Signal Sciences
Now part of fastly

Signal Sciences Documentation Archive

Edge Deployment

> **Note:** This information is part of a Limited Availability release. For more information, see our product and feature lifecycle descriptions.
>
> The Edge deployment method currently only supports the features of the Essential platform. Features included with the Professional or Premier platforms are not supported.

About 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.

Requirements

- A Signal Sciences corp and site.
- One or more Fastly services.
- Ensure your Fastly account is enabled for the edge security service. Contact support@fastly.com for additional information.
- A custom VCL file that you've set as the main VCL file.

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 and modifying VCL on the Fastly service.

Creating the edge security service

Create a new edge security service by calling the edgeDeployment 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:$SIGSCI_EMAIL" -H "x-api-token:$ACCESS_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

Map your corp and site to an existing Fastly service and synchronize the origins by calling the edgeDeployment/{fastlySID} API endpoint. You will need to replace `{fastlySID}` with the ID of the Fastly service.

> **Note:** This API call requires Fastly-Key for authentication. The Fastly API key must have write access to the Fastly Service ID.

```
curl -H "x-api-user:$SIGSCI_EMAIL" -H "x-api-token:$ACCESS_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}
```

Run this API call again for each Fastly service you want to deploy on. If your origins change, you will need to run this API call again to resynchronize the backends.

This API call makes changes and adds a new `sigsci_config` custom VCL file to your Fastly service. After making the API call, these changes will be left in an unactivated draft version. Activate the draft service version for the changes to take effect.

Calling the edge security service

You will need to call the new `sigsci_config` VCL file for your Fastly service to load it. Add the following line to your main VCL file:

```
include "sigsci_config";
```

Then add the following line to both the `vcl_miss` and `vcl_pass` subroutines of your service to call the edge security service.

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:

Upgrading Introduction

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

Cloud WAF Overview

What is Cloud WAF?

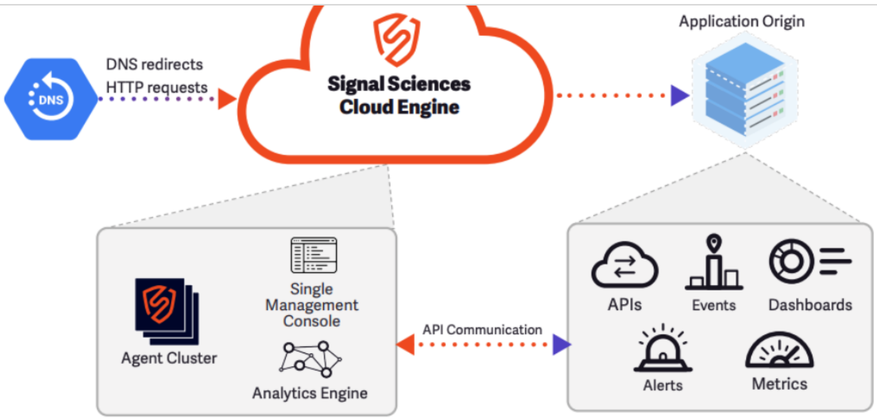
Cloud WAF is a hosted solution designed for customers that may not have full autonomy over their infrastructure and therefore do not wish to install a Signal Sciences agent and module into their respective environments.

For environments such as these, Cloud WAF is an easily deployable option that provides the same full security capabilities of other Signal Sciences agent-based deployment options with matching actionable insights, reporting and DevOps tool integrations.

Uniquely compared to other Cloud WAFs that customers may be used to, the Signal Sciences Cloud WAF shares a unified management console with all other deploymention 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.

What is required?

The install process for Cloud WAF can be completed in seconds, the only requirements for deployment are:

1. A simple DNS change
2. Your application's TLS/SSL certificate information

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/CNames at the apex (or root) of the domain. If your DNS provider does not support this common feature set, we can recommend several DNS providers based on your implementation. Reach out to your rep for more information.

Announcements

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, and is especially powerful for connecting the Fastly edge and the Fastly Next-Gen WAF (powered by Signal Sciences). The feature is available for Professional and Premier platform customers.

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.

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

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.

This page is your one-stop-shop to find information about what we're calling "Observed IPs." There are three stateful IP statuses we represented on lists: **Suspicious IPs**, **Flagged IPs**, and **Rate Limited IPs**. Now, you can find all of these lists in one convenient view.

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 lookback.

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 UI. 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 jumpstart 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 (ie. 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

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
- Corp Owners can view info about the tokens (like creator and IP), as well as info related to the changes above, like expiration, status (Disabled by Owner, Expired, Active)
- 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.

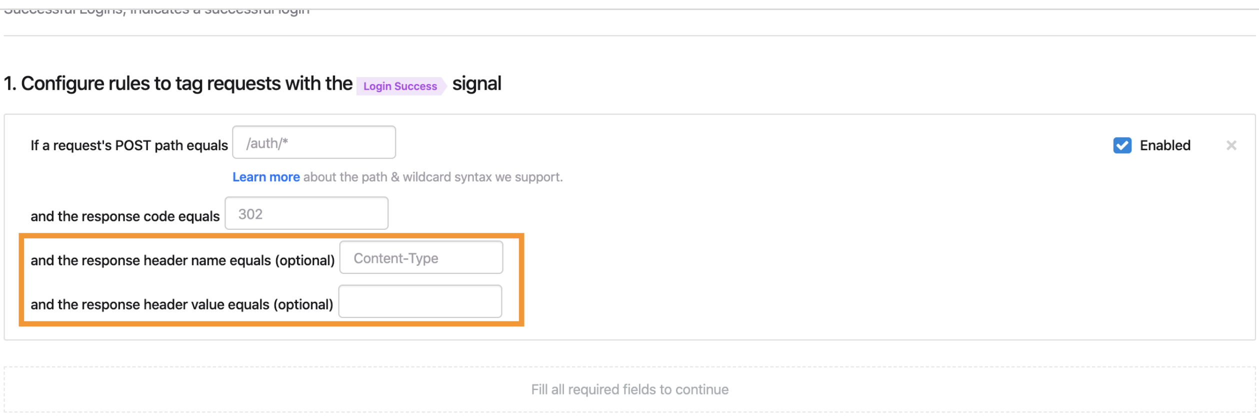
Templated rules response header and value conditions

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:



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/*

☑ Enabled

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)

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 support@signalsciences.com. If you're wondering if this affects you, don't worry! We've been reaching out to anyone this impacts to help them upgrade and we'll make sure that no one is left behind.

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 timeseries 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 latest 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:

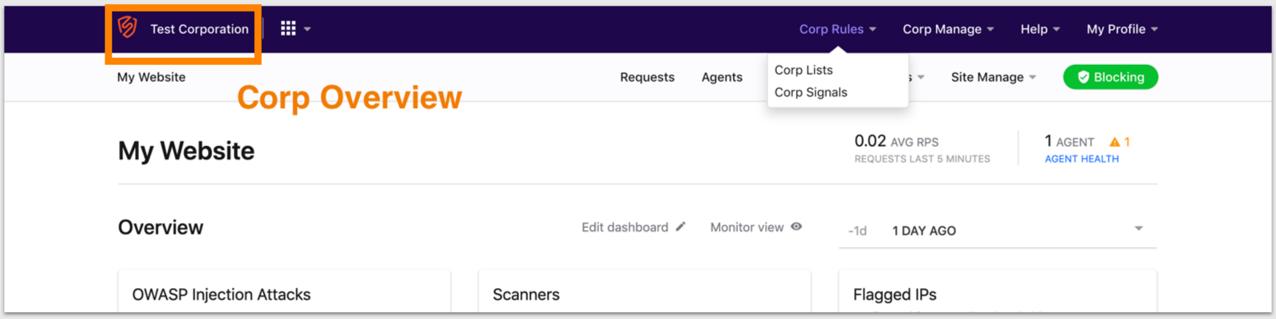
Renamed and reorganized categories:

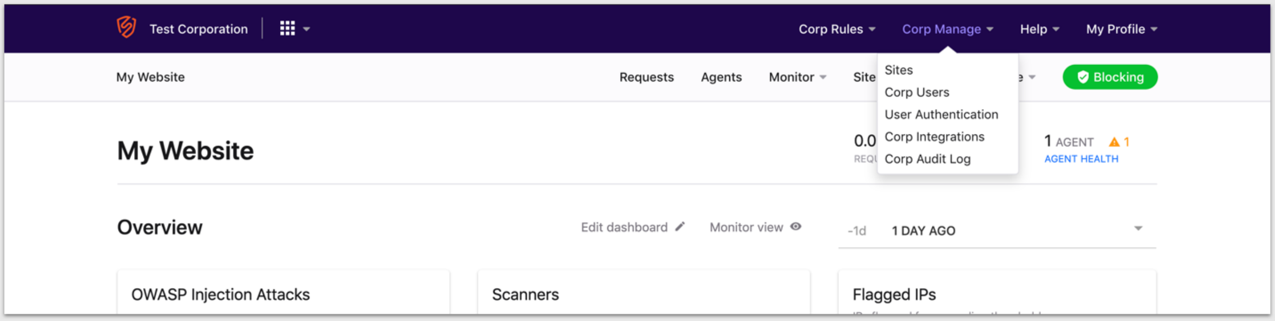
- Library is now "Corp Rules"
- Corp Tools is now "Corp Manage"

- 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

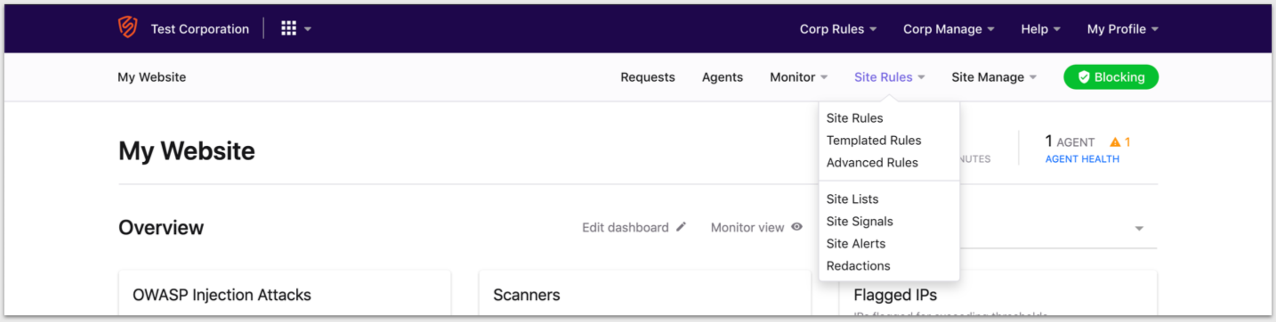
Page nomenclature changes include:

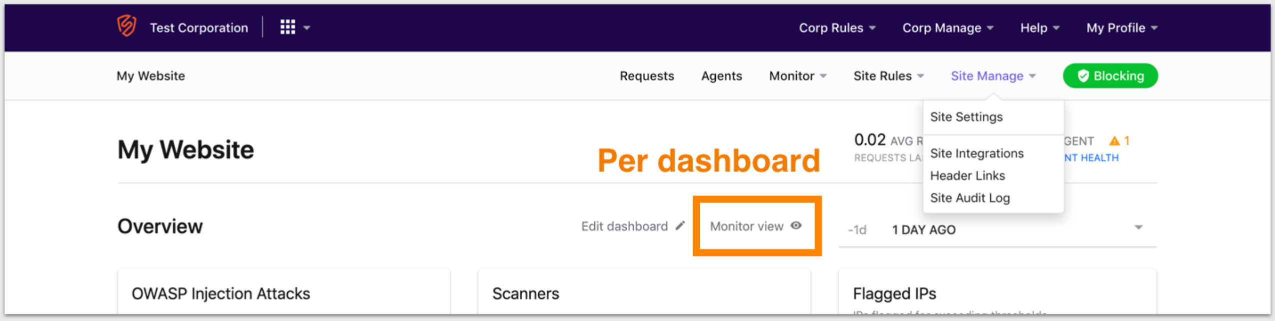
- “Activity” is now “Audit Log”
- “Settings” is now “User Authentication”
- “Week in Review” is now “Weekly Summary”
- “Data Privacy” is now “Redactions”
- “Dashboards” is now “Signals Dashboards”
- “Custom Alerts” is now “Site Alerts”





Site-level changes





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 UI 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 timerange 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 repo 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

cards from the Overview page. When displayed as a carousel, the Monitor will cycle through all cards.

Check out the new Custom Signals page!

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://status.signalsciences.net/ so that you can receive alerts in the rare occasion that Sigsci 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?

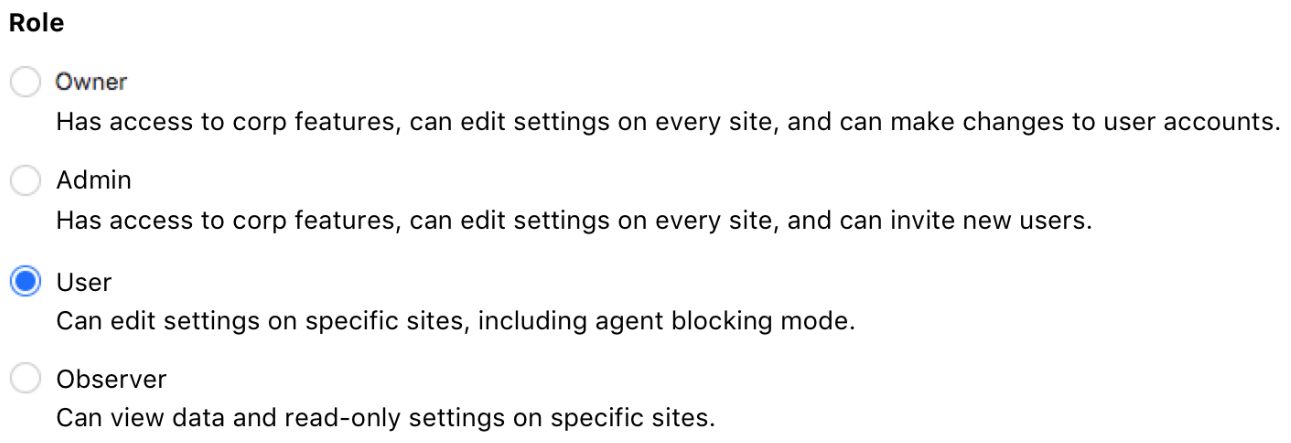
Full documentation for rules is available here.

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.

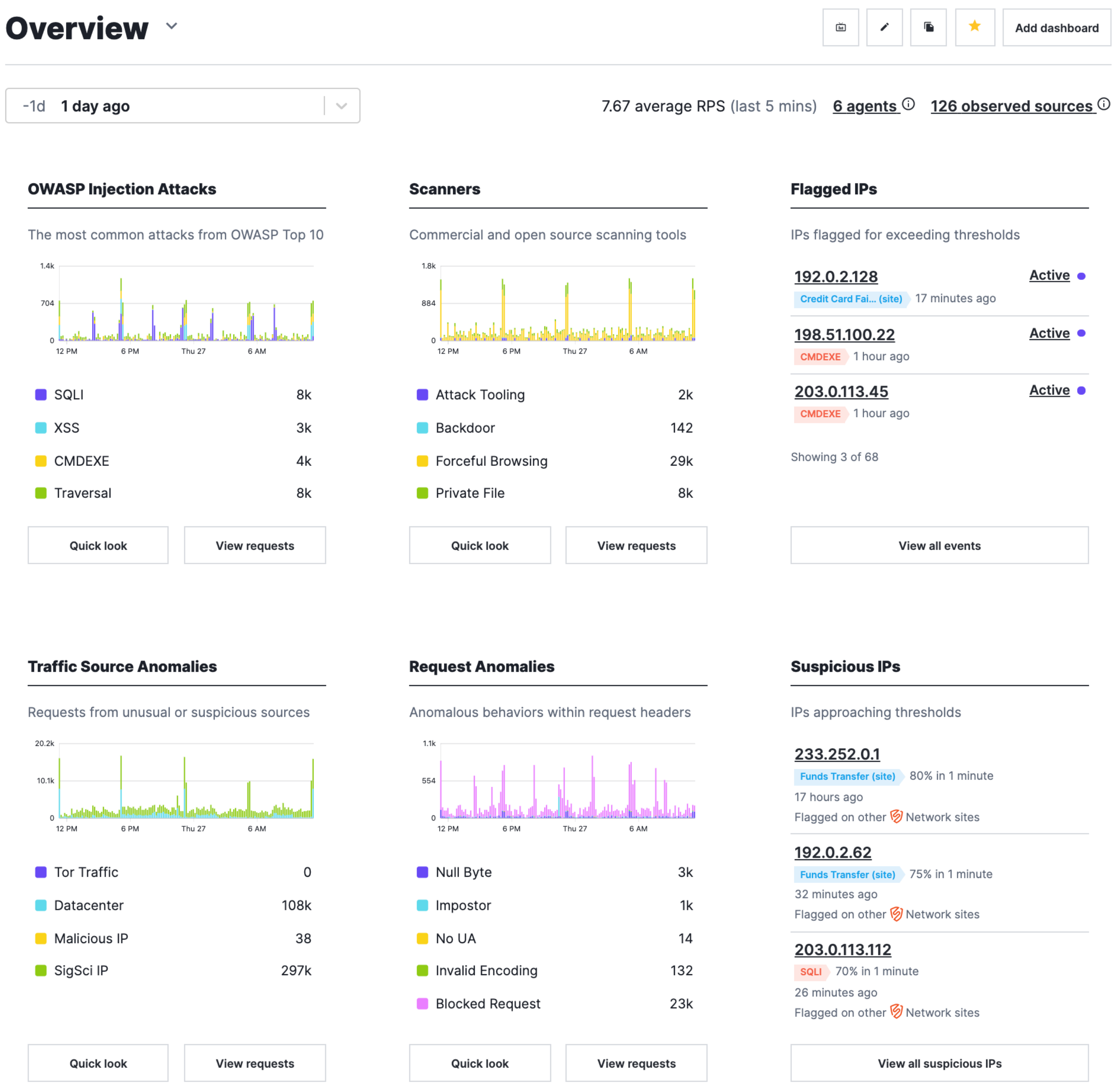
What is a decision?

is sent to the Signal Sciences backend (see the Privacy FAQ to learn how this is done in a safe and private manner). The backend aggregates attacks from across all of your agents, and when enough attacks are seen from a single IP, the backend reaches a decision to flag that IP. Agents will pull those 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 blocking.

The Overview Page

The 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

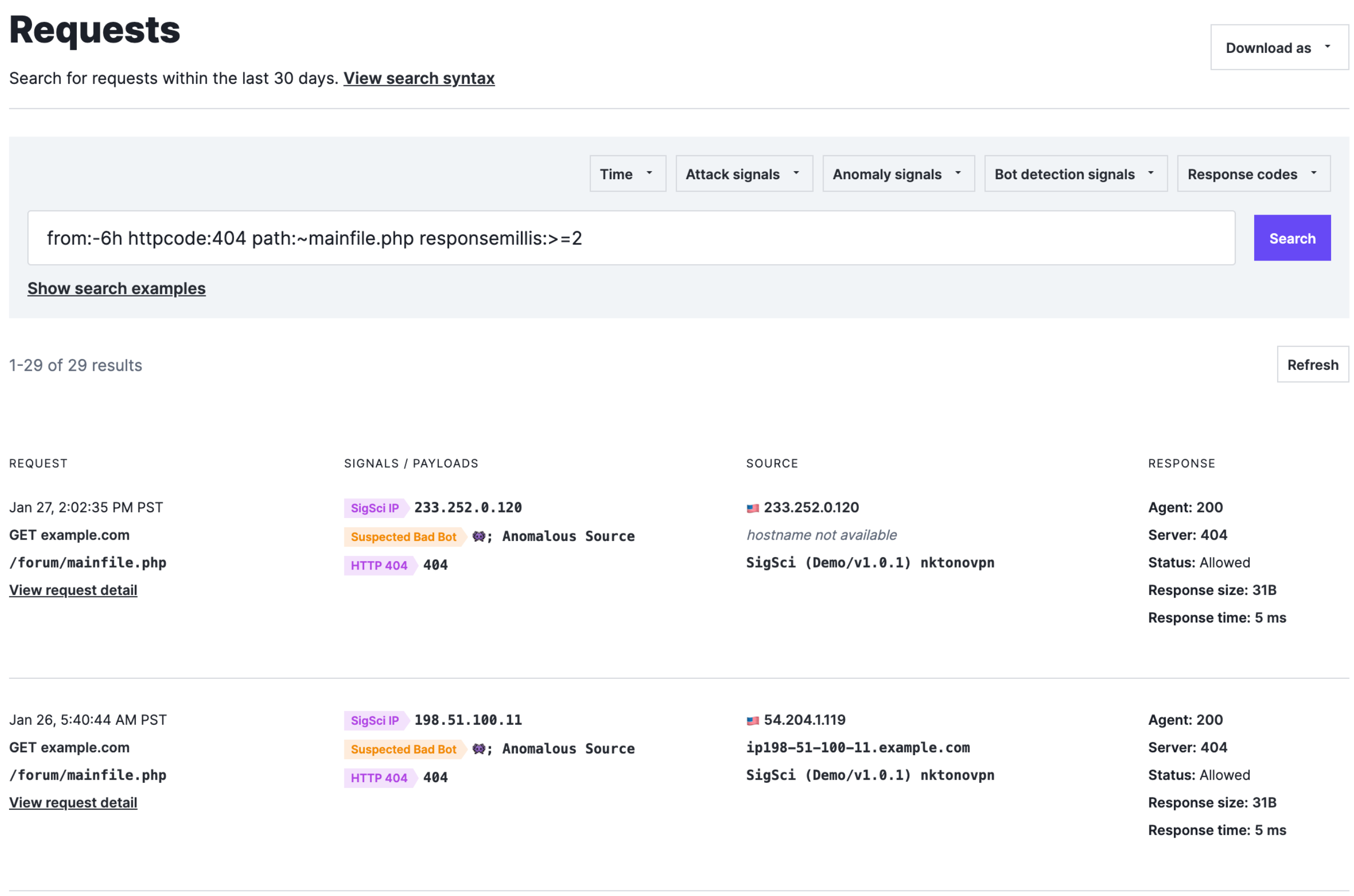
192.0.2.62
Funds Transfer (site) 75% in 1 minute
32 minutes ago
Flagged on other Network sites

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

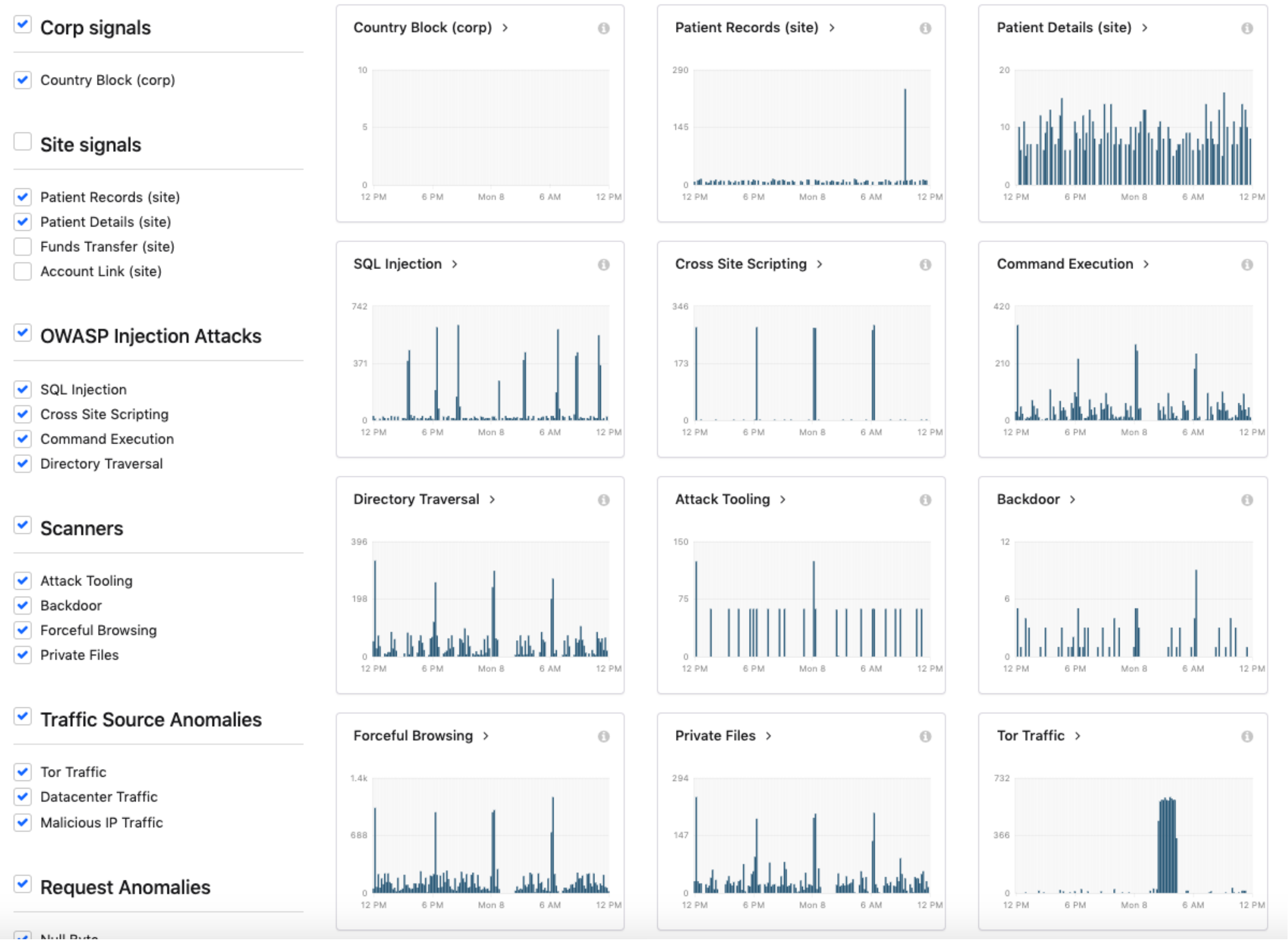
1-29 of 29 results

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`

NOTE: If you are using the EPEL repository with CentOS 7 or 8, you will want to install the **nginx-module-sigsci-epel_nxo.x86_64** module.

