) negative length values and time values (due to clock drift, lack of kernel having a monotonic clock, etc)
- Made general optimizations and improvements
- Redacted `Authorization` and `X-Auth-Token` HTTP request headers

0.214 2015-07-31

HIGHLY RECOMMENDED

- Removed incorrect WARNING log message of the form "Allocated buffer using Content-Length of 22 bytes for input stream", which was benign and was turned into a DEBUG message
- Added ability to send Scheme information to agent (i.e. `http` or `https`)
- Added ability to send back TLS (SSL) information to the agent, upgrade agent to at least 1.8.3385 for best results
- Made minor optimizations

0.207 2015-07-20

HIGHLY RECOMMENDED

- Fixed bug in requests with POST bodies > 4000 bytes, where input would get truncated. This bug appeared to manifest itself on some Apache configurations and not others. Regardless, this release is highly recommended for all.
- Added `X-SigSci-AgentResponse`, `X-SigSci-RequestID` request headers, bringing Apache to parity with other platforms
- With Agent 1.8.3186, `X-SigSci-Tags` is added indicating what was detected in the request

0.159 2015-07-13

- Enabled forward compatibility for upcoming feature

0.144 2015-07-06

- Enabled sending of response headers to Agent for upcoming features, which brings the Apache module to parity with other platforms
- Added support and inspect `PATCH` http methods
- Fixed possible issue with reading post bodies > 64k

- Fixed issues where the Signal Sciences dashboard would show a incorrect “Agent Response” of 0. For best results, upgrade Agent to at least 1.8.2718

0.133 2015-06-11

- Major cleanup and bug fix release. Highly recommended for all customers.
- Removed ability to send `Cookie` or `Set-Cookie` headers to the agent
- Removed deprecated communication protocol

CloudFoundry

Signal Sciences for Cloud Foundry

0.1.4 2017-03-21

- Added SIGSCI_REQUIRED variable setting, if true this will prevent the app from starting if the agent fails to start.

0.1.3 2017-03-16

- Added configurable health check feature for both the agent listener and upstream app process.

0.1.2 2017-03-12

- Reset port assignment to ensure app can start if agent fails to start.

0.1.1 2017-03-03

- Agent version can now be specified with the `SIGSCI_AGENT_VERSION` variable.
- Access logging disabled by default.
- Enable access logging by specifying a log file path with the `SIGSCI_REVERSE_PROXY_ACCESSLOG` variable.
- If agent keys are not provided the agent installation process will be skipped.

0.1.0 2017-02-07

- Initial release.
- Package can be extracted directly into existing buildpacks.

Dotnet

SignalSciences .NET Module Release Notes

1.6.1 2021-07-29

- Added support for Content-type application/graphql

1.6.0 2020-09-21

- Removed HTTP method filtering (now inspecting OPTIONS and CONNECT)
- Added support for blocking 300-599 status codes
- Added support for blocking with an HTTP redirect

1.5.5 2020-06-22

- Added support for Nuget packaging

1.5.4 2020-01-07

- Fixed TCP connection leak
- Updated default agent connection pool size changed and set to zero

1.5.3 2019-06-07

- Standardized release notes

1.5.2 2017-12-12

- Removed filterHeaders option
- Added support for multipart form post

1.5.1 2017-09-01

- Fixed module type

1.5.0 2017-04-18

- Fixed issue, now the response size will always be 0 or greater. No more sending -1 in RPC.Post/UpdateRequest
- Fixed issue preventing module from correctly calling RPC.PostRequest when the Agent returns a 406

Dotnet Core

SignalSciences .NET Core Module Release Notes

1.3.1 2023-04-25

- Added configuration option for custom content-types.

1.3.0 2020-08-24

- Added support for setting redirect location
- Added support for blocking on response code range 300 - 599
- Allowed OPTIONS and CONNECT methods

1.2.6 2020-06-18

- Fixed deployment pipeline

1.2.5 2020-06-17

- Added NuGet.org support

1.2.4 2020-02-28

- Added support for HTTP response AsyncFlush

1.2.3 2020-02-07

- Fixed runtime errors when upgraded to .NET Core v3.1

1.2.2 2019-09-09

- Fixed TCP connection leak

1.2.1 2019-06-07

- Fixed handling of xml content type

1.2.0 2019-04-19

- Added netstandard2.0 to TargetFrameworks
- Replaced the package reference for Microsoft.AspNetCore.All with Microsoft.AspNetCore

1.0.1 2018-11-05

- Set default agent connection pool size to zero

1.0.0 2017-10-26

- Initial release

Golang Module Release Notes

1.12.0 2023-01-10

- Replaced internal custom header extractor function with raw header extractor function

1.11.0 2022-01-18

- Improved `Content-Type` header inspection
- Standardized release notes

1.10.0 2021-05-26

- Added support for `application/graphql` content-type

1.9.0 2020-10-22

- Added `server_flavor` config option

1.8.2 2020-06-15

- Updated revision for github actions release

1.8.1 2020-06-15

- Added internal release metadata support

1.8.0 2020-06-15

- Deprecated the `AltResponseCodes` concept in favor of using all codes 300-599 as “blocking”
- Added HTTP redirect support

1.7.1 2020-04-06

- Updated the response recorder to implement the io.ReaderFrom interface
- Fixed some linter issues with missing comments on exported functions

1.7.0 2020-03-11

- Cleaned up configuration and added an `AltResponseCodes` option to configure alternative (other than 406) response codes that can be used for blocking

1.6.5 2020-01-06

- Updated the `http.ResponseWriter` wrapper to allow `CloseNotify()` calls to pass through

1.6.4 2019-11-06

- Updated helloworld example to be more configurable allowing it to be used in other example documentation
- Added the ability to support inspecting gRPC (protobuf) content