NOTE: If you are using the backports repository with Debian 9, you will want to install the **nginx-module-sigsci-bp-nxo** module.

Step 1 of 2

Choose your version

NGINX.org

1.14.1 or higher

1.10.0 - 1.14.1

1.9 or lower

NGINX Plus

> **Note:** The NGINX modules provided by Signal Sciences are built for specific distributions of NGINX (typically those provided by NGINX.org) and may not be compatible with a custom build of NGINX. If switching to an NGINX.org distribution is not an option, open a support ticket and/or contact your Signal Sciences account team for assistance.

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

Apache Module Overview

Compatibility

Our Apache module is distributed in binary form as an Apache shared module and supports Apache version 2.2 and 2.4

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

> **Note:** This information is part of a Limited Availability release. For more information, see our product and feature lifecycle descriptions.

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.

1. Log into the Signal Sciences console.
2. 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

Agent-Only Installation

The Signal Sciences agent can work with an optional module to increase deployment flexibility. If you don’t want to install a module, the 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 Isio 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

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.

- 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)

- Java Module Install
- Node.js Module Install
- .Net Module Install
- .Net Core Module Install
- Python Module Install
- PHP Module Install
- Golang Module Install
- IBM HTTP Server

No Module Option

- Cloud WAF
- Reverse Proxy Mode

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

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

General Troubleshooting

Is someone available to help me with console and/or agent/module issues?

Our whole team is at your disposal to help with any issues you have. Call, text, or email us with issues. And if all else fails, contact us.

Basics

What platforms does SigSci support for the module/agent?

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

Does SigSci provide an API?

Yes, and there is no difference between the customer API and the API Signal Sciences uses to power your dashboards. Full documentation for our REST API can be found here.

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 limits of Signal Sciences features?

Feature	Limit
Alerts	50
Lists	25 per corp + 25 per site
Items in a List	5000
Signals	100 per corp + 100 per site
Request Rules	1000 per corp + 1000 per site
Signal Exclusions	1000 per corp + 1000 per site
Rate Limit Rules	10 per site
Redactions	100

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

What does it mean for a feature to be listed as “experimental”?

Features listed as “experimental” are not fully developed and are subject to change. Use caution when building automated processes involving these features as their functionality may change as they progress.

Account

How do I add more users?

See User Management.

How do I add a new site?

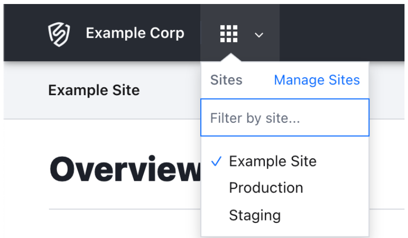
See Site Management.

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 navigate between sites?

To switch between sites, click on the site selector on the left side of the top navigation bar and select from the list of sites enabled on your account. This functionality will appear only if you have more than one site set up for your organization and if you have permissions to view multiple sites.



1. Log into the Signal Sciences console.
2. Select a site if you have more than one site.
3. Click **Agents** in the navigation bar near the top of the screen.

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.

Walkthrough

After successfully installing Signal Sciences, learn how to test and take full advantage of our product:

1. Testing with attack tooling
2. Investigating an attack
3. Testing blocking mode
4. Making security visible

Features

- Rules
- Rate Limit Rules
- Templated Rules
- Lists
- Custom Signals
- Site Alerts
- Events
- Observed Sources
- Custom Response Codes
- Corp Management
- Overview Page
- Corp Overview Report
- Using Single Sign-On
- IDP Provisioning
- Linking Fastly Accounts
- Audit Logs
- Verifying data privacy
- Verifying performance and reliability
- Header Links

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

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:

- Datadog
- Generic Webhooks
- JIRA
- Mailing List
- Microsoft Teams

- Slack
- VictorOps
- Cisco Threat Response / SecureX
- Sumo Logic

Release Notes Introduction

- Agent
- NGINX
- NGINX C Binary
- Apache
- IIS
- Dotnet
- Dotnet Core
- Java
- Heroku
- IBM Cloud
- Cloud Foundry
- Golang
- PHP
- Node.js
- HAProxy
- Python
- 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. To do this simply add `export HTTPS_PROXY=ip.or.host.name` to `/etc/default/sigsci-agent`. Restart the agent and verify the configuration.

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

Changing Hostname for Web Servers

The agent asks the OS for The hostname configuration by default. The agent can be configured to use a custom hostname in one of two ways:

1. Via the command line: `-server-hostname="": server hostname`
2. In the config file with `"server-hostname = value"`

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`
     - `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 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. Module installation instructions can be found here.
3. Restart the web server after module installation.
4. If the module is still not reporting and no data is showing in the console, check for issues related to domain socket permissions. By 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 SE Linux is enabled refer to the SELinux support guide
   - If using Ubuntu check for AppArmor and adjust security profiles if necessary:
   - `sudo apparmor_status`

- 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.

What is a “499” status code?

You may occasionally see the Signal Sciences agent return a status code of “499”. A “499” status code indicates the client closed the connection mid-request.

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 healthchecks 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_MAX_RECORDS
SIGSCI_SAMPLE_PERCENT
SIGSCI_UPLOAD_URL
SIGSCI_UPLOAD_INTERVAL
SIGSCI_UPLOAD_SEND_EMPTY
SIGSCI_DOWNLOAD_URL
SIGSCI_DOWNLOAD_INTERVAL
SIGSCI_SERVER_HOSTNAME
SIGSCI_CLIENT_IP_HEADER
SIGSCI_REVERSE_PROXY
```

```
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+, etc)

Supported Application Types

The Signal Sciences servlet filter module can be easily deployed to a variety of Servlet 3.0+ Java application servers (i.e. Apache Tomcat, Jetty, Glassfish, Resin, etc). 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 (which is `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"
````
~~~

Installation

1. Download or access the Java module:

   Download manually

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

   Access with Maven

   For Java projects using Maven for build or deployment, the Signal Sciences Java modules can be installed by adding the following to the

2. Update your application’s `web.xml` with filter and filter-mapping entry.

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>*

```
<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/sigsci.sock</param-value>
        </init-param>
        <init-param>
            <param-name>expectedContentTypes</param-name>
            <param-value>application/x-java-serialized-object</param-value>
        </init-param>
        <init-param>
        <param-name>excludeIpRange</param-name>
            <param-value>192.168.0.1-192.168.0.5,192.169.0.10-192.169.0.12,193.168.0.1,192.168.10.1-192.168.1
        </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>
```

3. Restart the Application Server.

   Note: If you want coverage across all web applications in your Application Server instance, the jar files in step one should be placed in the server classpath (in Tomcat that would be `%CATALINA_HOME%/lib`). The filter and filter-mapping entries detailed in step 2 should be applied to default deployment descriptor for the container (in Tomcat that would be `%CATALINA_HOME%/conf/web.xml`). Additional Agent configuration options are detailed on the agent configuration page.

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.

	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 IPs to be excluded from filter processing.
excludeIpRange optional, no default	A comma-delimited list of IP ranges or specific IPs 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 are 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>
    <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.

Caveats

- 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.
- IIS Module v2.0 and higher includes a utility—`sigscictl.exe`—that will output diagnostic info. 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 in MSI (Recommended) or a legacy ZIP archive.

https://dl.signalsciences.net/?prefix=sigsci-module-iis/

Alternatively, the IIS module is also downloadable via Nuget 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 (Recommended)

Install IIS Module via MSI

Double-click (or right-click -> 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.

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

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

At this point the installation is complete.

For advanced configuration, refer to the Configuration section.

To confirm that 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
```

Output should look similar to the following:

```
IIS Global Modules:

                            Name Image                                                                    Pre
-------------------------------- ------------------------------------------------------------------------ ---
               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          bit
```

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
```

Output should look similar to the following (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):
```

```
                           agentPort 737
                      statusPagePath
                               Debug False
                    ReuseConnections False
                         MaxPostSize 100000
                         AnomalySize 524288
               AnomalyDurationMillis 1000
                       TimeoutMillis 200
```

Full diagnostics information can be displayed with the following command:

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

Upgrade from previous ZIP install using the MSI

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.

Legacy install and configuration 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.

Install IIS Module via ZIP Archive

1. Extract the ZIP archive contents to the IIS Module install directory (`C:\Program Files\Signal Sciences\IIS Module`)
2. Open a terminal running as Administrator
3. Run the following in the Administrator terminal:

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

This will configure IIS to load the Signal Sciences module and register the configuration schema with IIS.

> **Note:** 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, however, that 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\`.

Uninstall IIS Module via ZIP Archive

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

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

Configuration

Typically, configuration changes are 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. Configuring via MSI or `SigsciCtl.exe` utility is recommended.

To set a configuration option when installing the MSI, just specify the option on the commandline in `option=value` format.via as follows:

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

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

```
"%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 such as follows:

```
    C:\WINDOWS\system32\inetsrv\config\schema: Date Size Name -------------------- ------------ -----------------
```

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:

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

To reset the configuration to defaults using PowerShell, run:

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

To set a configuration option via the `appcmd.exe` command line tool use the `-section:SignalSciences` option such as follows:

```
"%SYSTEMROOT%\system32\inetsrv\appcmd.exe" set config -section:SignalSciences -agentPort:737
```

To list the configuration using `appcmd.exe`, run (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:

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

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

Upgrading

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.

Cloud WAF Certificate Management

Uploading a Certificate for use within Cloud WAF

In this section we'll provide more information and details that are needed to upload an SSL/TLS certificate through the console for use within Cloud WAF. As of today, we only support certificates that are provided to us. Most commonly issued certificates are supported, including self-signed certificates.

Prerequisites

Before uploading your SSL/TLS certificate, ensure that your private key is not password protected, and certificate information is PEM formatted.

At this time, no more than 26 certificates can be uploaded and each certificate must contain no more than 100 hostnames.

Manage certificates

1. Log into the Signal Sciences console.
2. From the **Corp Manage** menu, select **Cloud WAF Certificates**. The Cloud WAF certificate management page appears.
3. The Cloud WAF certificate management page allows you to:
   - Upload certificates.
   - Manage existing certificates.

About the certificate

To proceed with uploading a certificate, we'll need information about the certificate and details from the certificate itself.

- **Name:** This names the cert within our system and makes managing certificates easier. Ensure that your name is more than 4 characters.

- **Domain(s):** This is the FQDN that you intend to protect with Cloud WAF. Note that the domain you input here must match what's in the certificate. If uploading a multi-domain SAN certificate, it is only necessary to include the domains that you intend to protect with Cloud WAF. Our default behavior is to grab all the hostnames in the certificate if no FQDNs are specified in this field.

Certificate details

Once the name and domain(s) have been input and the region selected, provide the certificate information.

> **Note:** Key/certificate information must be uploaded unencrypted and in PEM formatting.

- Private key
- Certificate body
- Certificate chain
  - Also known as the intermediate certificate. The certificate chain is not required for self-signed certificates.

What happens after my certificate has been uploaded?

Once your certificate has been successfully uploaded, your account rep will reach out to you once provisioning has been completed and will provide you with next steps.

Deleting a Certificate

Once a certificate has been uploaded, it can be deleted from the view certificates section. Certificates cannot be deleted if we are in the process of provisioning your cloud WAF.

1. Click **View** to the far right of the certificate. The view certificate page appears.
2. Click **Delete certificate** in the upper-right corner of the screen.

Limits

- Certificates must be PEM encoded and private key must not be password protected.
- Domains: At this time we can support no more than 100 domains in a single deployment.
- Certificates: At this time we cannot support more than 26 certificates per deployment.

Signal Sciences Agent Container Image

Docker Hub

The official `signalsciences/sigsci-agent` container image is now available from the Signal Sciences account on Docker Hub. This image can be pulled via `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

It is possible to build on top of the existing `sigsci-agent` container image using `FROM`, but 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

```
# 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 now included in the tarball sigsci-agent distribution. To build the image, you should download and unpack this archive and follow the instructions in the README.md from this archive.

Example:

```
make docker
```

This will build the recommended `sigsci-agent` container image by unpacking the tarball and running `make docker` from the unpacked directory to build the image tagged with `signalsciences/sigsci-agent:latest` and `signalsciences/sigsci-agent:<version>`. It is possible to use a custom name with 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`:

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

Ubuntu NGINX 1.14.1+

Add the Package Repositories

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Ubuntu 20.04 "focal"

Cut-and-paste the following script into a terminal:

```
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"

Cut-and-paste the following script into a terminal:

```
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"

Cut-and-paste the following script into a terminal:

```
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"

Cut-and-paste 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"

Cut-and-paste 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 the module with apt

Then install the 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*
```

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Restart the Nginx web service

```
sudo service nginx restart
```

Rules

The rules feature allows you to block, allow, and tag requests and exclude system signals for arbitrary sets of conditions. Rules can be created on individual sites (site rules) as well as the corp as a whole (corp rules) to be easily used in multiple sites.

Corp rules can be managed by going to **Corp Rules** > **Corp Rules**.

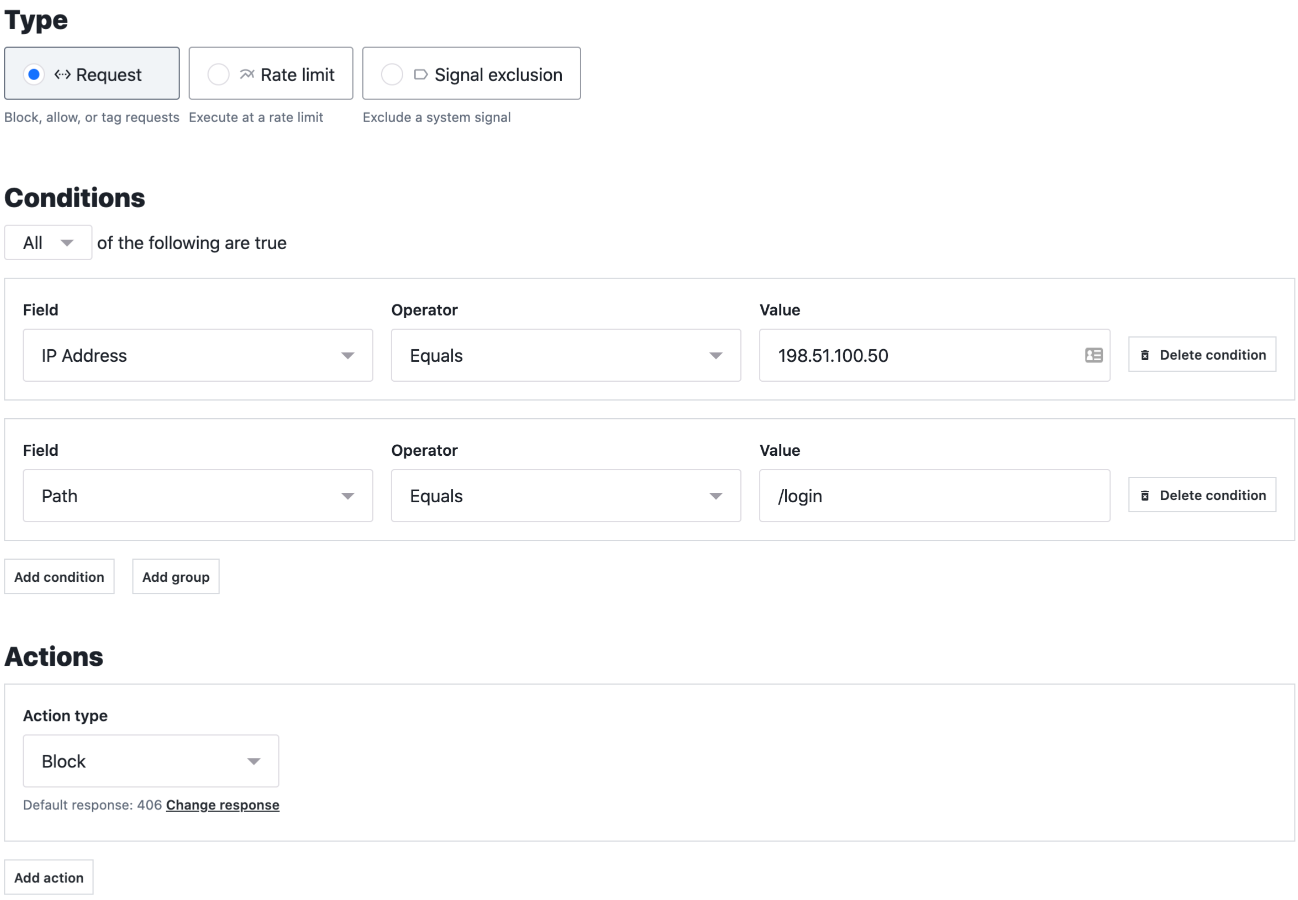
Site rules can be managed by going to **Rules** > **Site Rules**.

Request rules

Request rules allow you to define arbitrary conditions and either block, allow, or tag requests indefinitely or for a specific period of time.

The below example request rule blocks all requests to the `/login` page from the IP address `198.51.100.50`.

- The first condition has **IP Address** selected for the “Field”, **Equals** selected for the “Operator”, and `198.51.100.50` entered for the “Value”.
- The second condition has **Path** selected for the “Field”, **Equals** selected for the “Operator”, and `/login` entered for the “Value”.
- The “Action type” is set to **Block**.



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

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

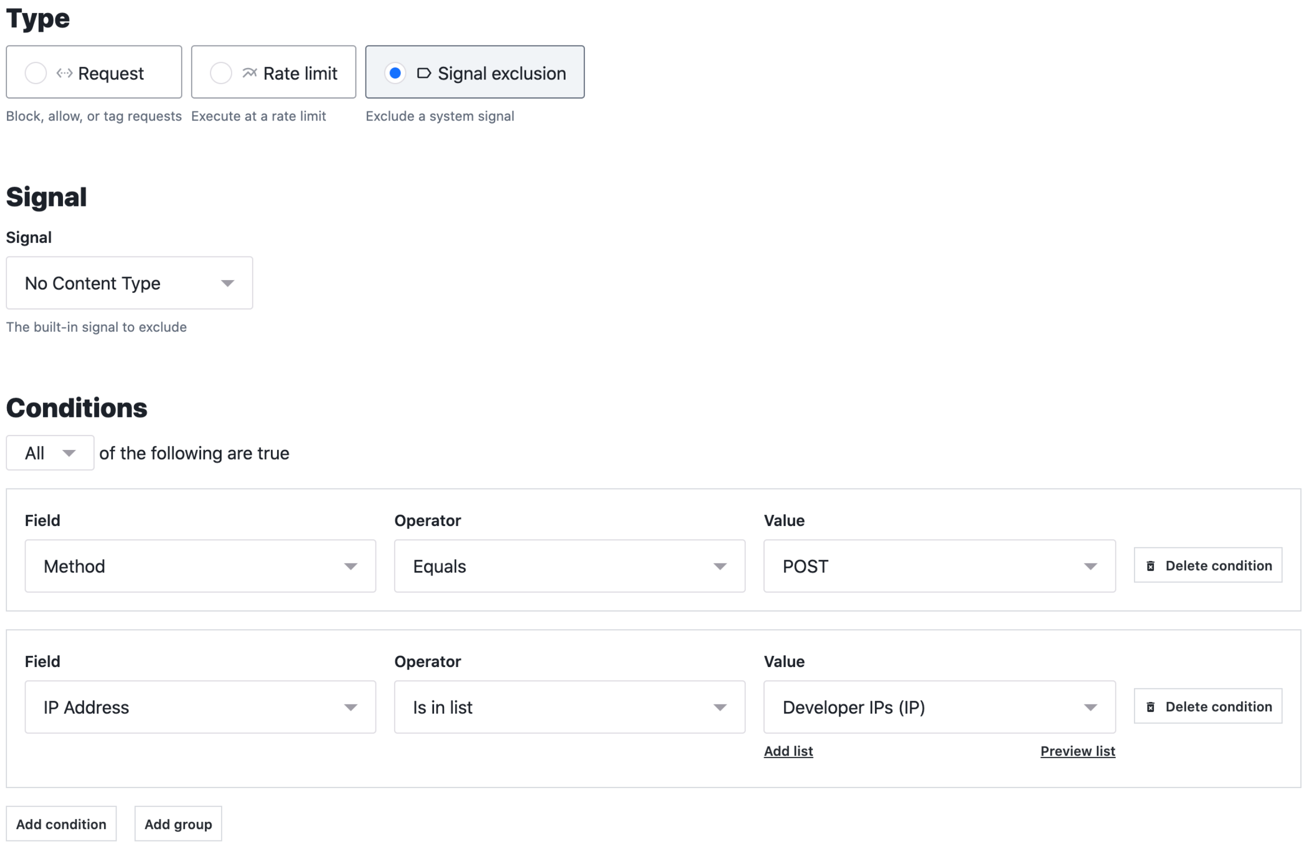
Path	String	Text or wildcard
POST parameter	Multiple	`Name (string), Value (string)`
Query parameter	Multiple	`Name (string), Value (string)`
Request cookie	Multiple	`Name (string), Value (string)`
Request header	Multiple	`Name (string), Value (string), Value (IP)`
Response code	String	Text or wildcard
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

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**.
- The first condition has **Method** selected for the "Field", **Equals** selected for the "Operator", and **POST** selected for the "Value".
- The second condition has **IP address** selected for the "Field", **Is in list** selected for the "Operator", and the **Developer IPs (IP)** list 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	Delete condition
IP Address	Is in list	Developer IPs (IP) Add list Preview list	Delete condition

Signal exclusion fields

Signal exclusion rules have the same fields as request rules as well as additional fields specific to the particular signal that's being excluded.

Field	Type	Properties
Parameter name	String	Text or wildcard
Parameter value	String	Text or wildcard

Rate limit rules

See Templated Rules for information about using templated rules.

Converting Requests to Rules

Individual requests in the Requests page can be converted into pre-populated rules, enabling you to easily allow, block, and tag similar requests.

How to convert a request to a rule

1. In the console, go to **Requests**.
2. Locate or search for the request you want to convert into a rule.
3. Click **View request detail**.
4. Click **Convert to rule** in the upper-right corner.
5. Select the type of rule you want to make (e.g., request, rate limit, or signal exclusion).
6. 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 IP address and path featured in the request.
7. Click **Continue**.
8. You will be taken to a pre-built rule with conditions featuring the request characteristics you selected. Modify the rule as needed, such as by adding and editing rule conditions.
9. Finish setting up the rule by setting:
    - What action it should take (e.g., block, allow, or tag requests).
    - Whether it should be enabled or disabled.
    - If it should automatically be disabled after a set period of time.
    - A description of the rule.
10. Click **Create site rule**.

Operators

When creating rules, operators (“equals”, “is in list”, etc.) 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—for example, to match a specific IP address or path.

Operator	Function	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	`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	`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	`203.0.113.169` Is in list “Known IP Addresses”

“Known IP Addresses”

Wildcards

The “Like (wildcard)” operator supports 0-or-many wildcards (`*`), single-character wildcards (`?`), character-lists (`[abc]`), character-ranges (`[a-c]`, `[0-9]`), alternatives (`{cat,bat,[fr]at}`), and exclusions (`[!abc]`, `[!0-9]`).

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.

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.

Path syntax best practices

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/`:

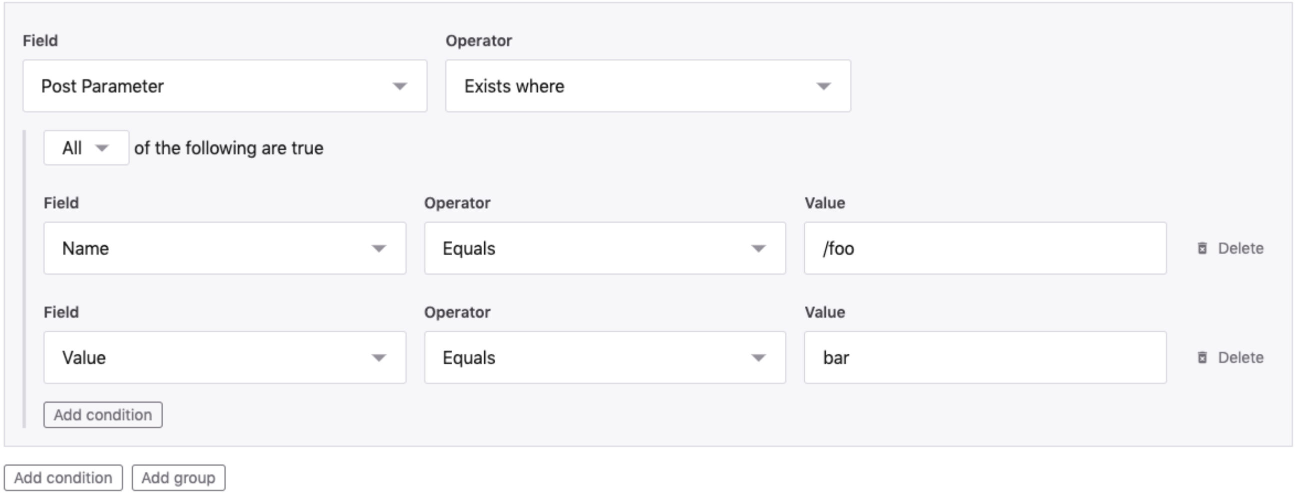
- `/login/` will return a match
- `/login` with not return a match

JSON POST body

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"]
  }
}
```

Geolocation

Geolocation 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 as of agent version 3.21.0.

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, such as 3.0.0 to 3.1.0. As of agent version 3.21.0, 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 or is not version 3.21.0 or later which will dynamically update, 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?

As of agent 3.21.0 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)

How do I update my agent?

See Upgrading the Agent for documentation on how to upgrade the agent.

Console

My Data Is not Showing In The Console With Working Module/Agent.

If both the agent and module are reporting as active within the console but no data is displayed when requests are processed there is a good chance that the system time on the agent is out of sync and thus events are being reported at times significantly in the past or future. This can happen especially 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, open the Agents page in the console and navigate to the graphs section. There is a graph for agent clock skew and this 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.

Checklist When No Signals Appear On The Console

1. Confirm the versions of your OS and web server are supported versions.
2. Confirm your agent is online via the console.
3. Confirm your module is detected via the console.
4. Confirm the console is receiving telemetry from the agent.
5. Confirm your systems Clock Skew minimal (see agent clock skew graph). Run NTP update on your system if necessary.
6. If Nginx is your web server run `/opt/sigsci/bin/check-nginx`
7. Collect logs for review by support:
   - Run the agent with the command line argument `-debug-log-all-the-things`
   - If Nginx is your web server collect your Nginx error.log
8. Collect configuration files `/etc/sigsci/agent.conf` and if running NGINX `/etc/nginx/nginx.conf`

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

Some customers have noticed that foreign domains are showing up in their console views. This can happen if the requestor is using a modified hosts file or forged host header so that it appears the target is a foreign host but has 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, 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

Step 1 - Add the Package Repositories

We’ll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Ubuntu 20.04 “focal”

Cut-and-paste the following script into a terminal:

```
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/sigsc
```

Ubuntu 18.04 “bionic”

Cut-and-paste the following script into a terminal:

```
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/sigs
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”

Cut-and-paste 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”

Cut-and-paste 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
```

Step 2 - Install the Signal Sciences Agent Package

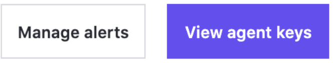
1. To install the package, running the following command.

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

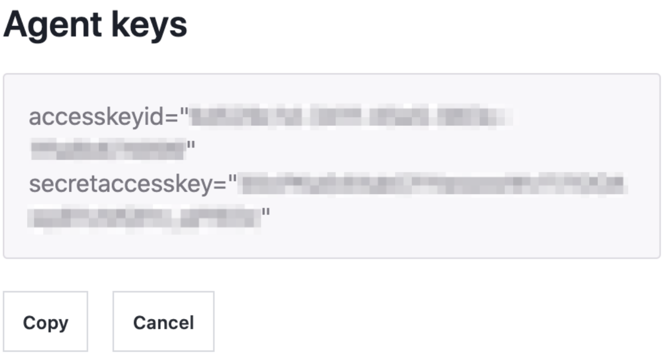
2. Create the file `/etc/sigsci/agent.conf`

3. Configure the agent by inputting the Agent Access Key and Agent Secret Key into the `/etc/sigsci/agent.conf`.

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

   - The Agent Access Key and Agent Secret Key will be visible within the window:

     Agent keys

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

   Example /etc/sigsci/agent.conf

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

   Additional configuration options are listed on the agent configuration page.

4. Start the Signal Sciences Agent

   Ubuntu 14.04 and lower `sudo start sigsci-agent`

   Ubuntu 15.04 through 17.10 `sudo systemctl start sigsci-agent`

   Ubuntu 18.04 and higher `sudo service sigsci-agent start`

- Explore module options

Ubuntu Apache Module Install

1. Install the Apache module using `apt-get`.

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

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the "Dynamic Shared Object (DSO) Support" section:

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

3. Restart the Apache web service.

   ```
   sudo service apache2 restart
   ```

Next Steps

- Verify Agent and Module Installation

Explore other installation options:

- Explore module options

Testing With Attack Tooling

The first thing you should do to test Signal Sciences is to run attack tooling against your site to verify that attack data is being captured and blocking is working correctly.

Running the scan

While you can use any attack tooling for testing, we recommend using Nikto which tests a wide variety of vulnerabilities. While Nikto is running, our agents will identify any malicious or anomalous requests and send relevant metadata to our backend, after redacting any sensitive information.

The next sections cover getting set up with Nikto and running a few scan scenarios.

Nikto Setup

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.

1. Nikto requires Perl to be installed, which can be verified by running `perl -v`. If Perl is not found on your system see http://learn.perl.org/installing/ for installation guides.
2. Download the latest version of Nikto from https://github.com/sullo/nikto/archive/master.zip. For more information about Nikto see https://cirt.net/Nikto2
3. At the command prompt use the command `unzip nikto-master.zip` to unzip the file. Then change directories to the program directory with the command `cd nikto-master/program/`.
4. To verify you are able to run Nikto run `./nikto.pl` and it will display the default help message. If you receive a permission denied error message, this can be resolved by running `chmod +x nikto.pl` to make the script executable, then run `./nikto.pl` again.

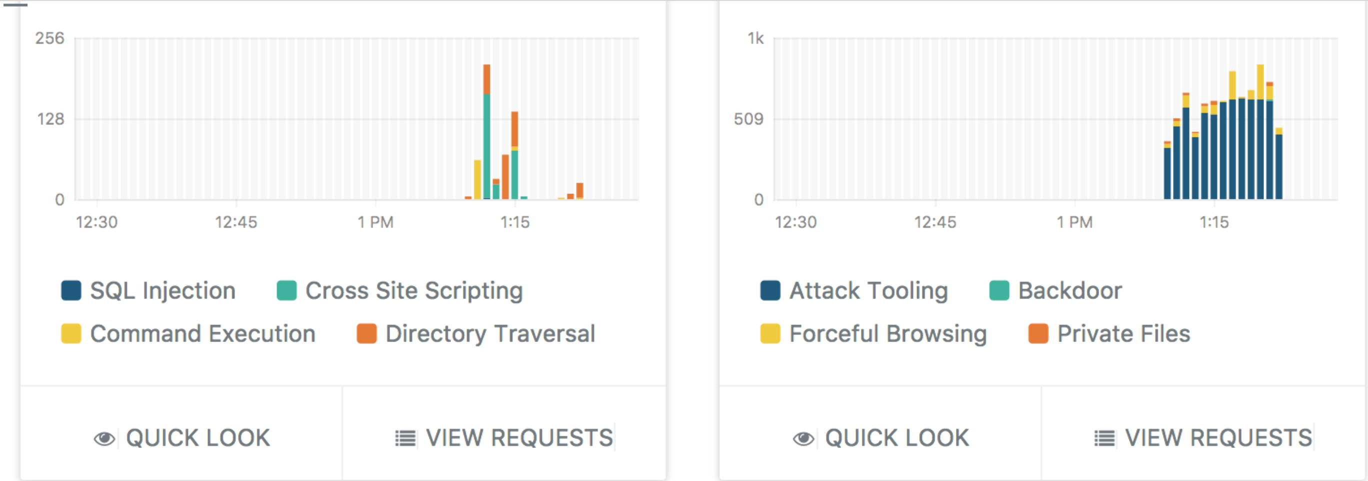
Scenario 1 - Detecting Attacks (Attack Tooling)

As the first test scenario, Nikto will be used to demonstrate Signal Sciences' attack tooling detection capability. For this scenario ensure the agent mode is set to "not blocking". To verify, log in to the Signal Sciences console (https://dashboard.signalsciences.net/) and confirm the top menu label displays "Not blocking".

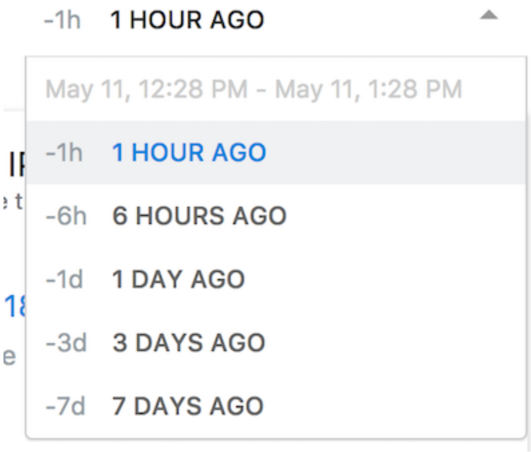
To initiate the first Nikto scan of your site run the following command:

```
./nikto.pl -h http://www.example.com
```

While the scan is running, attacks and anomalies will begin to appear within 30 seconds on the console. Example:



Note: You can modify the time period on the graph to limit or expand the amount of malicious traffic to display. For this scenario click the time menu selector on the top right corner of the Overview page and select 1 HOUR AGO.

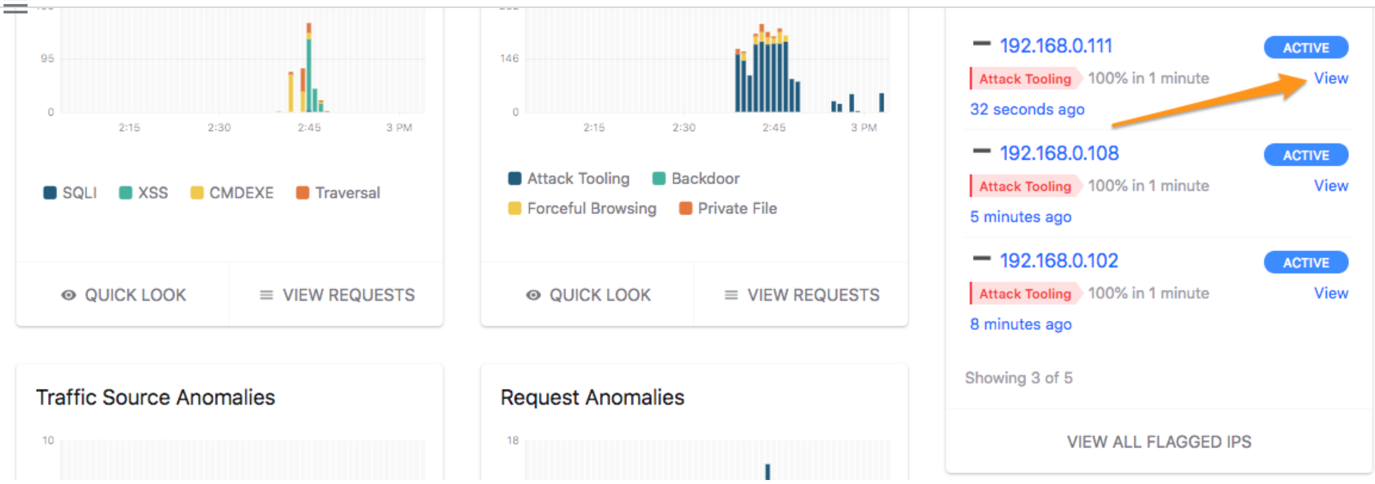


Note: The IP address you are running the scan from may briefly appear on the Suspicious IPs list, but it will ultimately appear on the Events list.

Common Question: What is the difference between the Suspicious IPs list and the Events list?

Answer: The Suspicious IPs list represents IP addresses that are the origin of requests containing attack payloads, but the volume of attack traffic from that IP address has not exceeded the decision threshold. 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).

Once the IP address appears on the Events list click on **View** or the time status to open the Events page for that IP address.



192.168.0.111 ACTIVE
Attack Tooling 100% in 1 minute View
32 seconds ago

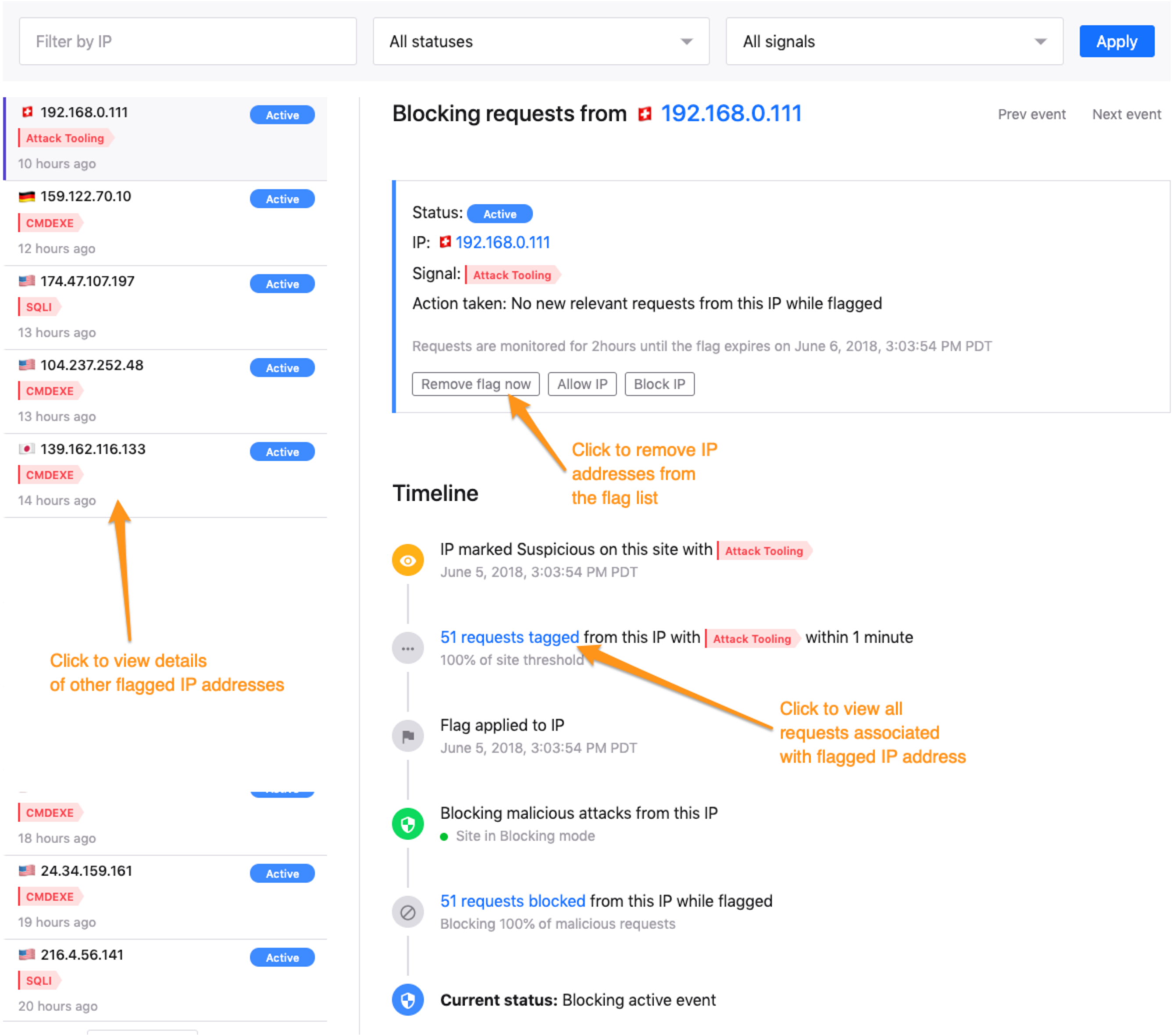
192.168.0.108 ACTIVE
Attack Tooling 100% in 1 minute View
5 minutes ago

192.168.0.102 ACTIVE
Attack Tooling 100% in 1 minute View
8 minutes ago

Showing 3 of 5

The Events page explains that the IP address was flagged because the agent tagged X number of requests with the Attack Tooling signal within a certain time based threshold. In the example screenshot below it states “51 requests tagged from this IP with Attack Tooling within 1 minute”. Additional information about time based thresholds can be found here.

Notice you may browse other events from this page as well. In addition, you can use the buttons on this page to allow the IP, block the IP, or remove the flag.



Filter by IP | All statuses | All signals | Apply

192.168.0.111 Active
Attack Tooling
10 hours ago

159.122.70.10 Active
CMDEXE
12 hours ago

174.47.107.197 Active
SQLI
13 hours ago

104.237.252.48 Active
CMDEXE
13 hours ago

139.162.116.133 Active
CMDEXE
14 hours ago

Click to view details
of other flagged IP addresses

CMDEXE
18 hours ago

24.34.159.161 Active
CMDEXE
19 hours ago

216.4.56.141 Active
SQLI
20 hours ago

Blocking requests from 192.168.0.111

Prev event Next event

Status: Active
IP: 192.168.0.111
Signal: Attack Tooling
Action taken: No new relevant requests from this IP while flagged

Requests are monitored for 2hours until the flag expires on June 6, 2018, 3:03:54 PM PDT

Remove flag now | Allow IP | Block IP

Click to remove IP
addresses from
the flag list

Timeline

IP marked Suspicious on this site with Attack Tooling
June 5, 2018, 3:03:54 PM PDT

51 requests tagged from this IP with Attack Tooling within 1 minute
100% of site threshold

Click to view all
requests associated
with flagged IP address

Flag applied to IP
June 5, 2018, 3:03:54 PM PDT

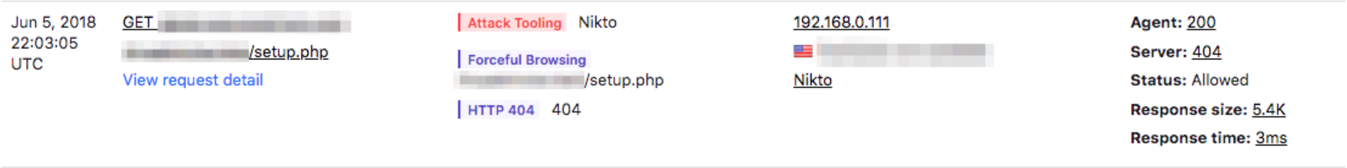
Blocking malicious attacks from this IP
Site in Blocking mode

51 requests blocked from this IP while flagged
Blocking 100% of malicious requests

Current status: Blocking active event

Common Question: Why was this IP address flagged only with the Attack Tooling signal and not other signals like XSS, or SQLi?

Answer: Many attack tools perform numerous requests to fingerprint the server being targeted before launching actual attacks. This is done to select payloads that may be more specific to that server's technology. This initial fingerprinting traffic won't contain malicious payloads but the agent still detects the tool based on certain characteristics of the traffic. A common characteristic is the User-Agent string, which in the case of Nikto will contain "Nikto". As a result, the amount of fingerprinting traffic Nikto generates was enough to cause the IP address to be flagged with the single Attack Tool signal. However, if you view the requests you can see the other signals that were also applied to each request. Referring to the events page screenshot above, you would click the link text **51 requests tagged** to view all related requests and the associated signals.