1.6.3 2019-09-12

- Added custom header extractor to the post request

1.6.2 2019-08-25

- Added support for a custom header extractor function

1.6.1 2019-06-13

- Cleaned up internal code

1.6.0 2019-05-30

- Updated list of inspectable XML content types
- Added `http.Flusher` interface when the underlying handler supports this interface

1.5.0 2019-01-31

- Switched Update / Post RPC call to async
- Internal release for agent reverse proxy

1.4.3 2018-08-07

- Improved error and debug messages
- Exposed more functionality to allow easier extending

1.4.2 2018-06-15

- Improved handling of the `Host` request header
- Improved debugging output

1.4.1 2018-06-04

- Improved error and debug messages

1.4.0 2018-05-24

- Standardized release notes
- Added support for multipart/form-data post
- Extended architecture to allow more flexibility
- Updated response writer interface to allow for WebSocket use
- Removed default filters on CONNECT/OPTIONS methods - now inspected by default
- Standardized error page
- Updated to contact agent on init for faster module registration

1.3.1 2017-09-25

- Removed unused dependency
- Removed internal testing example

1.3.0 2017-09-19

- Improved internal testing
- Updated msgpack serialization

1.2.3 2017-09-11

- Standardized defaults across modules and document
- Bad release

1.2.2 2017-07-02

- Updated to use signalsciences/tlstext

1.2.1 2017-03-21

- Added ability to send XML post bodies to agent
- Improved content-type processing

1.2.0 2017-03-06

- Improved performance
- Exposed internal datastructures and methods to allow alternative module implementations and performance tests

1.1.0 2017-02-28

- Fixed TCP vs. UDS configuration

0.1.0 2016-09-02

- Initial release

HAProxy Module Release Notes

1.4.0 2023-06-28

- Added new module configuration option for more granular inspection

1.3.1 2022-10-04

- Updated example SPOE configuration files

1.3.0 2022-01-19

- Improved `Content-Type` header inspection
- Improved the URL path and query information sent to agent
- Fixed the scheme information sent to agent (i.e. `http` or `https`)
- Added Ubuntu 20.04 (focal) support

1.2.3 2021-09-13

- Added example SPOE configuration files to communicate with signal sciences agent

1.2.2 2021-07-29

- Added Debian 11 (bullseye) support (2021-08-31)
- Added support for Content-type application/graphql
- Standardized release notes

1.2.1 2021-02-17

- Added cryptographic signatures to released RPM packages

1.2.0 2020-08-11

- Added support for setting redirect location
- Added support for blocking on response code range 300 - 599
- Added support for OPTIONS and CONNECT methods

1.1.12 2020-04-17

- Updated to support HAProxy 1.9 and above
- Added Debian buster support

1.1.11 2020-04-09

- Improved error handling when sending a blocking response

1.1.10 2020-04-06

- Corrected distribution tar file compression
- Added configurable support for custom response header `extra_blocking_resp_hdr` upon 406 responses

1.1.9 2020-02-05

- Added CentOS 8 (el8) support

1.1.8 2020-01-24

- Added explicit socket close

1.1.7 2019-10-03

- Fixed runtime error from method `res_add_header`

1.1.6 2019-06-06

- Fixed handling of xml content-types

- Reduced logging so that expected errors are not logged (set `sigsci.log_network_errors = true` to override)

1.1.4 2018-07-03

- Fixed issue with module not blocking on agent 406

1.1.3 2018-03-09

- Fixed packaging to remove extra directory layer
- Standardized release notes
- Added Ubuntu 18.04 packaging

1.1.2 2018-02-05

- ISSUE-10459 : Enabled timeout tests for module read and agent response

1.1.1 2018-01-12

- ISSUE-10459 : Updated to HAProxy 1.8
- Added support for multipart/form-data post

1.1.0 2017-11-15

- Breaking configuration change. To reduce pollution of the global namespace all `sigsci_XXX` configuration parameters should now be `sigsci.XXX`. No other functional changes.
- Made various minor corrections based on static analysis

1.0.5 2017-11-14

- Fixed bugs

1.0.4 2017-11-07

- Production release

0.0.3 2017-09-11

- Standardized defaults across modules and document

0.0.2 2017-09-07

- Fixed module type

0.0.1 2017-07-02

- Initial - alpha release

Heroku

SignalSciences Buildpack for Heroku Release Notes

0.2.2 2022-09-06

- Improved compatibility of ruby dependency.

0.2.1 2020-11-09

- Added server-flavor option to distinguish buildpack.

0.2.0 2020-06-15

- Added `SIGSCI_HEROKU_BIND_RACE_WORKAROUND=1` configuration to work around a race condition where the app might consume the listener port before the sigsci-agent can start listening
- Fixed the healthcheck not starting and not logging to stderr (enabled with `SIGSCI_HC=true`)
- Cleaned up the startup script and added more debugging output when setting `SIGSCI_HEROKU_BUILDPACK_DEBUG=2`

0.1.10 2020-05-19

- Added support to retry starting the agent on failure
- Added additional debugging on startup when `SIGSCI_HEROKU_BUILDPACK_DEBUG=1`

0.1.9 2018-10-01

- Added healthcheck logic to pass on status of reverse-proxied application
- Standardized release notes

0.1.8 2017-11-14

- Allowed directly specifying the agent download URL via `SIGSCI_AGENT_URL`

0.1.7 2017-10-17

- Added ability to leverage wait-for command during dyno startup to ensure web process starts before the agent starts
- Added handling of port assignment for unicorn app startup command