Jun 5, 2018
22:03:05
UTC

GET [redacted]
[redacted]/setup.php
View request detail

Attack Tooling Nikto
Forceful Browsing
[redacted]/setup.php
HTTP 404 404

192.168.0.111
[redacted]
Nikto

Agent: 200
Server: 404
Status: Allowed
Response size: 5.4K
Response time: 3ms

In this test scenario you learned the following:

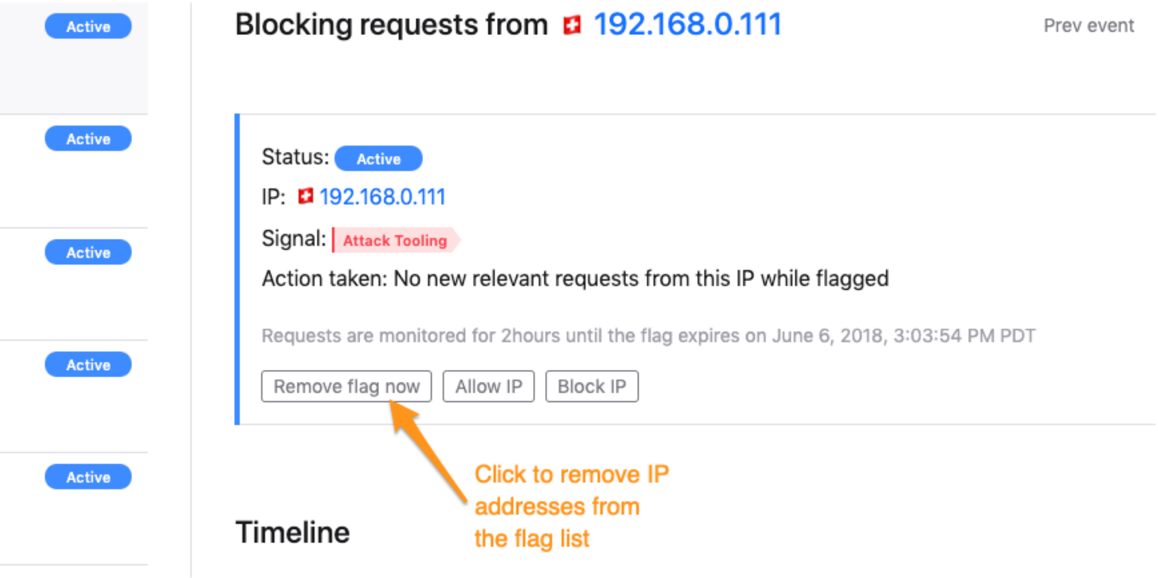
- How to run Nikto to generate attacks and anomalies against your web application.
- How to modify the graph time period for attacks and anomalies.

- How to navigate to details of all requests associated with an event.

Scenario 2 - Detecting Attacks

In this second scenario we'll modify our Nikto scan to demonstrate an IP address being flagged due to injection attacks, rather than just attack tooling. With Nikto this can be done by modifying the User-Agent string that is sent with each request.

Make sure the scanner's host IP address has been removed from the flagged list. To remove an IP address from the flagged list, navigate to the Events page by clicking **View** for the IP address in the Events list. Next, click the **Remove flag now** button and a dialog will prompt you for confirmation. Click the **Remove flag** button in the dialog to confirm removal.



Blocking requests from 192.168.0.111

Status: Active

IP: 192.168.0.111

Signal: Attack Tooling

Action taken: No new relevant requests from this IP while flagged

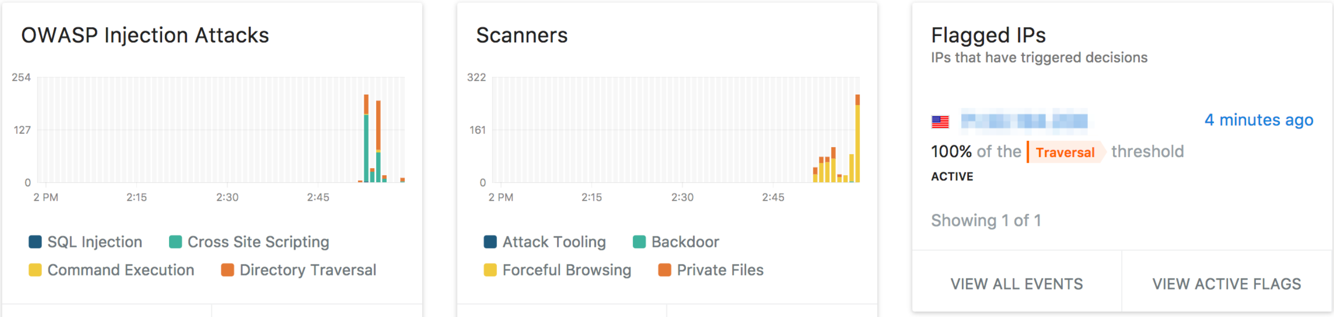
Requests are monitored for 2hours until the flag expires on June 6, 2018, 3:03:54 PM PDT

Click to remove IP addresses from the flag list

To initiate the scan with a modified User-Agent string use the following command:

```
./nikto.pl -useragent "MyAgent (Demo/1.0)" -h http://www.example.com
```

As before, the attacks and anomalies begin to populate on the console. Notice this time however that the majority of signals are due to various attacks and not attack tooling. This means modifying the User-Agent string worked and the IP address will eventually be flagged based on the various attacks.



[redacted] 4 minutes ago

100% of the Traversal threshold

ACTIVE

Showing 1 of 1

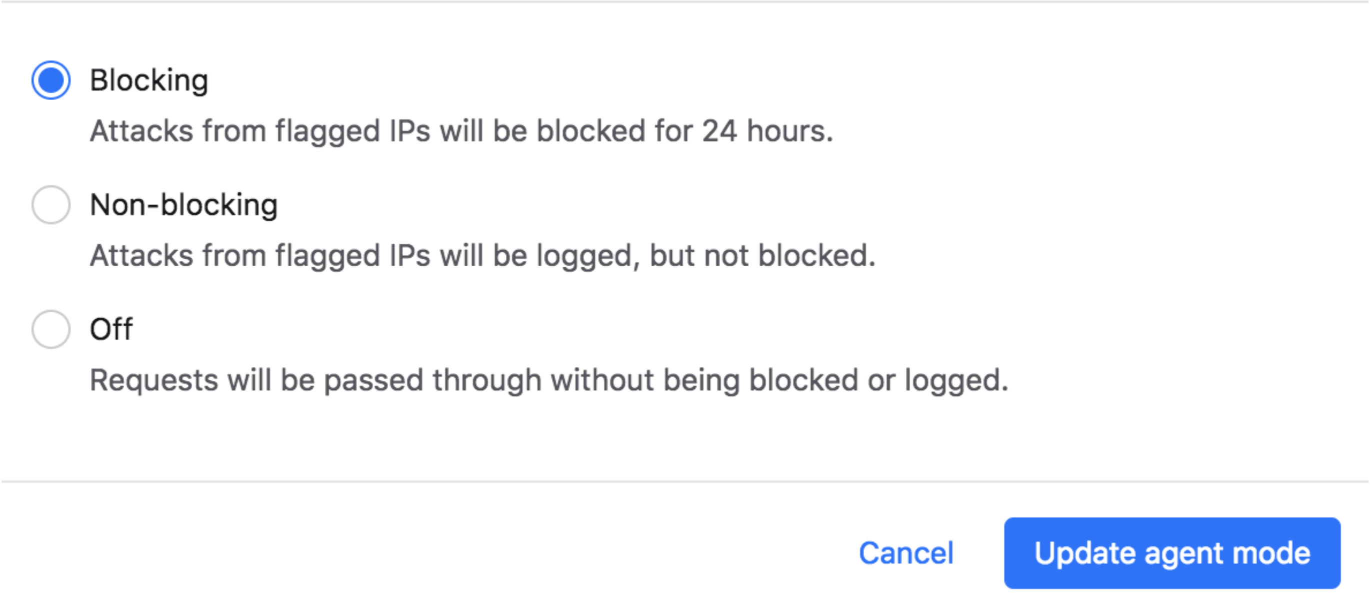
In this test scenario you learned the following:

- How to remove an IP address from the flagged list.
- How to modify Nikto's User-Agent string to avoid immediate detection as an attack tool.

Scenario 3 - Blocking Attacks Without Impacting Legitimate Traffic

For the third scenario you will run another scan, but this time with blocking mode enabled. With blocking mode enabled this scenario will demonstrate how Signal Sciences will allow legitimate traffic to continue accessing the site while malicious traffic from the same IP address is blocked. To perform this test you will need to use a web browser that is on the same system you are running the scan from. Before continuing, make sure to remove the scanning IP address from the flagged list.

Change the agent's mode to blocking. Click the top menu label **Non-blocking** to open the Agent Mode dialog.

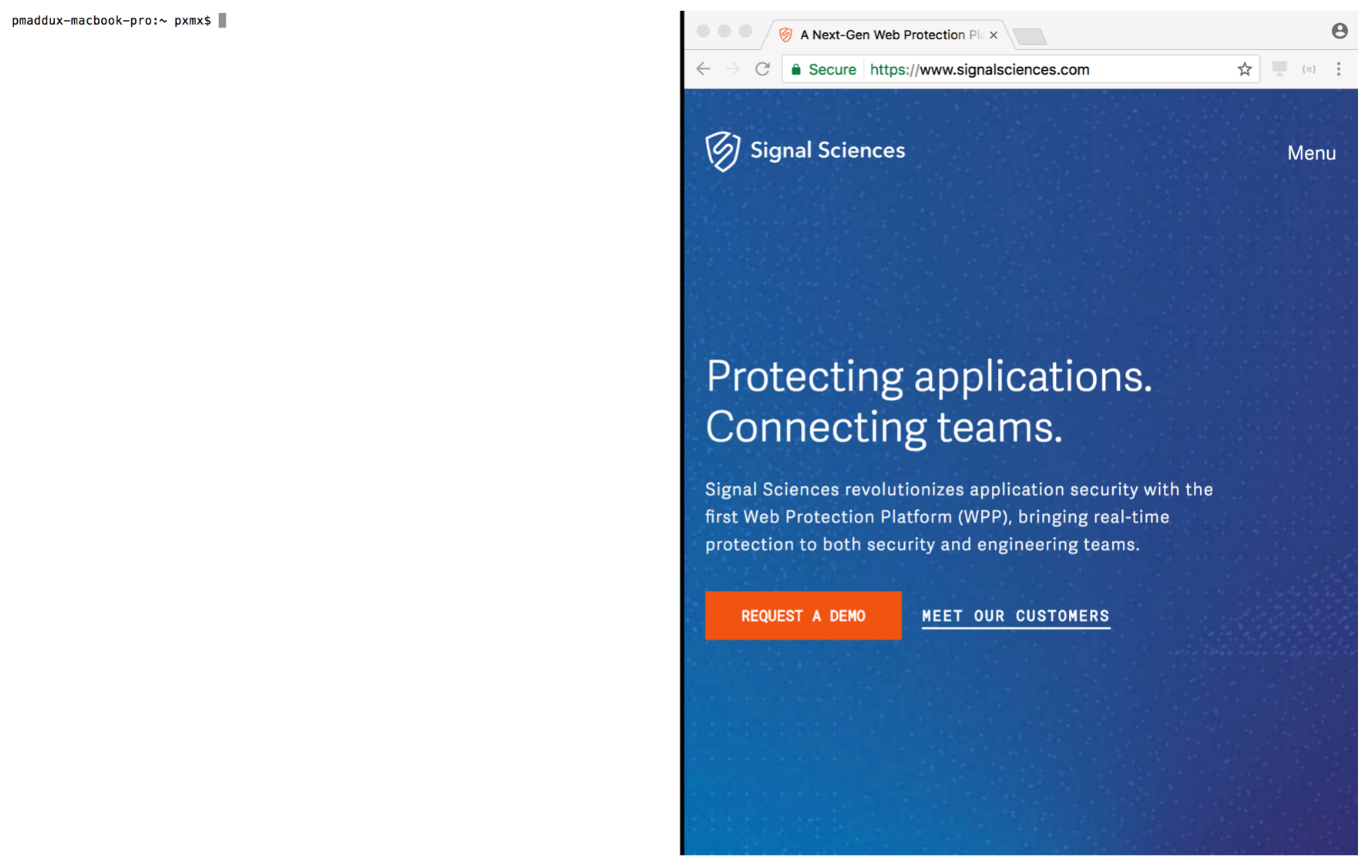


● Blocking
Attacks from flagged IPs will be blocked for 24 hours.

○ Non-blocking
Attacks from flagged IPs will be logged, but not blocked.

○ Off
Requests will be passed through without being blocked or logged.

Next, arrange your windows so the command shell window is side-by-side with a web browser window. This will allow you to view responses from the Nikto scan while navigating your site as a normal user would.



```
pmaddux-macbook-pro:~ pxmx$
```

A Next-Gen Web Protection Pl
Secure https://www.signalsciences.com
Signal Sciences
Menu
REQUEST A DEMO
MEET OUR CUSTOMERS

Protecting applications.
Connecting teams.

Signal Sciences revolutionizes application security with the first Web Protection Platform (WPP), bringing real-time protection to both security and engineering teams.

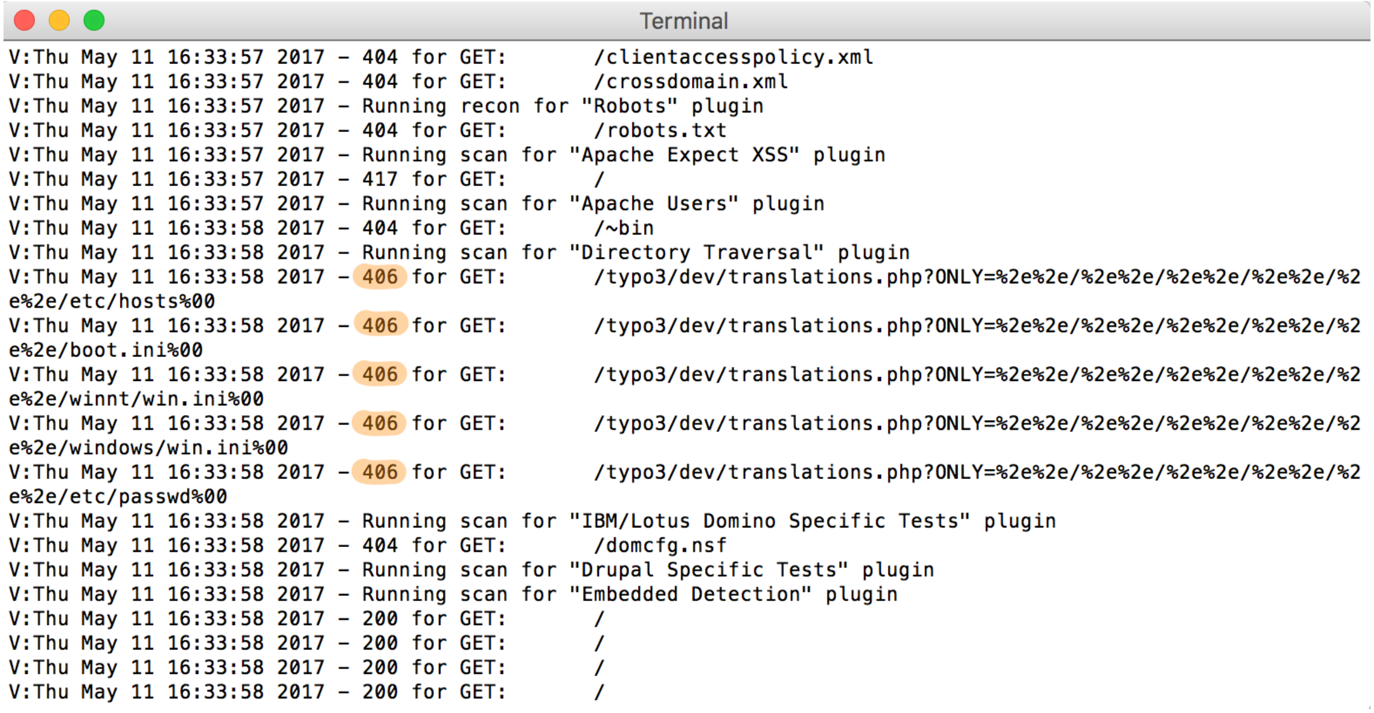
You are now ready to initiate a scan. However, this time add the following additional command line arguments to the Nikto command:

- `-D V` for verbose output, this will let you see when requests are blocked by Signal Sciences with an HTTP 406 response code.
- `-T 9` to tune the scan so it only tests for SQL Injection.

To initiate the with the additional arguments use the following command:

```
./nikto.pl -useragent “MyAgent (Demo/1.0)” -D V -T 9 -h http://www.example.com
```

complete quicker than previous scans. Repeat the scan as many times as desired and continue browsing the site to confirm legitimate user traffic is not blocked.



```
V:Thu May 11 16:33:57 2017 - 404 for GET:       /clientaccesspolicy.xml
V:Thu May 11 16:33:57 2017 - 404 for GET:       /crossdomain.xml
V:Thu May 11 16:33:57 2017 - Running recon for "Robots" plugin
V:Thu May 11 16:33:57 2017 - 404 for GET:       /robots.txt
V:Thu May 11 16:33:57 2017 - Running scan for "Apache Expect XSS" plugin
V:Thu May 11 16:33:57 2017 - 417 for GET:       /
V:Thu May 11 16:33:57 2017 - Running scan for "Apache Users" plugin
V:Thu May 11 16:33:58 2017 - 404 for GET:       /~bin
V:Thu May 11 16:33:58 2017 - Running scan for "Directory Traversal" plugin
V:Thu May 11 16:33:58 2017 - 406 for GET:       /typo3/dev/translations.php?ONLY=%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2
e%2e/etc/hosts%00
V:Thu May 11 16:33:58 2017 - 406 for GET:       /typo3/dev/translations.php?ONLY=%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2
e%2e/boot.ini%00
V:Thu May 11 16:33:58 2017 - 406 for GET:       /typo3/dev/translations.php?ONLY=%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2
e%2e/winnt/win.ini%00
V:Thu May 11 16:33:58 2017 - 406 for GET:       /typo3/dev/translations.php?ONLY=%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2
e%2e/windows/win.ini%00
V:Thu May 11 16:33:58 2017 - 406 for GET:       /typo3/dev/translations.php?ONLY=%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2
e%2e/etc/passwd%00
V:Thu May 11 16:33:58 2017 - Running scan for "IBM/Lotus Domino Specific Tests" plugin
V:Thu May 11 16:33:58 2017 - 404 for GET:       /domcfg.nsf
V:Thu May 11 16:33:58 2017 - Running scan for "Drupal Specific Tests" plugin
V:Thu May 11 16:33:58 2017 - Running scan for "Embedded Detection" plugin
V:Thu May 11 16:33:58 2017 - 200 for GET:       /
V:Thu May 11 16:33:58 2017 - 200 for GET:       /
V:Thu May 11 16:33:58 2017 - 200 for GET:       /
V:Thu May 11 16:33:58 2017 - 200 for GET:       /
```

Common Question: Why is an HTTP 406 response code used for blocking attacks?

Answer: 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, customers can customize the error message that is returned by the server however they would like.

In this scenario you learned the following:

- How to enable blocking mode.
- How to arrange your command shell window and browser window to observe Signal Sciences' blocking capability.
- When in blocking mode Signal Sciences only blocks malicious requests and not legitimate user requests, even when these request are coming from the same IP address.

Conclusion

Through this series of quick test scenarios, you have been able to prove both the detection capabilities of Signal Sciences, as well as the ability to use Signal Sciences in blocking mode to stop attacks without blocking legitimate traffic.

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, Owner Users 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

Creating API Access Tokens

1. Go to **Profile** > **API Access Tokens**

2. Click **Add API access token**

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

5. Click **Continue** to finish creating the token

Restricting User Permission to Create and Use API Access Tokens

Owner Users can restrict all users from creating and using API Access Tokens. After doing so, Owner Users 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.

Owner Users can enable API Access Token restrictions by following these steps:

1. Go to **Corp Manage** > **User Authentication**
2. Scroll down to the section labeled **API Access Tokens**.
3. Under **Access token permissions**, select **Restrict access by user**
4. A message will be displayed warning you about this setting and its restrictions. Click **Continue** to proceed.
5. Click **Update API Access Tokens** at the bottom to save this change

Granting Users Permission to Create and Use API Access Tokens

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

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

1. Go to **Corp Manage** > **Corp Users**
2. Click on the user you want to grant permission to
3. Click **Edit corp user** at the top
4. Under **Authentication** check the box labeled **Allow this user to create API Access Tokens**
5. Click **Update user** at the bottom

Deleting API Access Tokens

1. Go to **Profile** > **API Access Tokens**
2. Click **View** to the far right of the token you want to delete
3. Click **Delete API access token**
4. Click **Delete** to confirm you want to delete the token

Viewing Personal API Tokens

Owner Users can view a table of all access tokens across your corp by going to **Corp Manage** > **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 Corp-Wide API Access Token Settings

Setting Automatic Token Expirations

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

1. Go to **Corp Manage** > **User Authentication**
2. Scroll down to the section labeled **API Access Tokens**.
3. Under **Access token expiration**, click the toggle for **Custom expiration**

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 **Update API Access Tokens** at the bottom to save this change

Restricting API Access Token Usage by IP

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

1. Go to **Corp Manage** > **User Authentication**
2. Under **API Access Tokens**, there is a text box labeled **Restrict usage by IP (optional)**
3. Enter the IP addresses and IP ranges you want to limit token usage to in the text box. IP addresses must each use a new line.
4. Click **Update API Access Tokens** at the bottom to save this change

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")
        token    = os.Getenv("SIGSCI_TOKEN")
)

// 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
        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)
        }

        // Print out corp data
        fmt.Printf("%+v\n", corpResp.Data)
}
```

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

```
# 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://dash
```

Agent

Agent Release Notes

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
- Updated base geo IP data: January 2022

4.24.1 2021-12-10

- 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`

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 revproxy 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

4.18.0 2021-04-28

- Fixed a JSON parsing issue with strings ending in `\`
- Added initial functionality to support future GraphQL parsing
- Updated base geo IP data: April 2021

4.17.0 2021-03-04

- Improved SQLi processing
- Improved CMDEXE detection
- Updated base geo IP data: March 2021

4.16.0 2021-02-01

- Added Alpine 3.12 support
- Added initial support for envoy v3 APIs needed to run envoy with deprecated v2 API support disabled
- Fixed version reported by `--version` and other help/usage texts
- Improved redaction logic for `jsessionid` query parameters
- Improved CMDEXE processing
- Updated the Windows installer to install the agent service with a delayed automatic start to avoid a rare failure to start on boot
- Updated base geo IP data: January 2021

4.15.0 2020-12-16

- Fixed startup hang on tls-key files with trailing whitespace
- Added `windows-eventlog-level` configuration option to limit Windows event viewer logging, which now defaults to “warning” (was “info”) to reduce default logging output
- Updated third party dependencies

- Updated base geo IP data: October 2020

4.13.0 2020-09-15

- Improved revproxy upstream error reporting
- Added back signals missing from statsd output
- Added runtime support for future rate limiting enhancements
- Updated base geo IP data: September 2020

4.12.0 2020-08-11

- Improved statsd output by filtering out internal rate limiting metrics inadvertently translated as signals
- Added support on Windows to write select logs to the eventlog in addition to the file based logging
- Updated base geo IP data: August 2020

4.11.0 2020-07-16

- Fixed systemd support for Ubuntu 18.04
- Improved SQLi and CMDEXE detection
- Upgraded to Golang 1.14.5
- Updated base geo IP data: July 2020

4.10.0 2020-06-25

- Added support for additional blocking codes and redirects in revproxy and envoy modes
- Deprecated the `inspection-alt-response-codes` concept in favor of using all codes 300-599 as “blocking”
- Removed `X-Sigsci-*` HTTP response headers when blocking in envoy
- Fixed a revproxy configuration regression issue which caused a failure to connect to the upstream when the upstream URL was configured without explict ports for http (80) or https (443)
- Improved the reverse proxy `pass-host-header` configuration option to allow the hostname to be passed through to the upstream TLS handshake for SNI and certificate validation, avoiding the need to configure `tls-verify-servername`

4.9.0 2020-06-04

- Improved HTTP/2 support for reverse proxy listeners and upstreams
- Improved TLS support for reverse proxy listeners and upstreams
- Improved processing of client IP headers in Azure environments
- Improved CPU and memory performance
- Fixed revproxy and envoy modes so that they register the module with the dashboard on agent startup
- Fixed issue with some lists using non-ASCII characters
- Fixed parsing time duration values as integers in configuration flags and environment vars in addition to config files
- Upgraded to Golang 1.14.3
- Updated base geo IP data: June 2020

4.8.0 2020-05-11

- Added support for disabling revproxy upstream connection pooling with `conn-idle-max=0` and clarified the documentation
- Improved XSS, SQLi and CMDEXE detection
- Upgraded to Golang 1.13.10
- Updated base geo IP data: May 2020

4.7.0 2020-04-08

- Added experimental support for encrypted TLS keys for revproxy via the `tls-key-passphrase` option
- Added experimental jaeger tracing support for the envoy module via the `jaeger-tracing` option
- Added Unix domain socket support for the envoy grpc listener (as it was documented)
- Added Alpine 3.11 .apk support
- Improved SQLi detection
- Improved error handling of upstream HTTP/2 errors in revproxy to return 502 instead of 500
- Improved accuracy of some latency metrics
- Updated UserAgent field to not URL decode by default (decode only if required)

- Improved support for Windows installs using custom install location via `INSTALLDIR`
- Removed concurrent-write problem afflicting GOSH FilterFun when called from PRE/POST (INIT-time was ok)
- Improved XSS, SQLi and CMDEXE detection
- Added support for alternative blocking codes with envoy and revproxy via the `inspection-alt-response-codes` option

4.5.0 2020-02-06

- Improved latency for envoy integration
- Improved logging/metrics/debugging for envoy integration
- Updated `max-connection` to have a default based on the number of workers (typically set via `max-procs`) instead of defaulting to unlimited
- Added support for utilizing `max-connections` in envoy integration
- Improved support for Ambassador using existing envoy integration
- Added Debian 10 (buster) support
- Added CentOS 8 (el8) support
- Updated base geo IP data: February 2020

4.4.1 2020-01-21

- Updated the underlying rule execution engine to be more strict with parsing

4.4.0 2020-01-09

- Improved SQLi and PHP code injection detection
- Enabled HTTP/2 support for reverse proxy upstreams
- Improved response streaming for reverse proxy listeners
- Fixed extracting the Path and Query when processing requests without a URI field
- Upgraded to Golang 1.13.5

4.3.0 2019-12-05

- Added a workaround in Envoy gRPC mode for cases where HTTP/2 body data is missing
- Updated base geo IP data: December 2019

4.2.0 2019-11-11

- Optimized text matching under certain conditions
- Added `remove-hop-header` option in Reverse Proxy to mitigate HTTP request smuggling
- Added experimental `expose-raw-headers` option for added visibility into HTTP request smuggling
- Added WebSocket inspection of JSON payloads in Reverse Proxy
- Updated base geo IP data: November 2019

4.1.0 2019-10-03

- Updated base geo IP data: October 2019
- Upgraded to Golang 1.12.10

4.0.0 2019-09-17

- Added new functionality to speed list processing, which will make agent decisioning even faster
- Fixed a race condition that could prevent startup in Envoy gRPC mode

3.27.0 2019-09-02

- Added experimental support in Reverse Proxy to add a `Connection: close` header to responses for requests that may not be safe to continue
- Added support in Reverse Proxy to capture all inbound request headers
- Added support for setting application request headers
- Improved gRPC call cancelation detection for Envoy Proxy
- Upgraded from Golang 1.11 to 1.12.8

3.26.0 2019-07-09

- Updated base geo IP data: July 2019

3.25.0 2019-06-11

- Fixed false negative with XSS detection
- Fixed false negatives related to Transact-SQL
- Improved XSS javascript on-event detection
- Added signatures for Windows binaries
- Improved foundational architecture for future support of Envoy Proxy with better handling of timeouts
- Updated base geo IP data: June 2019

3.24.1 2019-05-30

- Improved detection of XML content-type to ensure request body will be processed

3.24.0 2019-05-20

- Improved XSS javascript on-event detection
- Fixed parsing the client IP when multiple headers (e.g., `X-Forwarded-For`) are present
- Fixed a race condition in the network interface "upstart service" configuration
- Fixed issue with how `rpc-workers` configuration value is parsed
- Added `inspection-*` options for revproxy and envoy
- Improved foundational architecture for future support of Envoy Proxy with better scalability and configurability
- Updated base geo IP data: May 2019

3.23.0 2019-04-29

- Fixed issue with how `max-procs` configuration value is parsed
- Fixed issue with commandline only options being bound to env vars (e.g., `SIGSCI_VERSION`)
- Added a `statsd-type` option when using a `dogstatsd` statsd server. Enabling this new option will allow for more intuitive reporting within Datadog.
- Improved foundational architecture for future support of Envoy Proxy with better detection of partial request body data

3.22.0 2019-04-10

- Improved foundational architecture for future support of Envoy Proxy by natively supporting the request body in Envoy 1.10+ without using Lua
- Fixed how the Reverse Proxy handles clients closing connections mid-request; it now logs 499 rather than 500
- Updated base geo IP data: April 2019

3.21.0 2019-03-21

- Fixed an issue in which a handful of agents were not receiving rule updates
- Improved support for dynamic geo IP updates to eliminate routine geo updates in the agent

3.20.0 2019-03-11

- Added support for dynamic geo IP updates to eliminate routine geo updates in the agent
- Updated base geo IP data: March 2019 (future updates will be dynamic)

3.19.1 2019-02-21

- Improved foundational architecture for future support of Envoy Proxy by improving error handling and logging

3.19.0 2019-02-11

- Improved multi-part processing
- Updated base geo IP data: February 2019

3.18.0 2019-02-04

- Fixed Reverse Proxy `inspection-timeout` so that the configured `inspection-timeout` is respected instead of waiting indefinitely for request analysis to complete
- Added Reverse Proxy queuing logic similar to how the agent works

- Removed full stack log in reverse proxy if the handler is aborted after response headers are sent

3.17.0 2019-01-09

- Added docker cpu cgroup detection - the agent detects a container start with `--cpus 4` as 4 cpus and adjust settings accordingly
- Improved XSS inspection (false negative)
- Updated Geo IP data: January 2019

3.16.0 2018-12-11

- Improved foundational architecture for future support of Envoy Proxy by improving how some dates are calculated
- Updated Geo IP lookup to resolve a few cases of incorrect countries being reported
- Updated Geo IP data: December 2018

3.15.1 2018-12-04

- Addressed Windows installer issue which could have caused the agent not to upgrade
- Improved foundational architecture for future support of Envoy Proxy by removing some known limitations: responses, HTTPxxx, login and registration signals can now be processed by the agent

3.15.0 2018-11-27

- Added foundational architecture for future support of Envoy Proxy
- Improved logging to capture egress proxy settings and better troubleshoot future issues

3.14.0 2018-11-14

- Upgraded from Golang 1.9 to 1.11
- Updated Geo IP lookup to resolve a few cases of incorrect countries being reported
- Updated Geo IP data: November 2018

3.13.0 2018-10-09

- Fixed rare instance where uploader may crash while fetching CPU statistics
- Updated Geo IP data: October 2018

3.12.1 2018-09-24

- Fixed an issue where the upload service may crash on startup
- Improved logging around agent service restarts on failure
- Improved help/usage text

3.12.0 2018-09-06

- Removed ulimit `data` and `as` 1gb constraint from upstart config. If needed, it is recommended to set to 1/4 the memory in `/etc/init/sigsci-agent.override`.
- Added a `statsd-metrics` filter option
- Improved config validation
- Improved logging
- Improved handling of file path separators in the configuration by normalizing them to the OS native format
- Added properties (version, icon, etc.) to the Windows executable
- Improved the Windows MSI packaging
- Added support for configuring multiple reverse proxy listeners from the command line or environment
- Improved CMDEXE inspection (false positives)
- Instrumented more memory information
- Documented experimental `statsd-metrics` descriptions
- Added the ability to decorate signals with meta data
- Fixed how path is decoded in URLs - do not decode `+` as a space
- Updated third party dependencies
- Updated September Geo IP

3.11.0 2018-08-08

Improved defaults for `max-procs`, `max-backlog`, and `max-records` based on number of CPU cores detected - especially on larger machines
- Improved performance of request/response context tracking
- Improved performance of RPC service
- Updated third party dependencies
- Updated sigsci-module-golang with latest version for the reverse proxy
- Updated August Geo IP

3.10.1 2018-07-17

- Fixed 3.10.0 changelog typos
- Fixed crash handling a fatal RPC listener service error
- Improved logging and handling of all fatal service errors

3.10.0 2018-07-10

- Updated the RPC address on Windows to use TCP by default (`127.0.0.1:737`)
- Fixed race in quieting reverse proxy logging (upstream fix)
- Updated third party dependencies
- Updated July Geo IP

3.9.4 2018-06-26

- Removed extraneous RPC warnings on startup

3.9.3 2018-06-25

- Fixed issue where the older (deprecated) reverse proxy config, via `reverse-proxy-*` configuration options, was not setting the defaults for new configuration values. These values were getting assigned zero values and were not allowing for inspection of the body due to the new `inspection-max-content-length` option being zero.

3.9.2 2018-06-20

- Reduced logging in reverse proxy by default
- Improved ability to close upstream connections when downstream closes in reverse proxy

3.9.1 2018-06-19

- Improved some testing tools
- Updated sigsci-module-golang with latest version for the reverse proxy

3.9.0 2018-06-11

- Improved generated agent documentation
- Enhanced internal architecture without any external changes
- Improved service restarts on configuration updates to allow manual control via new `rpc-reload-on-update` and `revproxy-reload-on-update` options
- Added options to better configure inspection in reverse proxy mode: `inspection-anomaly-duration`, `inspection-anomaly-size`, `inspection-debug`, `inspection-max-content-length`, `inspection-timeout`
- Adjusted default logging verbosity so that common TLS handshake issues do not fill up the logs
- Updated third party dependencies
- Updated June Geo IP

3.8.0 2018-05-02

- Improved the usage text for the reverse proxy options
- Improved generated agent configuration docs page, adding option links
- Improved detection/logging of RPC errors
- Adjusted `max-backlog` setting to scale with `max-procs` by default
- Added `response-header-timeout` and `request-timeout` reverse proxy options
- Improved CMDEXE false positives
- Updated third party dependencies
- Updated May Geo IP

`header-rewriting`

- Improved the usage text for the reverse proxy options

3.6.1 2018-04-16

- Improved CMDEXE false positives
- Improved usage text to document proxy settings
- Improved logging on startup when `log-out` is configured
- Improved rule execution error handling

3.6.0 2018-04-04

- Added more metrics around tracked contexts
- Improved CMDEXE false positives
- Updated April Geo IP
- Updated third party dependencies

3.5.0 2018-03-27

- Updated third party dependencies
- Added support for proxying WebSockets in reverse proxy mode

3.4.0 2018-03-15

- Improved error logging
- Added multipart/form-data support to reverse proxy mode
- Added more logging and TLS options to the reverse proxy listener config: `log-all-errors`, `tls-ca-roots`, and `tls-verify-servername`
- Improved CMDEXE false positives

3.3.0 2018-03-08

- Improved CMDEXE false positives
- Cleaned and standardized agent release notes
- Fixed Debian 9 (Stretch) systemd configuration issue
- Updated Mar Geo IP

3.2.1 2018-03-01

- Fixed potential crash on startup

3.2.0 2018-03-01

- Upgraded to Golang 1.9
- Improved runtime error logging
- Added support for post data parse errors

3.1.0 2018-02-22

- Updated Feb Geo IP
- Cleaned up some config options
- Allowed more flexibility in JSON parser
- Improved performance of GEOIP lookups
- Fixed issue with empty OS field on agents page
- Improved CMDEXE and LFI detection

3.0.3 2018-02-01

- Improved HTML5 parsing and XSS detection
- Improved SQLi false positives
- Updated geoip database

3.0.2 2018-01-12

- Changed copyright year to 2018
- Improved detection of a particular but invalid XSS
- Updated some error reporting metrics for better diagnostics
- Improved logging around detected agent service failure/restart

3.0.0 2018-01-08

- Added support for local country code lookups
- Added support for anonymizing IP addresses
- Added support for multipart form POST
- Expanded rule functionality in preparation for future rule updates
- Expanded feature flagging to allow for easier feature rollouts
- Expanded support for data redaction
- Expanded processing metrics
- Updated third party dependencies

2.2.1 2017-12-18

- Expanded rule functionality in preparation for future rule updates
- Fixed issue where ID/key was still required if in standalone mode

2.2.0 2017-12-04

- Expanded rule functionality in preparation for future rule updates
- Improved error handling of reverse proxy configurations on start and reload
- Fixed minor race condition under heavy service restart loads
- Updated third party dependencies

2.1.2 2017-11-14

- Adjusted some log messages (some too verbose, some not enough)
- Added ability for Windows installer to now start the agent service on installation, if agent.conf is already in place and contains required access keys
- Added support in reverse proxy for multiple listeners and a new configuration syntax while still allowing backwards compatibility: https://docs.signalsciences.net/install-guides/reverse-proxy/
- Improved automated agent configuration docs to be much more descriptive and easier to read: https://docs.signalsciences.net/install-guides/agent-config/
- Fixed issue with service startup on boot with older versions of Windows
- Updated third party dependencies
- Fixed issue when configuring the reverse proxy from ENV vars
- Fixed double config reload on SIGHUP

2.1.1 2017-11-13

- Temporarily reverted back to 2.0.1 (as 2.1.1) while investigating a reported issue with 2.1.0 on some platforms

2.1.0 2017-11-13

- Adjusted some log messages (some too verbose, some not enough)
- Added ability for Windows installer to now start the agent service on installation, if agent.conf is already in place and contains required access keys
- Added support in reverse proxy for multiple listeners and a new configuration syntax while still allowing backwards compatibility: https://docs.signalsciences.net/install-guides/reverse-proxy/
- Improved automated agent configuration docs to be much more descriptive and easier to read: https://docs.signalsciences.net/install-guides/agent-config/
- Fixed issue with service startup on boot with older versions of Windows
- Updated third party dependencies

2.0.1 2017-10-31

- Clarified release notes for 2.0.0

- Expanded rule functionality
- Removed all deprecated agent configuration options: `debug-log-rule-updates`, `site-keys`
- Improved config download failover error handling
- Fixed a race condition when a very small `download-interval` is used

1.23.4 2017-09-29

- Fixed false positive in CMDEXE

1.23.3 2017-09-28

- Fixed false positive in CMDEXE

1.23.2 2017-09-27

- Improved CMDEXE, SQLi and XSS detection
- Fixed issue where redacted iban/guid was not marked with the redaction type

1.23.1 2017-09-07

- Improved signal filtering
- Added tracking of GCE cloud deployment
- Reverted issue with RPC version compatibility

1.23.0 2017-09-06

- Improved CMDEXE and SQLi detection
- Added tracking of Azure cloud deployment
- Fixed issue calculating the connection open metric
- Fixed issue where redacted CC numbers were not marked with the redaction type
- Added support for configuring a failover download url via `download-failover-url`
- Fixed issue with RPC version compatibility
- Changed order in which dynamic config is applied allowing local overriding
- Changed log timestamps to microsecond resolution

1.22.0 2017-08-15

- Improved SQLi detection
- Improved reverse proxy config reload
- Prepped for upcoming HTTP/2 support in reverse proxy
- Allowed setting custom HTTP request headers via `custom-request-headers`
- Removed hardcoded logic to clear signals on allowlist - logic now in rule updates

1.21.0 2017-07-21

- Improved SQLi detection
- Removed old reverse proxy system in favor of the new system
- Disabled keepalives when the reverse proxy config is being reloaded to force new transactions onto the new configuration. In addition, the default timeout for this was moved from 10s to 30s.
- Updated which reverse proxy messages are logged to the UI

1.20.1 2017-06-27

- Added more metrics around inspection
- Fixed issue where reverse proxy was not honoring the `sample-percent`

1.20.0 2017-06-27

- Added more metrics to reverse proxy
- Added a `max-inspecting` config option to control the max transactions the WAF engine can be inspecting in parallel (currently reverse proxy only)

- Fixed issue where dynamic config was not applied on SIGHUP
- Allowed more dynamic service configuration (e.g., change from RPC to revproxy and back with SIGHUP)
- Added ability to log full stack trace and restart service should any service encounter a fatal error
- Isolated reverse proxy from agent errors
- Fixed race between downloader/SIGHUP handlers under heavy config change load
- Changed default 'download-interval' to 30s from 1m
- Improved SQLi detection

1.18.2 2017-05-02

- Added ability to reload the local config on a SIGHUP
- Added ability to log when a config option is changed, but not reloadable
- Added optional field RPCMsgIn#RequestID that allows a module to pass a RequestID (24 char hex) to use

1.18.1 2017-04-27

- Disabled restarting (zero downtime) reverse proxy on Windows due to inconsistent support
- Fixed potential panic with beta reverse proxy startup on Windows
- Quieted down some logging

1.18.0 2017-04-24

- Added ability to parse XML for processing via the agent

1.17.3 2017-04-20

- Fixed resource leak in configuration reload
- Fixed redaction of ID/key in log when using two argument form of CLI flags
- Removed deprecated `sigsci-configure` utility

1.17.2 2017-04-11

- Improved handling of Windows platform for zero-downtime restarts
- Made restarts less verbose

1.17.1 2017-04-06

- Added ability to restart (zero downtime) reverse proxy on config download

1.17.0 2017-03-27

- Fixed TLS reverse proxy listener handshake delivering HTTP
- Required the TLS 1.2 mandatory TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 cipher suite
- Improved compatibility in TLS HTTP handshake
- Added configurable reverse proxy listener read/write/idle timeouts
- Enabled versioned configuration by default
- Improved CMDEXE and PHP code injection functions

1.16.0 2017-03-14

- Improved JSON parser
- Defaulted to no access log in reverse proxy mode, `reverse-proxy-accesslog` will enable this feature
- Updated TLS ciphers to latest supported
- Reduced time till serving requests when starting in reverse proxy mode (typically under 10ms)

1.15.3 2017-02-28

- Fixed issue where agent internal services may stop on error
- Fixed issue where agent could not startup in standalone mode

1.15.2 2017-02-27

- Fixed potential crash when the reverse proxy didn't have permission to write to the access log

1.15.0 2017-02-24

- Disabled requirement of WAF config download before starting, allowing faster startup
- Added accesslog for reverse proxy mode via `reverse-proxy-accesslog`
- Added support for multiple reverse proxy upstreams
- Improved processing of `client-ip-header`
- Added `local-networks` option for more accurate client IP parsing
- Enabled specifying time durations as string vs nanosecs (e.g., “10s” vs 10000000000)
- Added ability to shutdown reverse proxies gracefully (see `reverse-proxy-shutdown-timeout`)
- Allowed config of all reverse proxy network parameters
- Allowed config of reverse proxy TLS min version, cipher suites, etc
- Allowed internal/self-signed certs on the upstream (default false) `reverse-proxy-insecure-skip-verify`
- Allowed more dynamic configuration of agent for future UI work
- Enabled restart of periodic services (uploader, downloader, etc.) on reconfiguration
- Corrected various minor SQLi false positive issues
- Deprecated use of `site-keys` option, support will be removed in a future release
- Updated third party dependencies

1.14.4 2016-12-16

- Improved stats collection via `sigsci-agent-diag`
- Improved separation of Windows and Unix code
- Improved upcoming config download versioning

1.14.3 2016-12-09

- Improved SQLi false positives
- Added more performance related stats collection to `sigsci-agent-diag`
- Added ability to collect agent profiling data via `sigsci-agent-diag`
- Improved handling of large POST and JSON payloads

1.14.2 2016-11-21

- Improved parsing of `client-ip-header` values

1.14.1 2016-11-15

- Improved SQLi detection
- Moved generic Linux and Windows artifacts to Linux/Windows directories

1.14.0 2016-11-10

- Added support for new config download format and versioning
- Improved SQLi detection
- Prepped for future custom rule expansions and detector ordering enhancements
- Added more performance related stats collection to `sigsci-agent-diag`
- Added metric to monitor context misses due to expired context
- Enabled adjusting the context expiration (`context-expiration`)

1.13.4 2016-10-03

- Internal release
- Fixed CHANGELOG release date for 1.13.3

1.13.3 2016-09-28

- Fail open more gracefully by returning an “OK” agent response when agent is “off”
- Added logging of `sample-percent` setting on agent startup
- Added logging of request processing mode changes (e.g., agent mode changed in UI)

1.13.2 2016-09-21

- Improved CentOS 5 initscripts
- Added new engine function: SetPath which allows for custom redactions of the path
- Updated third party dependencies

1.13.0 2016-09-09

- Added initial support for using TLS in reverse proxy mode
- Removed binaries from archive generated by `sigsci-agent-diag`
- Fixed container detection in `sigsci-configure` on systemd platforms
- Improved to allow only user/group access to read the config after using `sigsci-configure`
- Updated third party dependencies
- Added ability to collect log configured with log-out in `sigsci-agent-diag`

1.12.1 2016-08-23

- Fixed error displayed when running `sigsci-configure` on some platforms
- Added more diagnostics around docker/container installs to `sigsci-agent-diag`

1.12.0 2016-08-22

- Added diagnostics utility `sigsci-agent-diag` to help troubleshoot install issues
- Added Alpine Linux support! The released tarball (sigsci-agent-*version*.tar.gz) now contains a 100% static binary that will work on all Linux operating systems. In addition, this agent is compiled under Golang 1.7.0. Existing deb/rpm based packaging continue under 1.6.3.
- Updated third-party libraries to latest

1.11.4 2016-08-18

- Updated third party dependencies

1.11.3 2016-07-21

- Improved systemd support to start on reboot
- Added ability to automatically start agent on initial install and reboot

1.11.2 2016-07-20

- Restored https://dl.signalsciences.net/sigsci-agent/sigsci-agent_latest.tar.gz
- Made no operational changes

1.11.1 2016-07-19

- Corrected version number reporting
- Updated third-party dependencies

1.11.0 2016-07-14

- Added support for Ubuntu 16.04
- Switched to SemVer

1.10.8048 2016-07-05

- Improved SQLi detection
- Added Rules Engine v2 containing the following new functions
  - `SetClientIP`
  - `SetProtocol`
  - `SetTLSProtocol`
  - `SetTLSCipher`
  - `Reverse`
  - `StringReverse`
  - `DeepEqual`
  - `AddrIsPrivate`

- NewGlobMatcher

- Updated third-party dependencies

1.9.8026 2016-07-05

- Improved cleanup routines to be more efficient for higher capacity sites
- Allowed control of RPC workers via `rpc-workers` (default `rpc-workers=max-procs`)
- Added profiling option via `debug-profile=cpu|mem|block[,dir]`
- Cleaned up help text
- Cleaned up logging
- Improved Windows service support
- Updated third-party dependencies
- Fixed potential CPU metrics concurrency issue

1.9.7763 2016-06-07

- Improved agent startup messages for better diagnostics
- Added more agent logs to upload for better diagnostics
- Removed some extraneous cleanup on agent startup

1.9.7753 2016-06-06

- Improved agent startup messages to aid in debugging
- Added additional information on the agent's cgroup to be collected (Linux)
- Improved detection if running inside a docker container
- Improved Windows support
- Fixed stray logging call
- Updated third-party dependencies

1.9.7623 2016-05-24

- Changed the default listener address to `unix:/var/run/sigsci.sock`
- Started an additional legacy listener on the old `unix:/tmp/sigsci-lua` socket to aid in migrating modules
- Added support for more redaction types in the agent
- Improved redaction so the query string is now removed instead of confusingly replacing with "?redacted"
- Added experimental reverse proxy support to agent currently targeted at demos only
- Allowed the agent to better scale across available CPU cores by default by basing the default `max-procs` setting on total cores available: 1-3: `max-procs=1`, 4-5: `max-procs=2`, 6-15: `max-procs=3`, 16+: `max-procs=4`
- Added support for a new `RPC.ModulInit` call for future module use allowing better version tracking without requiring traffic
- Moved tagging of HTTP codes to the rules, which can be updated dynamically
- Upgraded some third party dependencies

1.8.7087 2016-04-10

- Added support for RHEL/CentOS 5
- Updated third-party dependencies

1.8.7007 2016-04-06

- Fixed bug in RPM packaging script for EL7 to make sure the systemd daemon config is reloaded on install/upgrade

1.8.6993 2016-04-05

- Added a more informative hello message to be displayed on agent start
- Added more control headers for testing with `-debug-rpc-test-harness`
- Fixed bug in RPCv1 protocol (e.g., `-rpc-version=1`) that could deadlock when connections were reused
- Added ability to export an agent PID metric to the collector
- Added new metric `agent.upload_metadata_failures` for number of http failures uploading data to the collector

1.8.6480 2016-02-26

1.8.6347 2016-02-17

- Added new flag, `-debug-rpc-test-harness` enables a mode to test RPC calls

1.8.6055 2016-02-03

- Fixed SQLi false positive involving a common English phrase
- Removed XSS false positive that occurred in unfortunate base64 encoded strings
- Made packaging fixes

1.8.5758 2016-01-19

- Added new flag, using `-debug-log-dropped-connections=1` which produces errors messages on why a connection was dropped.
- Added new flag, `-max-backlog` which controls the number of request that can be backlogged, currently defaults to 100
- Renamed flag, `-max-queue` to `-max-records` to better describe what it is: the maximum number of records that can be stored before being sent to the collector

1.8.5694 2016-01-13

- Made internal improvements in CPU utilization
- Improved handling of upload / download timeouts (followup from 1.8.5041)
- Added additional sanity checks around Unix domain socket listener to prevent multiple agents running concurrently
- Improved XSS false positives with clients uploading fully formed HTML or XML documents
- Fixed incorrect start command for upstart in sigsci-configure script

1.8.5304 2015-12-14

- Added ability to sample input requests via `-sample-percent` flag
- Added additional metrics collected on bytes read and written to web server, and CPU performance
- Improved XSS detection

1.8.5217 2015-12-09

- Improved performance and latency
- Reduced amount of data sent back, improving performance
- Made under the hood adjustments to enable future custom rules

1.8.5041 2015-12-01

- Reduced amount of data transmitted from agent to collector by up to 90%, resulting in better performance and latency
- Made rule updates gracefully timeout and retry if the network is stalled
- Added detection for MariaDB-specific SQLi

1.8.5016 2015-11-30

1.8.4972 2015-11-23

- Improved connection timeout handling for collector uploads

1.8.4891 2015-11-18

- Improved `Agent Off` mode to do even less work
- Fixed XSS false positive for inputs with benign embedded HTML involving background images
- Added new flag, `-max-connections` to control the number of simultaneous connections the agent can process. If the number is exceeded the connection is dropped. By default, there is no limit, but may change in the future.
- Added additional metrics collection on connections and request types that will appear on agent dashboards
- Partially restructured internal locking to reduce latency under high loads and concurrency
- Refreshed internally-used, third-party libraries (from the command line type `agent -legal` for the bill of materials)

1.8.4405 2015-10-21

- Changed it so agent now tokenizes the query string and post data in two ways simultaneously to handle platform differences (Ruby, Python, Golang uses one way, and PHP, `Node.Js`, .Net. does it another) to minimize false negatives

1.8.4284 2015-10-13

- Added redaction of query string in HTTP response header `Location`
- Added ability for “off mode” to still count number of requests coming in, which helps agents in debugging and in estimation of load
- Added inspection of top level JSON arrays (JSON objects already unpacked). For example input of `foo=bar&obj=["something", "apple"]` the values in the `obj` are now inspected for attacks. Input of `foo=bar&obj={"something", "apple"}` was already being inspected correctly. This improves reduction of both false positives and false negatives.
- Added redaction of sensitive data in the unlikely corner case of an “attack in the URI path (not the query string!) that contained a credit card”
- Included Golang runtime version in the Bill of Materials (`agent -legal`)
- Changed AgentEnabled to now indicate if the agent is processing requests or not; `0` means off, while `1` means it’s processing requests normally

1.8.4201 2015-10-08

- Fixed XSS false positive in fully formed XML documents that are POSTed

1.8.4186 2015-10-06

- Improved agent “off mode” to do even less work
- Added Bill of Materials reporting in agent, from the command line type `agent -legal` for details
- Added additional system metrics collection to aid in debugging
- (1.8.4180 and 4182 were redacted)

1.8.4053 2015-09-25

- Fixed configuration field parsing issue

1.8.4015 2015-09-21

- Added support for multiple sites on a single agent
- Migrated configuration file format from `INI` style to TOML
- Removed deprecated agent flags: `ssnet-active`, `ssnet-address`, `server-address`, `server-active`, `server-timeout`

1.8.3900 2015-09-03

- Fixed incorrect `provides` declaration in SysV init script

1.8.3874 2015-09-02

- Improved detection of XSS and SQLI in the URL path
- Improved XSS accuracy and performance
- Added ability to explicitly change number of CPUs used via command line `-max-procs`
- Added ability to manage maximum memory used by limit internal queue size via `-queue-length`
- Improved serialization
- Added and improved various agent metrics
- Improved ability to create more flexible blocking or blocklist rules

1.8.3719 2015-08-24

- Fixed incorrectly set response times of pure 404 errors
- Improved debug logging

1.8.3704 2015-08-24

- Fixed regression in 3611 release where 404 errors were not being recorded
- Made major improvement in concurrency which may provide up to 75% performance boost on high volume websites
- Started major rules engine upgrade

1.8.3611 2015-08-17

- Added ability to capture HTTP request and response headers (minus sensitive ones)
- Allowed custom rules (part 1)

- Improved performance and memory usage
- Improved SQLI and XSS detection

1.8.3385 2015-07-30

- Changed all internal counters to 64-bit integers, which allows long running agents to handle more than 4 billion requests and very large file outputs to be properly handled
- Made sure all errors get properly trapped and sent upstream, which will aid in remote debugging and better visibility on the dashboard
- Improved precision and accuracy in detecting SQLi attacks
- Added ability to receive URL scheme information (i.e. `http` or `https`)
- Added ability to receive TLS (SSL) protocol and cipher suite information from modules. For best results update the module to at least:
  - Apache 214
  - NGINX 1.0.0+346

1.8.3186 2015-07-22

- Added ability for agent (along with module) to set `X-SigSci-Tags` request headers indicating what tags or signals where detected in the request. For best results upgrade the module to at least:
  - Apache 207
  - NGINX 1.0.0+343
- Improved precision and accuracy in detecting SQLi

1.8.2964 2015-07-06

- Made internal changes to enable upcoming features

1.8.2950 2015-07-02

- Fixed sigsci-configure to now return the correct start command for the init system in use on installed system
- Added `password_confirmation` to built-in list of fields to redact
- `-debugStandalone` flag changed from `true`, `false` to `0` (normal behavior), `1` (no downloads), `2` (no uploads), and `3` (no network connections at all)

1.8.2718 2015-06-14

- Fixed issues where the Signal Sciences dashboard would show an incorrect “Agent Response” of 0. For best results, please upgrade the module to
- Apache 2.2.139 or Apache 2.4.139
- NGINX 1.0.0+320

1.8.2681 2015-06-10

- Improved documentation and help of command line flags and `-help`
- Reduced SQLi false positives

1.8.2327 2015-05-15

- Made allowlisting bug fixes and improvements
- Made data redaction bug fixes and improvements
- Removed legacy communication protocol

1.7 2015-04-16

- Added IntervalSet stuff to agent
  - #1689 sensitive parameter sanitization
  - #447 Inspection of JSON
  - #1720 improvements in libinjection to reduce false positives for SQLI
  - #1744 ditto for XSS
  - #1799 performance improvements in 400, 500 http errors
  - #1797 debug log improvements
  - #1851 XSS false positives

1.6 2015-02-13

- #1498 - Improved logging around WAF rule updates with `-debug-log-rule-updates` 0,1,2 (0=off, 1=updates only, 2=more...)
- #1141 - Made libinjection enhancements to detect certain attacks on IBM servers
- #741 - Added ability for agent to return timezone and zone offset information

1.5 2015-01-22

- Bumped minor version number to reflect new build process

1.4 2015-01-15

- Made minor performance improvement https://github.sigsci.in/engineering/sigsci/issues/1410
- Fixed libinjection xss
- Fixed agent to no longer send back entire query string #861
- Added various new stats
- Added ability to send back cli args #1140
- Added ability to send back localtime and utc time #749

1.3 2015-01-15

- Implemented major stability improvements

1.2.1 2015-01-13

- Added ability to set which request header contains the requesting client IP, see flag `-client-ip-header`

1.2 2015-01-13

- Added new option `-debug-log-all-the-things`, which turns on all logging (expensive!)
- Renamed option `-log-uploads` to `-debug-log-uploads`

1.1.1 2015-01-08

- Added new network code, matches module ver 0.06
- Changed connection to collector from TLS 1.0 to TLS 1.2
- Changed `-debug-log-web-inputs` and `-debug-log-web-outputs` from booleans, now it takes 0,1,2

1.1.0 2014-12-23

- Bumped minor version for Golang 1.4

1.0.4 2014-11-29

- UDS
- Dropped json

1.0.3 2014-11-29

- Added more errors to be logged and sent upstream

1.0.2 2014-11-29

- Added AgentBuildID to meta data
- Made other changes to the WAF agent

Datadog

Events Feed

Our Datadog event integration creates an event when IPs are flagged on Signal Sciences.

Adding a Datadog integration

1. Within Datadog, go to **Integrations** then **APIs**.
2. Press the **Create API Key** button and follow directions.
3. Copy the provided **API Key**.

6. Enter the **API Key** in the API Key field.
7. Click **Add**.

Activity types

Activity type	Description
flag	An IP 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>`.

        - Example:`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:

    - Add 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.

Architecture

What is the Signal Sciences architecture?

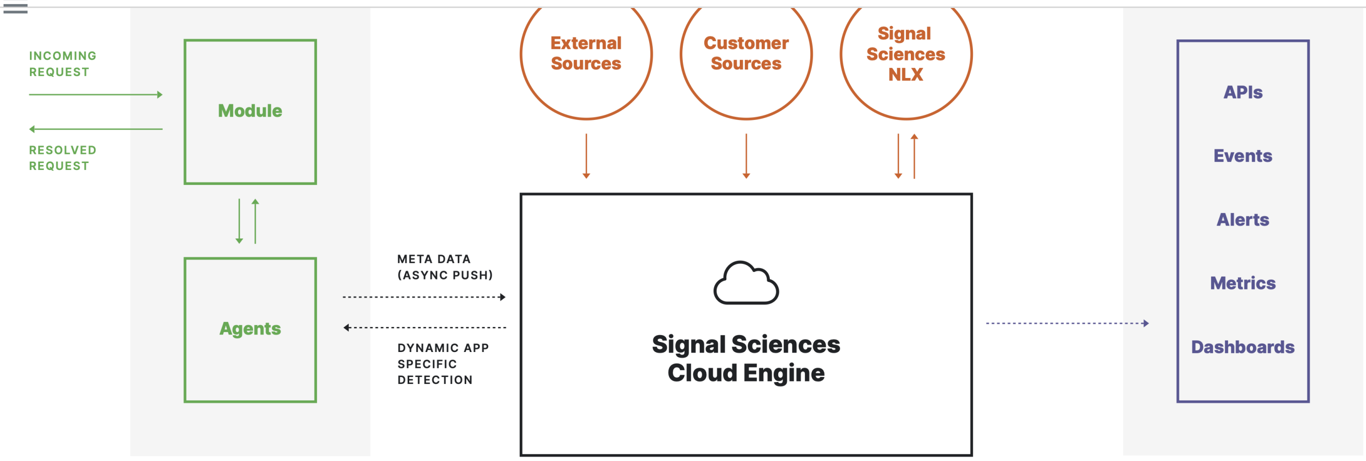
The Signal Sciences platform is an application security monitoring system that proactively monitors for malicious and anomalous web traffic directed at your web servers. The system is comprised of three key components:

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

The module is the architecture component that is responsible for directly interacting with requests. It listens for incoming requests and passes them to the agent for a decision. After receiving a decision from the agent, the module will block, allow, or tag requests in accordance 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 PHP, Node.js, Java, .NET, and Python, 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`
- `sigsci-agent-wafconf.s3.amazonaws.com`
- `sigsci-agent-wafconf-us-west-2.s3.amazonaws.com`

If the agent is unable to download from the primary S3 bucket, it will fallback to a secondary bucket in a second region until it can download from the 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.

What sort of scale do you support?

Our architecture allows us to support applications with high traffic volume. We are deployed across full production with companies in the top 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, 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.

Detection

Can Non-Datacenter Traffic be tagged as an anomaly?

By default Signal Sciences tags datacenter IP addresses as an anomaly. Tagging non-datacenter IP addresses as an anomaly can be achieved with a custom rule.

What does the Backdoor signal identify?

Our backdoor signal generally matches known backdoor filenames, many of which have been traditionally PHP (admin.php, r57.php, etc). 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.

How are JSON API payloads inspected and redacted?

Signal Sciences will automatically parse all JSON key/value pairs and treat them as any other request parameter so attack and anomaly detection, custom signals and redactions will all work properly in the context of these requests.

For example in the following sample requests we can see how redactions would work within the context of a request.

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}
```