0.1.6 2017-10-16

- Changed process start order to avoid 502s at dyno start up

0.1.5 2017-03-13

- Updated envronment variable names used to set values in conf file

0.1.4 2017-03-13

- Reset port assignment to ensure app can start if agent fails to start

0.1.3 2017-03-03

- Added ability to specify agent version with the `SIGSCI_AGENT_VERSION` variable
- Disabled access logging by default
- Added ability to enable access logging by specifying a log file path with the `SIGSCI_REVERSE_PROXY_ACCESSLOG` variable

0.1.2 2017-03-02

- Added support for Scala buildpack (proper port assignment)

0.1.1 2017-02-13

- Fixed README url

0.1.0 2017-02-07

- Refactored installation and setup process
- Removed usage of the sigsci reverse proxy binary

IBM Cloud

IBM Cloud Buildpack for Signal Sciences

1.0.2 2016-08-15

- Add start script for php buildpack
- Fix permissions on php start script
- A little script clean up
- Readme updates

1.0.1 2016-08-01

- Fix permissions

IIS

SignalSciences IIS Module Release Notes

3.4.0 2023-06-27

- Added additional module configuration for inspection

3.3.0 2022-09-26

- Updated RPC library.

3.2.0 2022-01-21

- Improved `Content-Type` header inspection
- Standardized release notes

3.1.1 2021-07-29

- Added support for `Content-type application/graphql`

3.1.0 2021-07-16

- Updated installer to not install 32-bit module on Win 2008 Server R2 and Win 7

3.0.0 2021-02-04

- Added improved Azure support for 32-bit, re-releasing as 3.0.0 for 32-bit app pool support in general

2.4.0 2021-01-28

- Added 32-bit app pool support; one installer for 32-bit, 64-bit or mixed app pools. 64-bit OS only.

2.3.0 2020-09-29

- Enhanced debug logging and moved some error level logging to debug level to reduce verbosity
- Added support for reporting of Azure site extension

2.2.0 2020-08-11

- Added support for using all codes 300-599 as “blocking”
- Added HTTP redirect support
- Removed restrictions on HTTP methods
- Fixed an issue where Windows event log entry descriptions were not resolved

2.1.2 2020-06-24

- Fixed an issue when connecting to agent on servers where the localhost resolves to IPv6 address

2.1.1 2020-06-23

- Added support for reading status page path from environment variable

2.1.0 2020-06-22

- Added support for Azure app services
- Added support for reading configuration from environment variables
- Changed log messages destination to standard Windows events

2.0.1 2020-03-05

- Fixed installer when installing on a machine without .NET 3.5 installed by default (e.g., Windows Server 2019)

2.0.0 2020-03-03

uninstalled first. A warning will appear during installation if this is the case.

- Changed default agent rpc-address from port 9999 to port 737 to match the agent default
- Updated the installer to detect non-default agent port configurations (i.e., detect old port 9999 configurations) and configure the IIS module to match
- Replaced the PowerShell utilities with a new `SigsciCtl.exe` utility to aid in manual configuration and diagnostics

1.10.2 2019-12-19

- Fixed handling of IIS application initialization preload requests
- Fixed an issue handling UAC in the installer
- Added a PowerShell script to the install to aid in diagnostics

1.10.1 2019-10-18

- Updated the installer

1.10.0 2019-10-08

- Added a `TimeoutMillis` configuration parameter to configure the inspection timeout
- Updated the installer

1.9.3 2019-06-07

- Fixed handling of xml content type

1.9.2 2019-05-22

- Added signatures to packages and DLL

1.9.0 2019-01-29

- Fixed race condition causing potential crash in RPC processing

1.8.0 2019-01-10

- Updated RPC library

1.7.3 2018-11-08

- Fixed race condition
- Improved logging
- Added config options `agentHost`, `MaxPostSize`, `AnomalySize` and `AnomalyDurationMillis`
- Default RPC version changed and set to RPCv0

1.7.2 2018-05-08

- Updated MSI installer to avoid installing for unsupported 32-bit application pools

1.7.1 2018-03-22

- Added MSI installer
- Standardized release notes

1.7.0 2018-02-02

- Fixed race condition

1.6.7 2018-02-01

- Added config options

1.6.6 2018-01-23

- Added support for multipart/form-data post
- Added debug logging option
- Fixed module registration priority

- Changed it to always send sensitive headers to agent, agent redacts sensitive headers

1.6.4 2017-09-11

- Standardized defaults across modules and document

1.6.3 2017-09-01

- Fixed module type

1.6.2 2017-04-17

- Fixed a bug where the response time for blocked requests was -1ms

1.6.1 2017-04-17

- Fixed a bug where a request that received a 406 from the agent would not call RPC.PostRequest

1.6.0 2017-04-16

- Added a stats page so you can easily see the module's various internal performance counters (request counts, error counts, RPC call counts, RCP call timing information). The page is disabled by default. To enable it, you'll need to follow the configuration instructions in README.md.

Java

Java Module Release Notes

2.5.2 2023-06-28

- Added additional module configuration for inspection

2.5.1 2022-06-02

- Fixed multipart form parsing bug with spring boot

2.5.0 2022-04-07

- Added compatibility for Jakarta

2.4.5 2022-02-14

- Improved utilization of CPU and memory resources

2.4.0 2022-01-18

- Improved `Content-Type` header inspection
- Added support for Servlet 3.0 `getParts()`, `getPart()` APIs.