Sent to Signal Sciences

```
POST /request HTTP/1.1
Host: api.example.com

password=
```

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"}
```

Sent to Signal Sciences

```
POST /request HTTP/1.1
Host: api.example.com

zip=<script>alert(1)</script>
```

Installing the Java Module as a Jetty Handler

Requirements

- Jetty 9.2 or higher

Supported Application Types

support if you are interested in doing this.

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 (which is `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"
````
~~~~~

Installation

The installation of the Jetty module varies slightly depending upon how you have deployed Jetty (i.e. embedded vs. stand alone).

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".

Download

Download manually

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

Access with Maven

For Java projects using Maven for build or deployment, the Signal Sciences Java modules can be installed by adding the following to the

Install

Embedded Jetty

The Signal Sciences Jetty module is currently implemented as a Handler. To use this, you will need to make a simple change to your

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

Simple Example Server

For a more complete example, see the `sigsci-jetty-simple-example` JAR files in the distribution. Included are the binaries, source and javadoc for a simple working example. The binary JAR is executable and can be run with something like the following, which will start the simple server and point it at an agent running on TCP port 5000 on the local host (requires 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
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 Science Simple Example Server started (htt
00:00:00.986 [qtp123456789-12] INFO  c.s.example.RequestLogger - "GET /test/ HTTP/1.1" 302
```

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

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

About the Signal Sciences Service Broker for VMware Tanzu

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

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

Installation

1. Download the product file from Pivotal Network.
2. Navigate to the Ops Manager Installation Dashboard and click **Import a Product** to upload the product file.
3. Click **Add** next to the uploaded Signal Sciences Service Broker tile in the Ops Manager **Available Products** view to add it to your staging area.
4. Click the newly added **Signal Sciences Service Broker** tile.
5. Click the **Buildpack Settings** tab and set the `sigsci_buildpack_decorator` Buildpack Order to zero.
6. Click **Save**.
7. 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.

Rate Limit Rules

> **Note:** Rate limit rules are only included with the Premier platform. They are not included as part of our Professional or Essential platforms.
>
> Rate limit rules require agent version 3.12 or above.

Rate limit rules allow you to define arbitrary conditions (e.g., IP is `198.51.100.50`, method is POST, and path is `/login`) and automatically begin to block or tag requests that pass a user-defined threshold (e.g., 100 requests in 1 minute).

Glossary

Term	Definition
Client	The source from where requests originate
Client Identifier	The parts(s) of requests used to identify an individual client
Threshold	How many requests must be detected before a client is rate limited
Interval	The period of time requests must be detected during to pass the threshold
Counting signal	The signal that needs to cross the threshold for a client to be rate limited
Action signal	The signal that is logged or blocked when a client is rate limited. May be the same or different from the counting signal.
Action	Whether requests are logged or blocked
Duration	How long a client remains rate limited

are visible on the requests page of the console if they have also been tagged with other signals.

2. Requests tagged with the **counting signal** by the rate limit rule are tallied and counted towards the **threshold** of the rule.
3. When enough requests with **counting signals** from a given client are detected and the **threshold** of a rate limit rule is crossed, the client is rate limited.
4. Subsequent requests originating from the rate limited client matching the conditions of the rate limit rule are still tagged with the **counting signal**.
5. Subsequent requests originating from the rate limited client that have been tagged with the **action signal** are tagged with the `Rate Limit` signal.
   - If the **action** is set to "block", the requests are blocked and tagged with the `Blocked Request` signal.
   - If the **action** is set to "log", the requests are not blocked and no additional signals are added.

Example rate limit rules

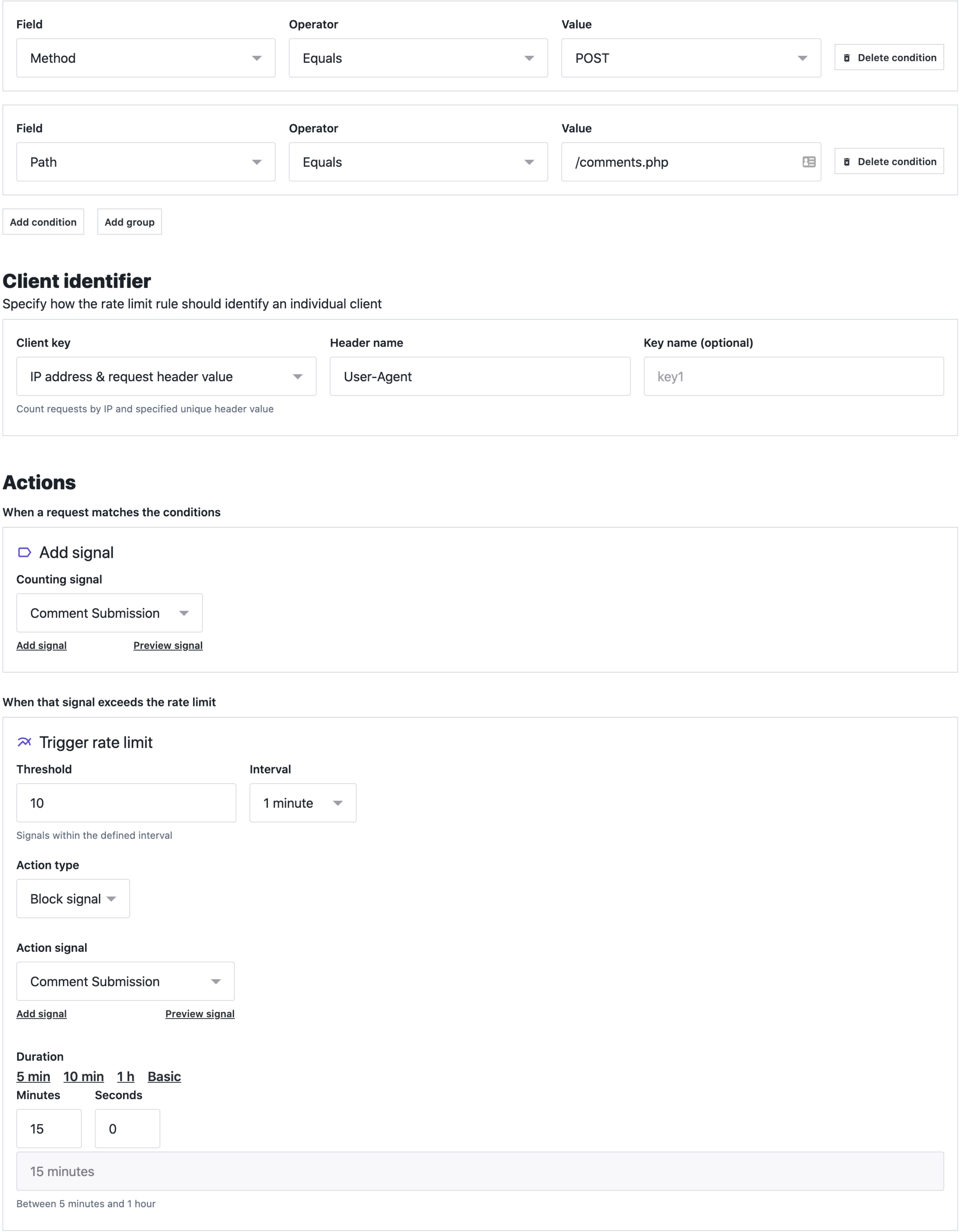
The following example rules demonstrate how to use rate limiting for a couple of common use-cases, illustrating why you may configure your rate limit rules in certain ways. Be aware that the values such as paths and response codes used in these examples may not be the same as those used by your particular application.

Rate limit comment submissions

Rate limit rules can use the same signal for both the **counting signal** and the **action signal**. This example rule demonstrates how to rate limit users' ability to submit comments.

This rule looks for POST requests to the `/comments.php` file and tags them with the `Comment Submission` custom signal as the **counting 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 as the **client identifiers** to track requests from this user.

When 10 requests (the **threshold**) tagged with the `Comment Submission` signal (the **action 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**) for the next 15 minutes (the **duration**).



Field	Operator	Value	
Method	Equals	POST	Delete condition

Field	Operator	Value	
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

When a request matches the conditions

Add signal

Counting signal: Comment Submission

When that signal exceeds the rate limit

Trigger rate limit

Threshold	Interval
10	1 minute

Signals within the defined interval

Action type: Block signal

Action signal: Comment Submission

Duration

Minutes	Seconds
15	0

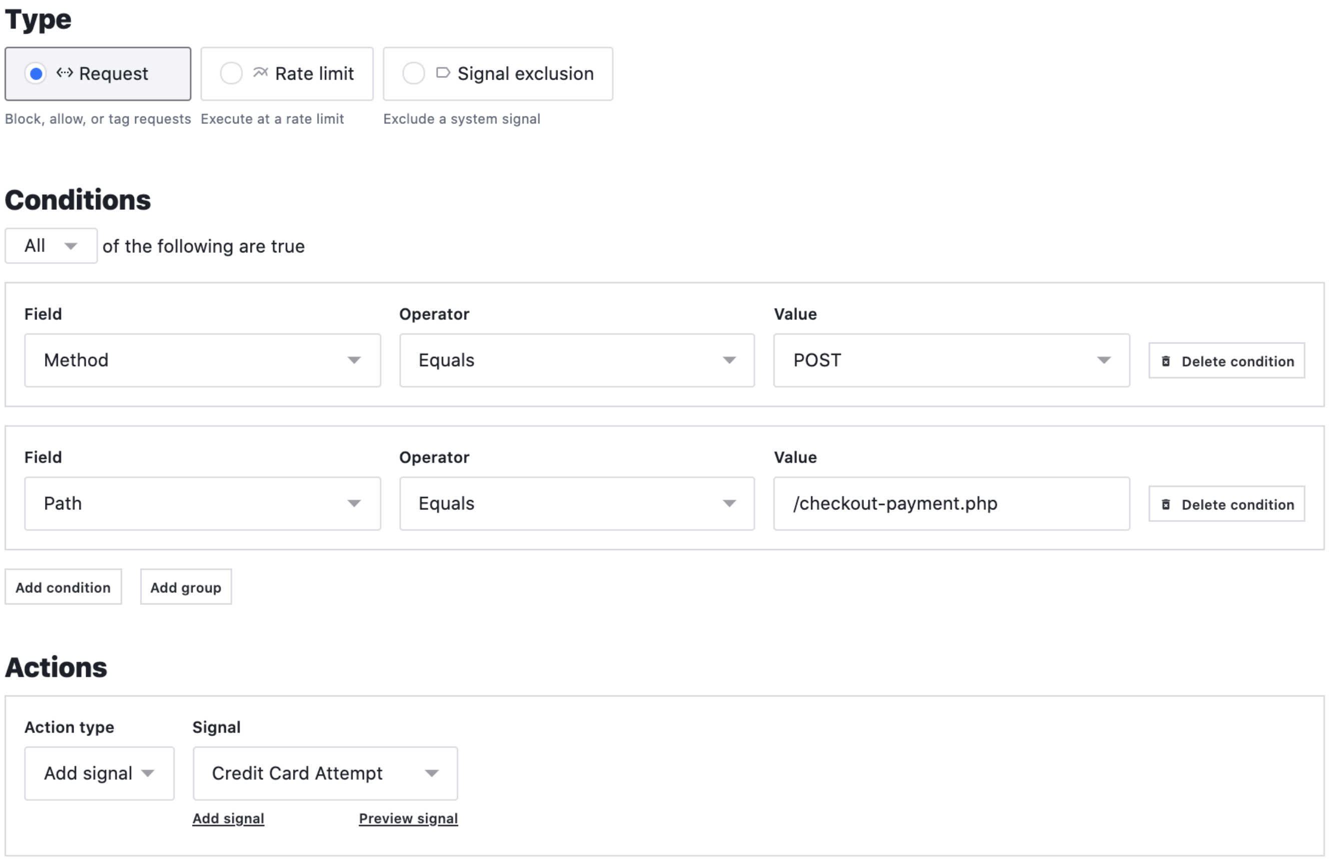
15 minutes

Between 5 minutes and 1 hour

Credit card validation attempts

This example rule demonstrates how to rate limit credit card validation attempts after too many failed attempts. This is example where the **counting signal** and the **action signal** are different.