2.3.0 2021-08-31

- Removed dependencies from Apache `http-core` and `http-client` to address potential security vulnerabilities

2.2.4 2021-06-15

- Improved rethrowing application exceptions in container
- Added support for `Content-type application/graphql`

2.2.3 2021-03-22

- Added bypass options by CIDR block, IP range, path or hostname

2.2.2 2020-11-10

- Fixed a bug with reading integer headers

2.2.0 2020-08-17

- Fixed an issue where query parameters added during the forward to JSP page or another servlet are missing

2.1.4 2020-07-27

- Added support for redirect, blocking and allowing options and connect

2.1.3 2020-04-02

- set thread pool and queue size

2.1.2 2020-03-03

- Improved support for Servlets 3.1 async features
- Added support for configurable agent response codes

2.1.1 2020-02-25

- Added support for agent response code 429

2.1.0 2020-02-13

- Added support for Servlets 3.1 async features
- Fixed an issue where module caused agent traffic spike at the start of stress tests

2.0.4 2020-02-04

- Fixed an issue where HTTP response header with multiple values caused an exception in RPC post request

2.0.3 2020-01-27

- Fixed an issue where Unix socket close caused RPC errors

2.0.2 2019-12-04

- Fixed a rare null pointer exception error in RPC post request
- Fixed an issue where `null` HTTP header value is returned instead of an empty string
- Improved debug log

2.0.0 2019-11-21

Introducing version 2.0 of the Signal Sciences Java module. This release includes a 2x performance improvement and better utilization of memory resources. JAR dependencies have been updated and isolated to work in more environments. No configuration changes are required. As is best practice, it's advised to deploy in a staging environment before production. The specifics of the optimizations are as follows:

- Created shaded jar file with no dependencies and moved all packages to `signalsciences` namespace
- Fixed RPC connections tracking code that was running in O(n) time
- Minimized temporary buffers usage during (de)serialization, reading and writing of `msgpack` data to sockets
- Minimized number of buffers used to cache the post body and avoided unnecessary copying
- Minimized reflection usage to (de)serialize Java objects to/from `msgpack` stream

1.2.0 2019-05-03

- Added support for Netty
- Fixed a rare unix connection leak
- Reduced logging around RPC connection errors

1.1.3 2019-03-07

- Added config option `expectedContentTypes` that can accept space separated media types and these additional media types are added to the list of valid content types checked by the module before sending the post body to agent for inspection

- Changed to parse post body only if content-type is `application/x-www-form-urlencoded`
- Fixed an issue where module reported invalid version 1.X

1.1.1 2019-01-25

- Added config option to work around missing post body when asynchronously handling request

1.1.0 2018-10-31

- Updated jars to match maven conventions
  - `sigsci-module-java-{version}.jar` contains the module classes without dependencies (see `pom.xml`)
  - `sigsci-module-java-{version}-shaded.jar` bundles dependencies following maven shaded classifier `<classifier>shaded</classifier>`
- Updated dependencies to latest
- Fixed a rare issue where an exception would cause the filter chain to be called twice

1.0.5 2018-10-04

- Fixed an issue where a null header name or value would cause an exception

1.0.4 2018-09-28

- Fixed a rare error handling case that could have resulted in leaked open connections

1.0.3 2018-06-27

- Added debug for filter conflict errors

1.0.2 2018-01-26

- Added support for multipart/form-data post
- Fixed class loader issue with multiple versions of `asm.jar`
- Updated default `sigsci-agent` unix socket

1.0.1 2017-09-08

- Fixed module type
- Fixed default RPC timeout and max post size

1.0.0 2017-08-07

- Bumped version

0.4.0 2017-08-03

- Added support for java servlet filter

0.3.0 2017-04-05

- Added ability to forward XML-like post bodies to agent
- Revamped TCP RPC
- Initial Unix RPC

0.2.0 2017-03-06

- Fixed issue; reading post content via `getInputStream`, `getReader` and `getHeader*` should behave the same as Jetty
- Changed module defaults to be consistent with other Signal Sciences modules

0.1.6 2017-02-10

- Added support for jetty 9.3.x and 9.4.x

0.1.5 2016-09-30

- Added source for jetty handler to serve as an example

0.1.3 2016-09-19

- Reduced logging around failures to reconnect to agent

0.1.2 2016-09-16

- Added simple example server with source to packages

0.1.1 2016-09-15

- Added javadoc packages

0.1.0 2016-09-08

- Initial beta release

NGINX

NGINX Module Release Notes

1.6.0 2023-04-17

- Added new module configuration option for more granular inspection

1.5.1 2022-10-17

- Added support for Kong 3.0
- Added support for Ubuntu 22.04 (Jammy) (2023-01-17)

1.5.0 2022-01-19

- Improved `Content-Type` header inspection

1.4.3 2021-07-29

- Added support for `Content-type application/graphql`
- Standardized release notes (2021-08-31)
- Added Debian 11 support (2021-08-31)

1.4.2 2021-03-10

- Added checksum to `sigsci-module-nginx.tar.gz`

1.4.1 2021-02-18

- Added cryptographic signatures to released RPM packages

1.4.0 2020-06-25

- Added ability to pass OPTIONS, CONNECT, and all http methods to the agent
- Added ability to allow any waf response code received from agent, 300 to 599 as blocking
- Added support for setting `Location` header if agent responds with `X-Sigsci-Redirect`
- Added Ubuntu 20.04 (Focal Fossa) support (2020-09-07)

1.3.1 2020-01-30

- Added Debian 10 (buster) support
- Added CentOS8 (EL8) support

1.3.0 2019-07-12

- Updated module to identify rewritten PreRequests

1.2.9 2019-06-18

- Updated module to identify PreRequests

1.2.7 2019-05-23

- Fixed handling of XML content-type to ensure POST body will be read

1.2.6 2018-10-01

- Added NGINX environment override `SIGSCI_NGINX_DISABLE_JIT` to disable the JIT
- Added explicit socket close

1.2.5 2018-06-28

- Fixed handling of bad json elegantly rather than error exception

1.2.4 2018-04-26

- Added option to reuse TCP or Unix socket connection when agent `-rpc-version=1` is used