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 — Execute at a rate limit
○ 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

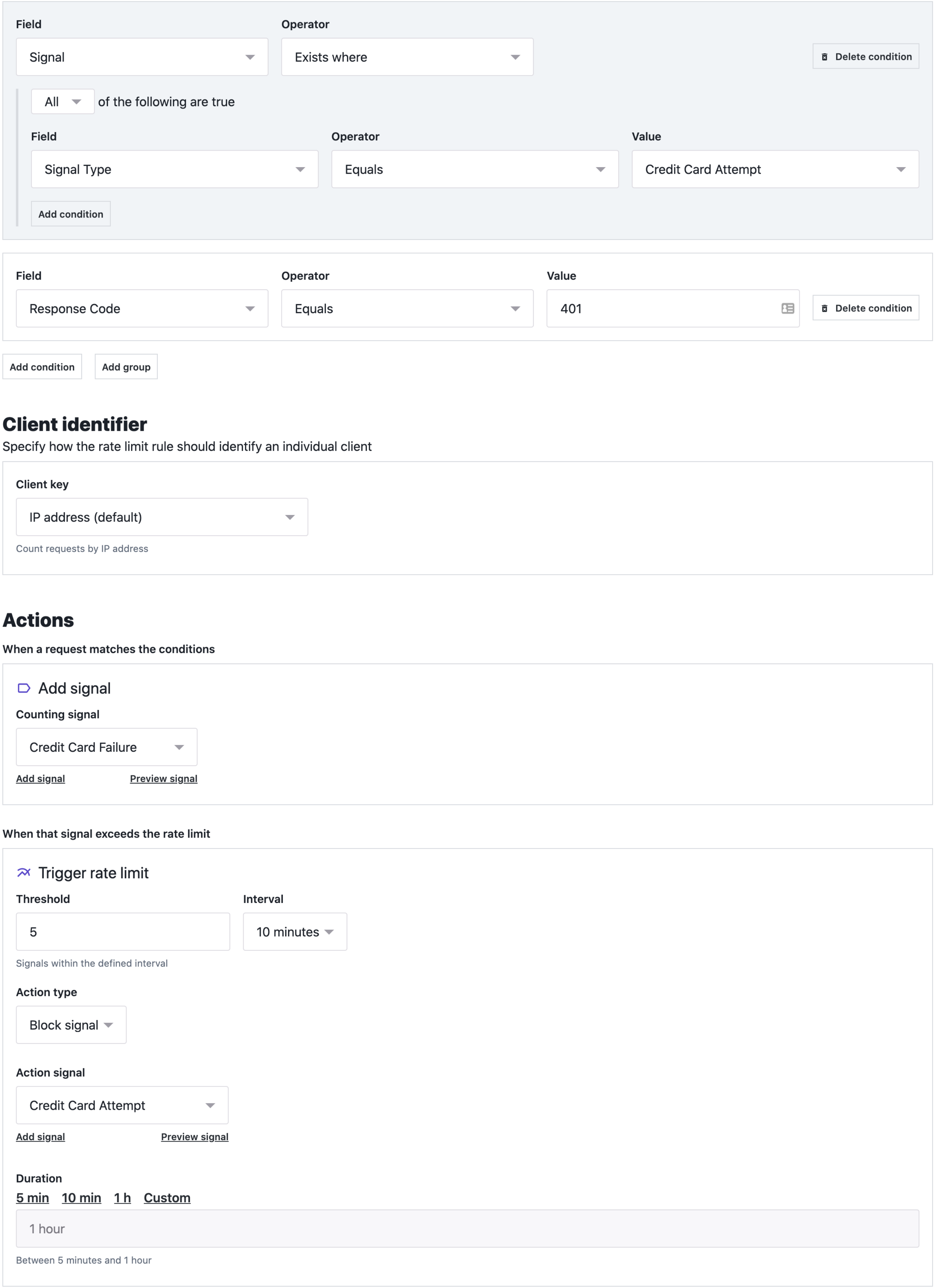
Actions

Action type	Signal
Add signal	Credit Card Attempt

Add signal Preview signal

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 **counting 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**) for the next hour (the **duration**).



Field	Operator	
Signal	Exists where	

All of the following are true

Field	Operator	Value
Signal Type	Equals	Credit Card Attempt

Field	Operator	Value
Response Code	Equals	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

When a request matches the conditions

Add signal

Counting signal

Credit Card Failure

When that signal exceeds the rate limit

Trigger rate limit

Threshold	Interval
5	10 minutes

Signals within the defined interval

Action type

Block signal

Action signal

Credit Card Attempt

Duration

5 min 10 min 1 h Custom

1 hour

Between 5 minutes and 1 hour

Which requests are blocked when a client is rate limited is determined solely by whether or not the **action signal** is present. This means that, after a client has been rate limited, any requests tagged with that signal by request rules will also be blocked if the rate limit rule **action** is set to block.

Other rate limit rule limitations

- A given signal can only be used as the counting signal for a single rate limit rule. A signal can't be used as the counting signal in more than one rate limit rule.
- A site can only have up to 5 rate limit rules using client identifiers other than IP address. For example, if you create 5 rate limit rules that use cookie value as the client identifier, all subsequent new rate limit rules on that site can only use IP address as the client identifier.

Rate limit 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

Kubernetes Reverse Proxy

Introduction

In this example, the Signal Sciences agent runs in a Docker sidecar and proxies all incoming requests for inspection before sending 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 just 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.

Getting and Updating the Signal Sciences Agent Container Image

The official `signalsciences/sigsci-agent` container image available from the Signal Sciences account on Docker Hub is the recommended place to get the image. If you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

The documentation references 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 this case you should specify a 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

keep some consistency, 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, then just replace `latest` with the agent version. You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
   image: signalsciences/sigsci-agent:4.1.0
   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 the `latest`), apply a custom tag, then use that custom tag in the configuration. You will want to specify `imagePullPolicy: Never` so that local images are only updated manually. 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 (_) (e.g., 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 (ID and secret key)
- A volume to write temporary files

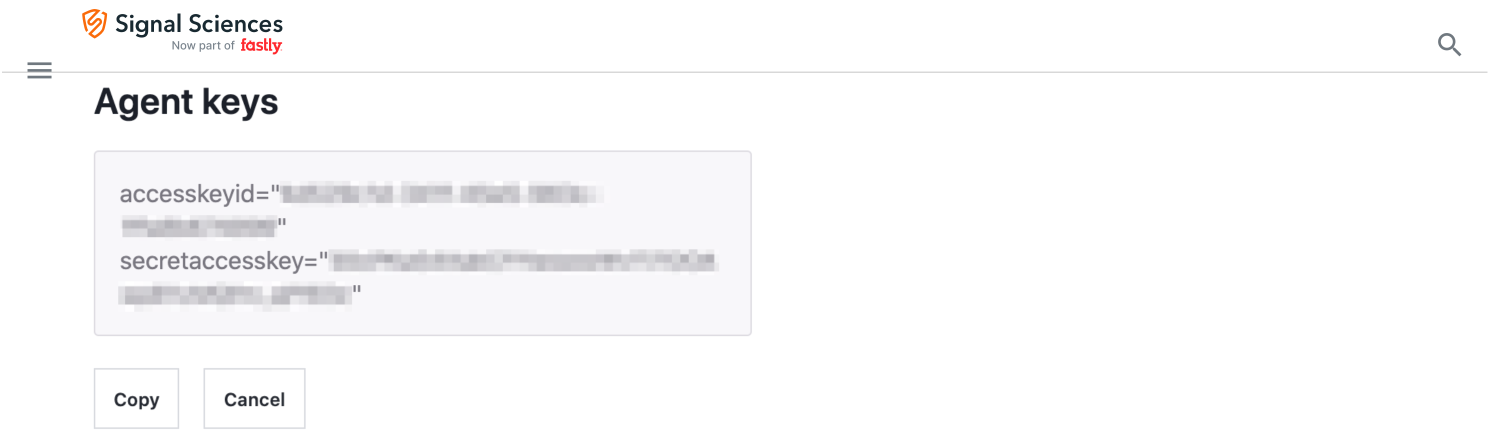
Agent Credentials

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

- SIGSCI_ACCESSKEYID: Identifies the site that the agent is configured against
- SIGSCI_SECRETACCESSKEY: The shared secret key to authenticate and authorize the agent

The credentials can be found by following these steps:

1. Log into the Signal Sciences console.
2. Click on **Agents**. The Agents page appears.



Agent keys

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

Because of the sensitive nature of these values, it is recommended to use the builtin `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded the 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.

Using secrets via environment variables is done using the valueFrom option instead of the value option such as follows:

```
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 documentation 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 with something like the following YAML:

```
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:

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

See the documentation on secrets for more details.

Agent Temporary Volume

For added security, it is recommended that the `sigsci-agent` container be executed with the root filesystem mounted read only. The agent, however, 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. Typically this is just configured in the `volumes` section of a deployment.

```
volumes:
 - name: sigsci-tmp
   emptyDir: {}
 - 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 Science Agent as a Reverse Proxy in Front of a Web Application without the Signal Sciences Module

If the web application does not support a Signal Science Module (or installing a module is not desired), then the `sigsci-agent` container can be configured to run as a reverse proxy in front of the web application in the same pod.

Changing the Application Port and Replacing it with the Signal Sciences Agent

To configure Signal Sciences with this deployment type you must:

- Change the port in which the web application listens (e.g., from 8000 to 8001 or similar)
- Add the sigsci-agent container to the pod, configured in reverse proxy mode to listen on the original web application port and proxy requests to the new web application listener port
- Add an `emptyDir{}` volume as a place for the `sigsci-agent` to write temporary data

The following set of changes reconfigures the web application originally using port 8000 to use an alternate port of 8001 adding the `sigsci-agent` as a reverse proxy listening on a the original web application port 8000 with an upstream of the new web application port 8000.

Change the Application Port to an Alternate Port

Change the application configuration (in this case the first argument) and the `containerPort` to an alternate port (was 8000):

```
...
      containers:
      # Example helloworld app running on port 8001 without sigsci configured
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
        - localhost:8001
        ports:
        - containerPort: 8001
```

Add the Signal Sciences Agent as a Reverse Proxy

```
...
      containers:
      # Example helloworld app running on port 8001 without sigsci configured
      - name: helloworld
        image: signalsciences/example-helloworld:latest
        imagePullPolicy: IfNotPresent
        args:
        - localhost:8001
        ports:
        - containerPort: 8001
      # 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:
```

```
          valueFrom:
            secretKeyRef:
              name: sigsci.my-site-name-here
              key: secretaccesskey
        # Configure the revproxy listener to listen on the original web application port 8000
        # forwarding to the app on the alternate port 8001 as the upstream
        - name: SIGSCI_REVPROXY_LISTENER
          value: "http:{listener='http://127.0.0.1:8000',upstreams='http://127.0.0.1:8001',access-log='/dev/stdou
        ports:
        - containerPort: 8000
        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
```

Adding the Signal Sciences Agent Temp Volume Definition to the Deployment

Finally, the agent temp volume needs to be defined for use by the other containers in the pod. This just 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

Alternatively, if the application listener should not (or cannot) be reconfigured in the pod, modify the Kubernetes service to point to the listener port exposed by the `sigsci-agent` reverse proxy instead of directly to the web application. The `sigsci-agent` can then be configured to proxy to the application port inside the pod.

The following set of changes adds the `sigsci-agent` as a reverse proxy listening on a new port 8001 with an upstream of the application port on 8000 and changes the service to point to the reverse proxy on port 8001 instead of directly to the application on port 8000:

Add the Signal Sciences Agent as a Reverse Proxy to Proxy to the Application Port

```
...
      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:
```

```
          valueFrom:
            secretKeyRef:
              name: sigsci.my-site-name-here
              key: secretaccesskey
        # Configure the revproxy listener to listen on the new service port 8001
        # forwarding to the app on 8000 as the upstream
        - name: SIGSCI_REVPROXY_LISTENER
          value: "http:{listener='http://127.0.0.1:8001',upstreams='http://127.0.0.1:8000',access-log='/dev/stdou
        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
```

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
```

> **Note:** The above modification assumes that `sigsci` secrets were added to the system. Adding the Signal Sciences Agent Temp Volume Definition to the Deployment

Finally, the agent temp volume needs to be defined for use by the other containers in the pod. This just 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: {}
```

Ubuntu NGINX 1.10-1.14

Add the Package Repositories

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Ubuntu 20.04 "focal"

```
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”

Cut-and-paste the following script into a terminal:

```
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”

Cut-and-paste the following script into a terminal:

```
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”

Cut-and-paste 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”

Cut-and-paste 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
```

Enabling Lua for NGINX

For older versions of NGINX, we require NGINX to be built with Lua and LuaJIT support. It is recommended to first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

Nginx.org distribution

1. The first step is to install the dynamic Lua NGINX Module appropriate for your NGINX distribution:

Ubuntu distribution

Enable Lua by installing the `nginx-extras` package with the following command:

Check that Lua is loaded correctly

To verify that Lua has been loaded properly load the following config(ex: `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:
```

```
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 successfully loading the 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 and Configure the Signal Sciences NGINX Module

1. Install the module

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

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

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

3. Restart the NGINX Service to initialize the new module

   Ubuntu 14.04 and lower

```
sudo systemctl restart nginx
```

HAProxy Module Install

Requirements

- HAProxy 1.7 or higher
- Lua module enabled on host

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

Installation

Agent configuration changes

> **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.

After installing the Signal Sciences agent, you will need to create the directory structure for the Unix domain socket under chroot:

```
sudo mkdir -p /haproxy-chroot-directory/var/run/
```

Then, add the following line to your agent configuration file (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

Installation with Package Manager

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

OS	Command
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`

Manual Installation

Alternatively, the HAProxy module can also 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. After downloading the module `.tar.gz` archive, create the directory it 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

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

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

HAProxy 1.9+

In addition to the HAProxy configuration file edits above, if you are running HAProxy 1.9 or higher, 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

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"
```

Upgrading

To upgrade the HAProxy module, you will need to download and install the latest version of the module.

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

Extracting Your Data

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

Limits for concurrent connections to this endpoint:

- **Two** per site
- **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

# Initial setup
api_host = 'https://dashboard.signalsciences.net'
email = os.environ.get('SIGSCI_EMAIL')
password = os.environ.get('SIGSCI_PASSWORD')
corp_name = 'testcorp'
site_name = 'www.example.com'

# 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)
```

```
# Authenticate
auth = requests.post(
    api_host + '/api/v0/auth',
    data = {"email": email, "password": password}
)

if auth.status_code == 401:
    print 'Invalid login.'
    sys.exit()
elif auth.status_code != 200:
    print 'Unexpected status: %s response: %s' % (auth.status_code, auth.text)
    sys.exit()

parsed_response = auth.json()
token = parsed_response['token']

# Loop across all the data and output it in one big JSON object
headers = {
    'Content-type': 'application/json',
    'Authorization': 'Bearer %s' % token
}
url = api_host + ('/api/v0/corps/%s/sites/%s/feed/requests?from=%s&until=%s' % (corp_name, site_name, from_time, 
first = True

print '{ "data": ['

while True:
    response_raw = requests.get(url, headers=headers)
    response = json.loads(response_raw.text)

    for request in response['data']:
        data = json.dumps(request)
        if first:
            first = False
        else:
            data = ',\n' + data
        sys.stdout.write(data)

    next_url = response['next']['uri']
    if next_url == '':
        break
    url = api_host + next_url

print '\n] }'
```

Red Hat Agent Installation

Step 1 - Add the Package Repositories

> **Note:** If you are installing a Red Hat Agent older than 4.4.0, set `gpgcheck=0` in the following scripts.

Red Hat CentOS 8

Cut-and-paste 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
```

```
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

Cut-and-paste 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.

Cut-and-paste 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
```

Step 2 - Install the Signal Sciences Agent Package

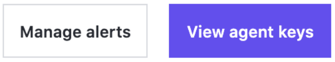
1. To install the package, running the following command.

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

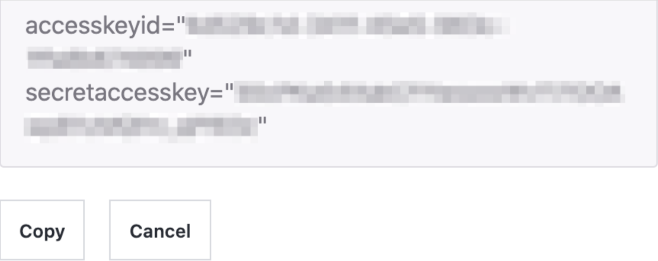
2. Create the file `/etc/sigsci/agent.conf`

3. Configure the agent by inputting the Agent Access Key and Agent Secret Key into the `/etc/sigsci/agent.conf`.

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

   - The Agent Access Key and Agent Secret Key will be visible within the window:



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

Example /etc/sigsci/agent.conf

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

Additional configuration options are listed on the agent configuration page.

4. Start the Signal Sciences Agent

   RHEL 7/CENTOS 7 or higher

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

   RHEL 6/CENTOS 6

   ```
   start sigsci-agent
   ```

Next Steps

Install the Signal Sciences Module:

- Explore module options

Red Hat Apache Module Install

Red Hat CentOS 8 / RHEL 8

1. First install the Signal Sciences Apache Module using `yum`.

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

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the “Dynamic Shared Object (DSO) Support” section:

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

3. Restart Apache httpd.

   ```
   sudo systemctl restart httpd
   ```

Red Hat CentOS 7 / RHEL 7

1. First install the Signal Sciences Apache Module using `yum`.

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

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the “Dynamic Shared Object (DSO) Support” section:

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

3. Restart Apache httpd.

   ```
   sudo systemctl restart httpd
   ```

Red Hat CentOS 6 / RHEL 6

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

Apache 2.4 Install Command for 64-bit module:

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

Apache 2.2 Install Command for 32-bit module:

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

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the “Dynamic Shared Object (DSO) Support” section:

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

3. Restart Apache httpd.

```
sudo service httpd restart
```

Next Steps

- Verify Agent and Module Installation

Explore other installation options:

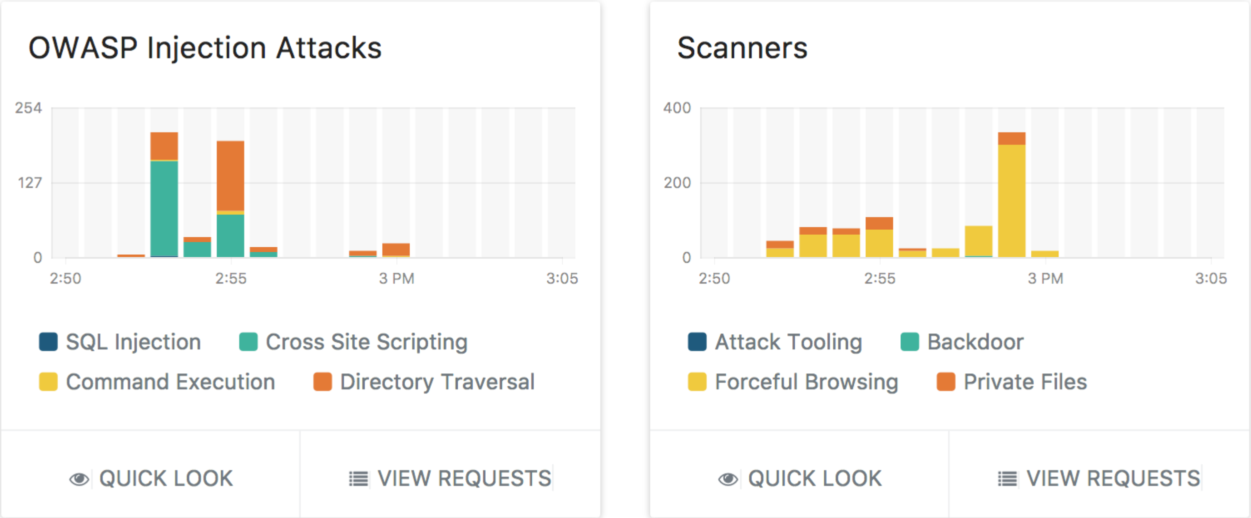
- Explore module options

Investigating An Attack

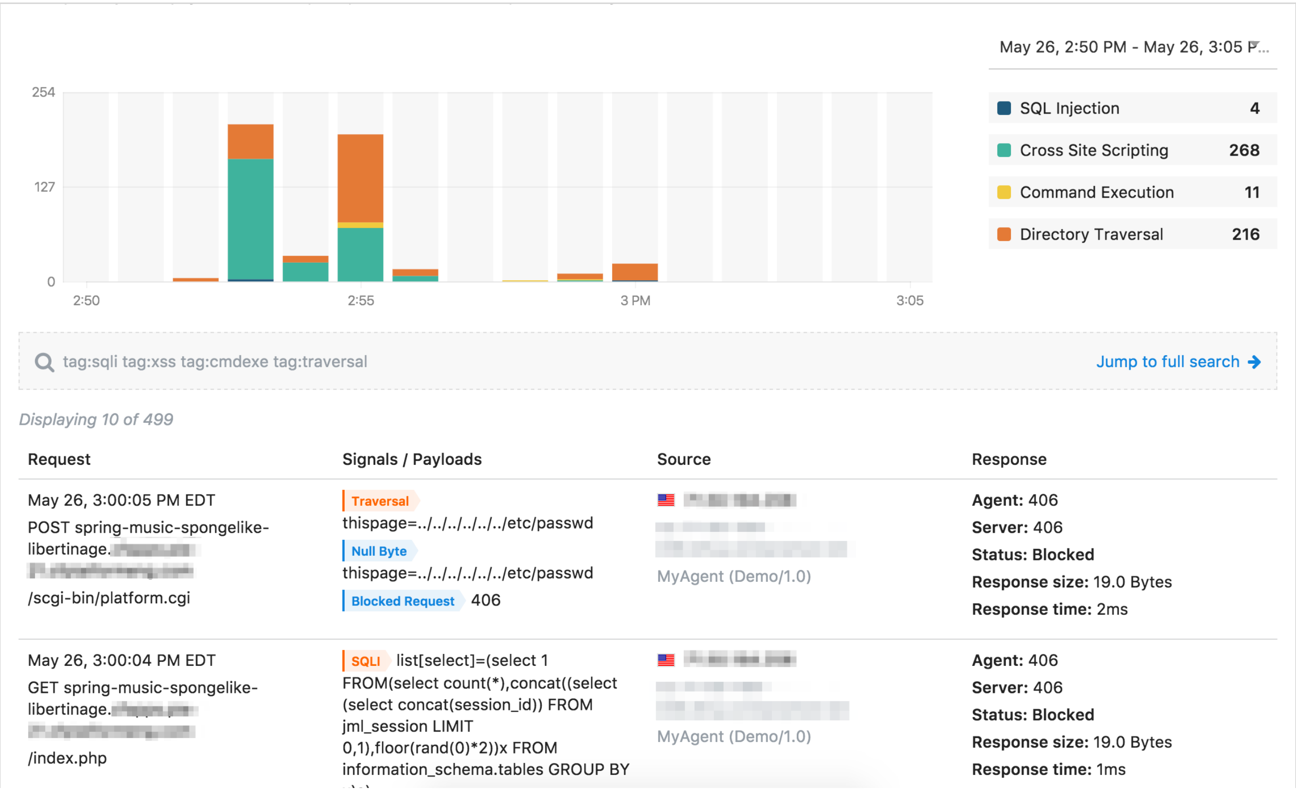
Now that you’ve run attack tooling against your site, you can start to explore the data available in Signal Sciences:

Using the Attack and Anomaly Panels

1. The attack and anomaly panels on the Overview page show the signals we’ve identified over time.
   - You can zoom into a particular date range by clicking and dragging on the chart. Your time selection will be carried through as you drill down into your data.