1.2.3 2018-04-06

- Added Ubuntu 18.04 (Bionic Beaver) package

1.2.2 2018-03-27

- Added Kong plugin
- Added Debian 9 (stretch) package

1.2.1 2018-01-30

- Added support for multipart/form-data post

1.2.0 2017-10-07

- Improved logging
  - Debug logging performance penalty minimized
  - Ad-hoc data is now JSON encoded for clarity and safety
  - Each message is tagged with `NETWORK`, `DEBUG` or `INTERNAL`
- Updated third-party dependencies to latest
  - `rxi/json.lua`
  - `fperrad/lua-MessagePack`
- Standardized defaults across modules and document

1.1.8 2017-09-01

- Fixed module type

1.1.7 2016-12-12

- Disabled debug log by default

1.1.6 2016-12-09

- Cleaned up `log_debug` output

1.1.5 2016-11-30

- Cleaned up network error logging
- Added `log_debug` option to aid in debugging
- Added ability to detect and warn for non-LuaJIT installs due to recent compatibility issues

1.1.4 2016-09-01

- Disabled exit if NGINX returns the HTTP method as nil

1.1.2 2016-07-20

- Added new download option at https://dl.signalsciences.net/sigsci-module-nginx/sigsci-module-nginx_latest.tar.gz

1.1.1 2016-07-14

- Added support for Ubuntu 16.04 (Xenial Xerus)

1.1.0 2016-07-13

- Changed default socket to `/var/run/sigsci.sock` to allow systemd to work without reconfiguration
- Allowed XML mime types to be passed through to Agent, which allows the Agent to inspect XML documents
- Removed header filtering, as that is now down in the agent, which allows custom rules and other actions on cookie data
- Updated `lua-MessagePack` to latest
- Fixed NGINX validator script

1.0.0+428 2016-03-16

- Added license information to packages
- Fixed version reporting bug

1.0.0+424 2016-03-15

- Cleaned up some error messages surrounding timeouts
- Fixed bug reading agent responses when `-rpc-version=1` is used
- Built additional package formats

1.0.0+417 2016-03-07

- Fixed bug with version reporting in dashboard

1.0.0+416 2016-02-26

- Added backward compatibility support for using the agent RPCv1 protocol (e.g., with `-rpc-version=1`)

1.0.0+411 2016-02-17

- Originally HTTP methods that were inspected where explicitly listed (allowlisted, e.g. “GET”, “POST”). The logic is now inverted to allow all methods not on an ignored list (blocklisted, e.g. “OPTIONS”, “CONNECT”). This allows for the detection of invalid or malicious HTTP requests.

1.0.0+408 2016-02-03

- Implemented packaging fixes

1.0.0+407 2016-01-27

- Added support for inspecting HEAD requests
- Improved return speed if post request has an invalid method

1.0.0+388 2015-11-10

- Made network and internal error logging configurable, with network error logging off by default, which will help prevent flooding web server logs with messages if the agent is off or not running
- Allowed “subrequest processed” used in certain configurations of NGINX

1.0.0+378 2015-10-07

- Improved error handling and standardized error message format

1.0.0+369 2015-09-15

- Added ability to optionally allow a site access key to be specified in `prerequest` and `postrequest` functions

1.0.0+361 2015-08-17

This was a maintenance release with general improvements

- Added new feature on startup to send a `notice` message in the error log describing the components used in the module
- Upgraded pure-Lua MessagePack to 0.3.3, which contains minor performance improvements and allows use of various Lua tool chains
- Allowed module to run using plain Lua (not LuaJIT). We strongly recommend LuaJIT as using plain Lua may have severe performance issues. However this does allow options for very low volume servers and aids in debugging.
- Added ability to ensure response time value is non-negative (on machines lacking a monotonic clock and/or clock drift, the value can occasionally go negative)
- Made minor performance improvements and API standardization

1.0.0+346 2015-07-31

- Added ability to send Scheme information to agent (i.e. `http` or `https`)
- Added ability to send TLS (SSL) protocol and cipher suite information to agent, upgrade agent to at least 1.8.3385 for best results

1.0.0+344 2015-07-21

- Improved clarity when NGINX is misconfigured

1.0.0+343 2015-07-13

- Enabled setting of request headers from Agent response, requires Agent 1.8.3186 and greater
- Added `X-SigSci-RequestID` and `X-SigSci-AgentResponse` request headers, allowing integration with other logging systems
- Fixed “double signal” issue first noticed in 1.0.0+320

1.0.0+327 2015-07-07

- Fixed compatibility to support NGINX version 1.0.15

1.0.0+322 2015-07-06

- Added support for inspection of HTTP `PATCH` method

1.0.0+320 2015-06-14

- Fixed issues where the Signal Sciences dashboard would show an incorrect “Agent Response” of 0 (for best results, upgrade Agent to at least 1.8.2718)

Known Issues (fixed in 1.0.0+343)

- Requesting a static file, or a missing file, that results with a custom error page may result in “double signal” on the dashboard (i.e., one request generates two entries). This is due to a bug in the NGINX state machine with custom error pages. We are actively working to find a solution.

1.0.0+315 2015-06-11

- Updated to bring module up to latest API specification to enable future features

---

NGINX 1.10 Lua Module

nginx110-lua-module release notes

2.3.2 2017-04-17

- Add amazonlinux 2016.09 package

2.3.1 2017-03-07

- Add epel 6,7 packages

2.3.0 2017-02-16

- Add debian8 packages
- Upgrade lua-nginx-module to 0.10.7

2.2.0 2016-11-02

- Upgrade to 1.10.2

2.1.0 2016-09-13

- Major upgrade, 2.1.0 to indicate working with nginx 1.10.0 to 1.10.1

1.10.1.2 2016-09-09

- CentOS 6 support

1.10.1.1 2016-08-23

- Initial

NGINX 1.11 Lua Module

nginx111-lua-module release notess

2.7.0 2017-03-21

- Add 1.11.8,9,10
- update configure flags for >= 1.11.5 to use –with-compat

2.6.1 2016-12-23

- Add debian8 packages

2.6.0 2016-11-21

- Upgrade to nginx 1.11.6
- Upgrade ngx-lua to 0.10.7

2.5.0 2016-11-02

- Upgrade to 1.11.5

2.4.0 2016-09-13

- Major upgrade, 2.4.0 supports 1.11.0 to 1.11.4

1.11.3.2 2016-09-09

- CentOS 6 support

1.11.3.1 2016-09-07

- Initial

NGINX 1.12 Lua Module

NGINX 1.12 Lua Module Release Notes

1.1.3 2019-09-24

- add el/7 builds for amazonlinux

1.1.2 2018-06-22

- add epel builds for centos7

1.1.0 2018-05-03

- Updated lua-nginx-module to 0.10.13
- Added debian 9 (stretch) package
- Added ubuntu 18.04 (bionic) package
- Standardized release notes

1.0.3 2017-10-23

- Added 1.12.2 to build matrix

1.0.2 2017-10-05

- Added amazonlinux2017.09 to matrix

1.0.1 2017-07-22

- Added per-point version packages
- Added jenkins build_number as iteration (release in rpm terms)

1.0.0 2017-07-12

- First build for nginx 1.12.1
- lua-nginx-module 0.10.8
- LuaJIT 2.0.5

NGINX C Binary

NGINX C Binary Module Release Notes

1.1.7 2022-09-13

- Fixed memory leak in the event of a pre-request failure (released 2022-09-13)
- Added support for NGINX 1.25.0 on Alpine Linux 3.13 - 3.17, Amazon Linux 2 LTS, Amazon Linux 2023, Centos 7 - 9, Debian 10, Debian 11, Ubuntu 18.04 LTS - 22.04 LTS (release on 2023-05-30)
- Added support for NGINX Plus Release 28 (R28) and Release 29 (R29) on AmazonLinux 2023 (released 2023-05-17)
- Added support for Alpine Linux 3.18 (amd64 and aarch64) NGINX versions 1.20.2 - 1.24.0 and NGINX Plus R28 & R29 (released 2023-05-16)
- Added support for NGINX Plus Release 29 (R29) on Alpine Linux 3.13, 3.14, 3.15, 3.16, 3.17, Amazon Linux 2 LTS, CentOS 7.4+, Debian 11, Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS (released on 2023-05-11)
- Added support for NGINX 1.23.4 and 1.24.0 on Alpine Linux 3.14 - 3.17, Amazon Linux 2 LTS, Centos 7 - 9, Debian 11, Ubuntu 18.04 LTS - 22.04 LTS (release on 2023-04-11)
- Added support for NGINX 1.23.3 on Debian 11 (released 2023-01-11)
- Added support for NGINX 1.23.3 (released 2023-01-06)
- Added support for NGINX Plus Release 28 (R28) on Alpine Linux 3.13, 3.14, 3.15, 3.16, 3.17, Amazon Linux 2 LTS, CentOS 7.4+, Debian 11, Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS (released on 2022-11-30)
- Added support for NGINX 1.14.2, 1.16.1, 1.18.0, 1.20.2, 1.22.1, 1.23.2, and NGINX Plus Release 27 (R27) on Alpine 3.17 (released 2022-11-30)
- Added support for NGINX 1.14.2, 1.16.1, 1.18.0, 1.20.2, 1.22.1, 1.23.2, and NGINX Plus Release 27 (R27) on CentOS stream9 / RHEL 9 (released 2022-11-21)
- Added support for NGINX 1.22.1 and 1.23.2 on Alpine (3.13 - 3.16), Amazon Linux 2, CentOS (7 & 8), Debian 11, Ubuntu (18.04, 20.04, 22.04) (released 2022-11-01)

- Added support for NGINX 1.19.0,1.19.1,1.19.2,1.19.3,1.19.4,1.19.5,1.19.6,1.19.7,1.19.8,1.19.9,1.19.10,1.20.0,1.20.1,1.20.2,1.21.0,1.21.1,1.21.3,1.21.4,1.21.5,1.21.6,1.22.0,1.23.0 on CentOS 7 (released 2022-10-25)

- Added support for NGINX Plus Release 25 (R25) on CentOS 7, CentOS 8, and Amazon Linux 2 (released 2022-10-24)

- Added support for NGINX Plus Release 26 and 27 (R26 and R27) on CentOS 7 and CentOS 8 (released 2022-10-05)

- Added support for AArch64 Alpine Linux 3.14, 3.15, 3.16 with NGINX (1.16.0-1.23.0) (released 2022-09h20)

1.1.6 2022-04-14

- Improved WebSocket messages inspection
- Added support for NGINX Plus Release 26 (R26) (released 2022-04-19)
- Added support for Alpine 3.15 (released 2022-04-22)
- Added support for Alpine 3.14 (released 2022-05-12)
- Added support for Alpine 3.16 (NGINX 1.22.0 only) (released 2022-05-31)
- Added support for Ubuntu 22.04 (NGINX 1.22.0 only) (released 2022-05-31)
- Added support for NGINX 1.22.0 (released 2022-05-31)
- Added support for NGINX 1.23.0 (released 2022-07-07)
- Added support for NGINX Plus Release 27 (R27) (released 2022-07-11)
- Added Arm64 support for NGINX 1.21.6,1.22.0,1.23.0 for CentOS 7,8 (released 2022-07-18)
- Added Arm64 support for NGINX Plus Release 25 (R25) and Release 27 (R27) support for Amazon Linux 2, CentOS 7,8, Ubuntu 18.04, 20.04, 22.04, Debian 10,11 (released 2022-07-18)
- Added support for Amazon Linux 2(NGINX 1.23.0 only) (x86-64 and arm64) (released 2022-07-18)
- Added support for Ubuntu 20.04 (focal) (NGINX 1.20.1) (x86-64 and arm64) (released 2022-07-21)
- Added support for NGINX 1.18.0,1.19.0,1.19.1,1.19.2,1.19.3,1.19.4,1.19.5,1.19.6,1.19.7,1.19.8,1.19.9,1.19.10,1.20.0,1.20.1,1.20.2,1.21.0,1.21.1,1.21.3,1.21.4,1.21.5,1.21.6,1.22.0 on Ubuntu 22.04 (x86-64 and arm64) (released 2022-07-21)
- Added support for NGINX 1.18.0,1.19.0,1.19.1,1.19.2,1.19.3,1.19.4,1.19.5,1.19.6,1.19.7,1.19.8,1.19.9,1.19.10,1.20.0,1.20.1,1.20.2,1.21.0,1.21.1,1.21.3,1.21.4,1.21.5,1.21.6 on Alpine 3.16 (x86-64) (released 2022-07-22)
- Added support for NGINX Plus Release 26 (R26) (released 2022-08-25)

1.1.5 2021-05-17

- Added support for NGINX 1.19.10 on Alpine 3.14 (released 2022-03-07)
- Added Arm64 support for NGINX 1.18.0,1.19.4,1.19.5,1.19.6,1.19.7,1.19.8,1.19.9,1.19.10 on Ubuntu 18.04, 20.04 and Debian 10,11 (released 2022-03-25)
- Added Arm64 support for NGINX 1.20.0,1.20.1,1.20.2,1.21.0,1.21.1,1.21.3 on Ubuntu 18.04, 20.04 and Debian 10,11 (released 2022-03-15)
- Added support for Debian 10 (buster) NGINX 1.14.2 (released 2021-09-28)
- Standardized release notes (2021-09-01)
- Added support for Debian 11 (bullseye) NGINX 1.18.0 (released 2021-09-01)
- Added support for Debian 9 and backports NGINX 1.14.1
- Added support for CentOS 7 & 8 EPEL versions of NGINX
- Added support for NGINX Plus Release 24 (R24)
- Added support for NGINX 1.19.7, 1.19.8, 1.19.9, 1.19.10, 1.20.0, 1.20.1, 1.21.0, 1.21.1, 1.21.2, 1.21.3 and 1.21.4
- Added cryptographic signatures to released RPM packages
- Added support for Alpine 3.13 and Alpine 3.14
- Added support for NGINX Plus Release 25 (R25)
- Added support for NGINX 1.20.2 (released 2021-11-16)
- Added support for NGINX 1.21.6 (released 2022-02-15)
- Added support for NGINX 1.21.5 Alpine Linux (3.11, 3.12, 3.13, 3.14), Debian (10, 11), Ubuntu (18.04, 20.04) (2022-04-12)

1.1.4 2021-01-13

- Fixed a rare issue where module failed to add request headers received from the agent
- Added support for NGINX Plus Release 23 (R23)
- Added support for Ubuntu 20.04 (Focal Fossa)
- Added support for NGINX 1.19.6

1.1.2 2020-10-05

- Fixed a rare HTTP POST request timeout issue when the external authentication used

1.1.1 2020-09-10

- Fixed a rare HTTP/2 request timeout issue when the external authentication used
- Released packages for NGINX 1.19.3 (2020-10-01)

1.1.0 2020-08-27

- Fixed processing of HTTP/2 requests that may result in `-2` agent responses
- Fixed handling of internal HTTP/2 request
- Fixed a rare HTTP request timeout issue when the external authentication used

1.0.46 2020-07-10

- Fixed crash for HTTPS request with malformed or HTTP/0.9 type header line
- Released packages for NGINX 1.19.1 and 1.19.2

1.0.45 2020-07-08

- Added support for setting `Location` header if agent responds with `X-Sigsci-Redirect`

1.0.44 2020-06-15

- Added ability to pass non-406 WAF blocking response codes from the agent
- Added support for Amazon Linux 2
- Added support for NGINX 1.10.3-fips for Ubuntu 16.04 (Xenial Xerus)
- Added support for NGINX 1.19.0 and NGINX Plus Release 22 (R22)

1.0.43 2020-05-11

- Added support to inspect WebSockets

1.0.42 2020-04-21

- Released packages for NGINX 1.18.0 stable
- Released packages for NGINX 1.17.10
- Removed support for Ubuntu 19.04 in favor of 19.10 as per https://wiki.ubuntu.com/DiscoDingo/ReleaseNotes

1.0.41 2020-04-07

- Released packages for NGINX Plus Release 21 (R21)

1.0.40 2020-03-30

- Added support for sigsci-nginx-ingress-controller

1.0.39 2020-03-23

- Released packages for NGINX 1.17.9

1.0.38 2020-03-11

- Added Alpine Linux support

1.0.37 2020-02-19

- Fixed UDS path length check

1.0.36 2020-02-11

- Added CentOS (EL8) support

1.0.34 2020-01-17

- Fixed dependency ordering issue with the NGINX NDK

1.0.33 2020-01-02

- Released packages for NGINX 1.17.7

1.0.32 2019-12-04

- Released packages for NGINX Plus Release 20 (R20)
- Fixed installers to avoid interfering with existing NDK module installs

1.0.31 2019-11-21

- Updated to log RPC errors in detail
- Updated to use latest NGINX Development Kit (NDK) - version 0.3.1

1.0.30 2019-11-19

- Released packages for NGINX 1.17.6
- Updated source to build with NGINX < 1.13.4

1.0.30 2019-10-10

- Released packages for NGINX 1.17.4

1.0.29 2019-09-12

- Built NGINX and NGINX Plus as EL6 for Amazon Linux image 2018.03

1.0.28 2019-09-12

- Fixed nginx-org build for Amazon Linux image 2018.03

1.0.27 2019-09-06

- Released packages for NGINX Plus Release 19 (R19)

1.0.26 2019-09-05

- Fixed sending post-msg request to agent even when missing context
- Added support for Debian 10 buster

1.0.25 2019-08-30

- Added support for Amazon Linux image 2018.03

1.0.24 2019-08-22

- Fixed post to handle invalid content-length and chunked requests

1.0.23 2019-08-14

- Released packages for NGINX 1.16.1 and 1.17.3

1.0.22 2019-08-07

- Released packages for NGINX 1.14.1 and 1.17.2

1.0.21 2019-08-06

- Fixed handling of internal requests

1.0.20 2019-07-09

- Released packages for NGINX 1.12.2

1.0.18 2019-06-13

- Eliminated sending of duplicate messages to agent

1.0.17 2019-06-05

- Released packages for NGINX 1.17.0

1.0.16 2019-06-03

- Released packages for NGINX 1.16.0
- Added support for Ubuntu 19.04 (Disco Dingo)

1.0.15 2019-05-22

- Released packages for NGINX 1.15.3

1.0.14 2019-04-22

- Released packages for NGINX Plus Release 18 (R18) (1.15.10)

1.0.13 2019-04-18

- Released packages for NGINX 1.15.12

1.0.12 2019-04-10

- Updated dependencies for CentOS packages

1.0.11 2019-04-03

- Released packages for NGINX 1.15.10

1.0.10 2019-03-30

- Fixed TLS parameter interrogation

1.0.9 2019-03-27

- Fixed handling of missing host header value

1.0.8 2019-03-15

- Released packages for NGINX 1.15.7, 1.15.8, and 1.15.9
- Released package for NGINX Plus Release 17 (R17) (1.15.7)

1.0.7 2019-02-26

- Set rewrite phase as default

1.0.6 2019-02-20

- Added support for rewrite phase processing

1.0.5 2019-01-29

- Updated package for NGINX Plus with dependency `nginx-plus-module-ndk` - NGINX Plus Release 17 (R17)
- Cleaned up package uninstall script

1.0.4 2019-01-28

- Removed (nginx.org)ndk lib from NGINX Plus - NGINX Plus Release 17 (R17)

1.0.3 2018-12-19

- Re-certified with latest release - NGINX Plus Release 16 (R16)

1.0.1 2018-11-28

- Updated config checks for port and time values
- Updated READMEs for install

1.0.0 2018-11-01

- Built packages for NGINX 1.15.2 and NGINX Plus

NodeJS

Node.js Module Release Notes

Unreleased

2.1.3 2022-12-09

- Pruned dependencies to remove stale references

2.1.2 2022-06-13

- Pruned dependencies to remove stale references

2.1.1 2022-02-23

- Fixed logging bug for post and update inspection steps

2.1.0 2022-01-18

- Improved `Content-Type` header inspection

2.0.2 2021-10-05

- Fixed issue with post body processing for NodeJS v16

2.0.1 2021-09-27

- Fixed debug logging bug

2.0.0 2021-09-13

- Refactored sigsci.js to allow the addition of new web frameworks without code duplication
- Standardized release notes

1.6.4 2021-03-25

- Added requirement of at least msgpack5 3.6.1 explicitly to address CVE-2021-21368

1.6.3 2020-09-17

- Fixed timeout error logging

1.6.2 2020-09-15

- Updated dependencies

1.6.1 2020-08-03

- Fixed logging bug

1.6.0 2020-07-30

- Added support for Hapi v17

1.5.2 2020-03-23

- Added null check for response headers

1.5.1 2019-10-17

- Added support for Hapi v18 testing framework

1.5.0 2019-09-26

- Added Hapi v18 support

1.4.8 2019-02-08

- Fixed possible `multipart/form-data` post body corruption

1.4.7 2018-01-29

- Added support for `multipart/form-data` post

1.4.6 2017-09-19

- Added option to enable debug log

1.4.5 2017-08-23

- Fixed module type

1.4.4 2017-04-26

- Fixed possible race condition

1.4.3 2017-03-22

- Added ability to forward XML-like post bodies to agent

1.4.2 2017-03-07

- Added ability to close connection on `UpdateResponse` and `PostResponse` callback

1.4.1 2017-03-06

- Prevented crashing in some error handling cases
- Fixed bug that caused invalid RPC requests to be sent to the Signal Sciences agent
- Trimmed whitespace around header values
- Updated third-party dependencies in shrinkwrap

1.4.0 2017-02-10

- Improved logging
- Improved jshint static analysis
- Updated third-party dependencies in shrinkwrap

1.3.2 2017-02-09

- Fixed configuration of TCP/IP vs UDS

1.3.1 2016-09-15

- Improved handling of TLS and null pointer issue for Hapi

1.3.0 2016-08-15

- Added initial Hapi support
- Corrected code to conform to standard

- Made no changes, released to improve download experience

1.2.0 2016-07-13

- Removed header filtering from module, as this is now done in the agent
- Improved packaging

1.1.1 2016-05-27

- Fixed issue where the remote socket address was not set correctly

1.1.0 2016-05-12

- Standardized support for nodejs.express to behave like other express middleware
- Added support for Restify
- Fixed minor cosmetic issues to log messages, and code simplification

1.0.1 2016-05-05

- Fixed support for nodejs.express
- Improved timeout error messages

1.0.0 2016-05-02

- Initial release

Signal Sciences Help Center