2. At the bottom of each panel there are **Quick Look** and **View Requests** buttons. Clicking on the **Quick Look** button will display a summary view of the data in the graph.



May 26, 2:50 PM - May 26, 3:05 P...

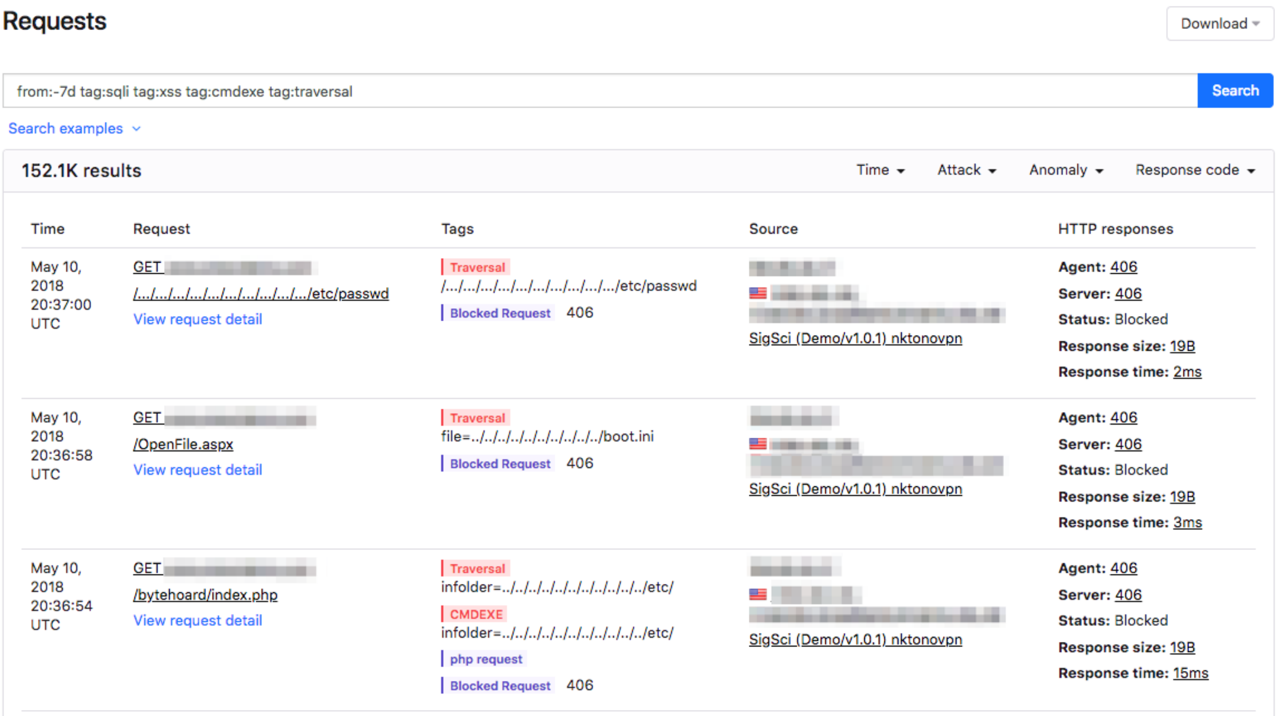
SQL Injection	4
Cross Site Scripting	268
Command Execution	11
Directory Traversal	216

tag:sqli tag:xss tag:cmdexe tag:traversal   Jump to full search

Displaying 10 of 499

Request	Signals / Payloads	Source	Response
May 26, 3:00:05 PM EDT POST spring-music-spongelike-libertinage.[redacted] /scgi-bin/platform.cgi	Traversal thispage=../../../../../../etc/passwd Null Byte thispage=../../../../../../etc/passwd Blocked Request 406	[redacted] MyAgent (Demo/1.0)	**Agent:** 406 **Server:** 406 **Status: Blocked** **Response size:** 19.0 Bytes **Response time:** 2ms
May 26, 3:00:04 PM EDT GET spring-music-spongelike-libertinage.[redacted] /index.php	SQLI list[select]=(select 1 FROM(select count(*),concat((select (select concat(session_id)) FROM jml_session LIMIT 0,1),floor(rand(0)*2))x FROM information_schema.tables GROUP BY	[redacted] MyAgent (Demo/1.0)	**Agent:** 406 **Server:** 406 **Status: Blocked** **Response size:** 19.0 Bytes **Response time:** 1ms

3. Clicking on the **View Requests** button will take you to the search page with the data from the graph already filtered. The search page shows individual requests that contain attack or anomaly data. In addition to general metadata (HTTP request, hostname, response code, response size, etc.), we display the specific attacks and anomalies under the "Signals/Payloads" column.



Requests

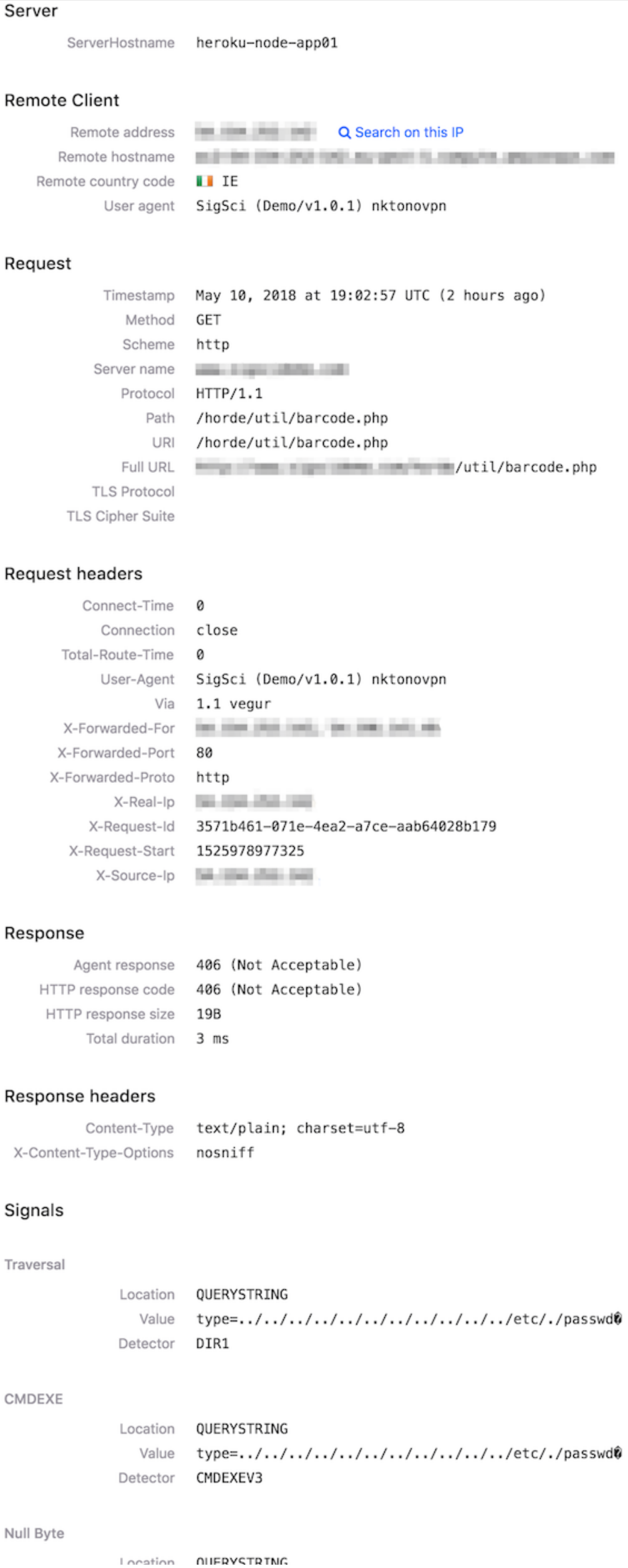
from:-7d tag:sqli tag:xss tag:cmdexe tag:traversal

152.1K results

Time	Request	Tags	Source	HTTP responses
May 10, 2018 20:37:00 UTC	GET [redacted] /../../../../../../../../../../../etc/passwd View request detail	Traversal /../../../../../../../../../../../etc/passwd Blocked Request 406	[redacted] SigSci (Demo/v1.0.1) nktonovpn	**Agent:** 406 **Server:** 406 **Status:** Blocked **Response size:** 19B **Response time:** 2ms
May 10, 2018 20:36:58 UTC	GET [redacted] /OpenFile.aspx View request detail	Traversal file=../../../../../../../../../../boot.ini Blocked Request 406	[redacted] SigSci (Demo/v1.0.1) nktonovpn	**Agent:** 406 **Server:** 406 **Status:** Blocked **Response size:** 19B **Response time:** 3ms
May 10, 2018 20:36:54 UTC	GET [redacted] /bytehoard/index.php View request detail	Traversal infolder=../../../../../../../../../../../etc/ CMDEXE infolder=../../../../../../../../../../../etc/ php request Blocked Request 406	[redacted] SigSci (Demo/v1.0.1) nktonovpn	**Agent:** 406 **Server:** 406 **Status:** Blocked **Response size:** 19B **Response time:** 15ms

- You can filter by any value by clicking on any of the signals or links. For example, clicking on the source IP will constrain the results to all requests by that IP.
- To view full request details, click **View request detail**.

4. The request details page lists all of the metadata we've 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.



Server

ServerHostname heroku-node-app01

Remote Client

Remote address [redacted] Search on this IP
Remote hostname [redacted]
Remote country code 🇮🇪 IE
User agent SigSci (Demo/v1.0.1) nktonovpn

Request

Timestamp May 10, 2018 at 19:02:57 UTC (2 hours ago)
Method GET
Scheme http
Server name [redacted]
Protocol HTTP/1.1
Path /horde/util/barcode.php
URI /horde/util/barcode.php
Full URL [redacted]/util/barcode.php
TLS Protocol
TLS Cipher Suite

Request headers

Connect-Time 0
Connection close
Total-Route-Time 0
User-Agent SigSci (Demo/v1.0.1) nktonovpn
Via 1.1 vegur
X-Forwarded-For [redacted]
X-Forwarded-Port 80
X-Forwarded-Proto http
X-Real-Ip [redacted]
X-Request-Id 3571b461-071e-4ea2-a7ce-aab64028b179
X-Request-Start 1525978977325
X-Source-Ip [redacted]

Response

Agent response 406 (Not Acceptable)
HTTP response code 406 (Not Acceptable)
HTTP response size 19B
Total duration 3 ms

Response headers

Content-Type text/plain; charset=utf-8
X-Content-Type-Options nosniff

Signals

Traversal

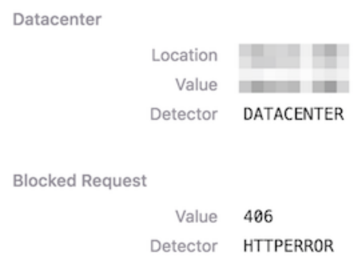
Location QUERYSTRING
Value type=../../../../../../../../../../../../etc/./passwd�
Detector DIR1

CMDEXE

Location QUERYSTRING
Value type=../../../../../../../../../../../../etc/./passwd�
Detector CMDEXEV3

Null Byte

Location QUERYSTRING



Datacenter

Location [redacted]
Value [redacted]
Detector DATACENTER

Blocked Request

Value 406
Detector HTTPERROR

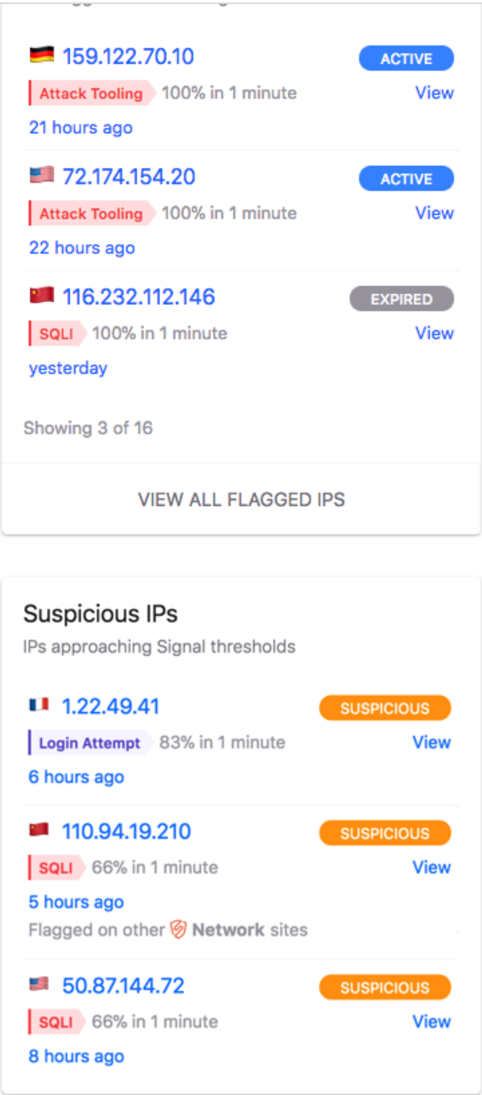
Note: Because we only send over the parts of a request that we consider anomalous and redact sensitive data, you may need additional context to fully investigate an attack or anomaly. To address this use case, we recommend using a header link to add a link to your internal systems on the request details page via a linking identifier (e.g., an X-Request-Id response header).

Using the Flagged and Suspicious IPs Lists

1. The Events and Suspicious IPs lists on the Overview page list IP addresses that are the origin of requests containing attack payloads.

   Suspicious IPs represent IP addresses from which requests containing attack payloads have originated, but the volume of attack traffic from that IP address has not exceeded the decision threshold. 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).

- If a suspicious IP has been detected as malicious and flagged by other sites on the Signal Sciences network, there will be an indicator stating "Flagged on other Signal Sciences Network sites".
- If a flagged IP is listed as "Active", it is currently being blocked (if the agent mode is set to "blocking") or logged (if set to "not blocking").
- If a flagged IP is listed as "Expired", then the event has ended and requests from that IP address will no longer be blocked or logged.



159.122.70.10 ACTIVE
Attack Tooling 100% in 1 minute View
21 hours ago

72.174.154.20 ACTIVE
Attack Tooling 100% in 1 minute View
22 hours ago

116.232.112.146 EXPIRED
SQLI 100% in 1 minute View
yesterday

Showing 3 of 16

VIEW ALL FLAGGED IPS

Suspicious IPs

IPs approaching Signal thresholds

1.22.49.41 SUSPICIOUS
Login Attempt 83% in 1 minute View
6 hours ago

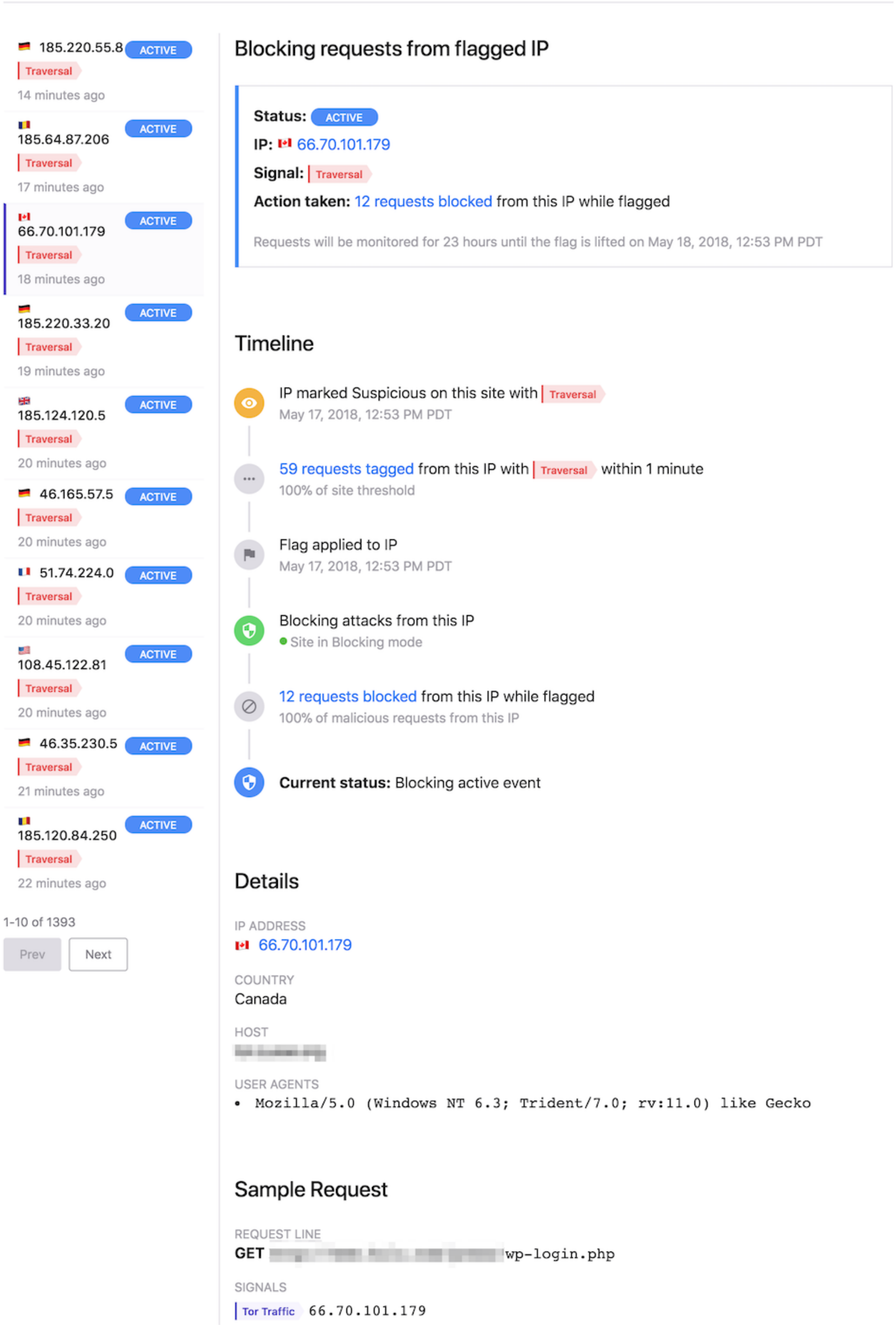
110.94.19.210 SUSPICIOUS
SQLI 66% in 1 minute View
5 hours ago
Flagged on other Network sites

50.87.144.72 SUSPICIOUS
SQLI 66% in 1 minute View
8 hours ago

2. Clicking directly on the IP address will take you to the search page displaying all requests from that IP address.

3. Clicking on **View** will take you to the Events page for that IP address. This page provides detailed information about the event associated with this IP address, including:

- The signal assigned to the event.
- A timeline of what transpired during this event.
- Additional details about the event.



185.220.55.8 ACTIVE
Traversal
14 minutes ago

185.64.87.206 ACTIVE
Traversal
17 minutes ago

66.70.101.179 ACTIVE
Traversal
18 minutes ago

185.220.33.20 ACTIVE
Traversal
19 minutes ago

185.124.120.5 ACTIVE
Traversal
20 minutes ago

46.165.57.5 ACTIVE
Traversal
20 minutes ago

51.74.224.0 ACTIVE
Traversal
20 minutes ago

108.45.122.81 ACTIVE
Traversal
20 minutes ago

46.35.230.5 ACTIVE
Traversal
21 minutes ago

185.120.84.250 ACTIVE
Traversal
22 minutes ago

1-10 of 1393

Prev Next

Blocking requests from flagged IP

Status: ACTIVE
IP: 66.70.101.179
Signal: Traversal
Action taken: 12 requests blocked from this IP while flagged

Requests will be monitored for 23 hours until the flag is lifted on May 18, 2018, 12:53 PM PDT

Timeline

IP marked Suspicious on this site with Traversal
May 17, 2018, 12:53 PM PDT

59 requests tagged from this IP with Traversal within 1 minute
100% of site threshold

Flag applied to IP
May 17, 2018, 12:53 PM PDT

Blocking attacks from this IP
● Site in Blocking mode

12 requests blocked from this IP while flagged
100% of malicious requests from this IP

Current status: Blocking active event

Details

IP ADDRESS
66.70.101.179

COUNTRY
Canada

HOST
[redacted]

USER AGENTS
- `Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko`

Sample Request

REQUEST LINE
GET [redacted]`wp-login.php`

SIGNALS
Tor Traffic `66.70.101.179`

4. The timeline illustrates the actions that occurred during the event. This includes when the IP address was identified as suspicious, how many requests were received from the IP before it was flagged, when the IP was flagged, and how many requests were blocked or logged accordingly.

5. The “Details” section provides additional, detailed information regarding the event. Depending on the nature of the attack, this can include the host, user agents, file paths, and country of origin.

6. The “Sample Request” highlights a single request received during the event, including the request itself and the signals applied to it. Clicking on **View this request** will take you to the request details page for that request.

Now that you know how to investigate and drill down into the data captured by Signal Sciences, learn how to [test blocking mode](/using-signal-sciences/walkthrough/testing-blocking-mode).

Generic Webhooks

Our generic webhooks integration allows you to subscribe to notifications for certain activity on Signal Sciences.

Adding a webhook

1. Go to **Manage** > **Site Integrations**.
2. Click **Add site integration** and select the **Generic Webhook** integration.
3. Paste in a URL to receive the notifications and choose which activity you want to trigger the webhook.
4. Choose whether to receive notifications for all activity or specific activity.
5. Click **Add**.

Notifications format

Notifications are sent with the following format:

```
{
  "created": "2014-12-09T10:43:54-08:00",
  "type": "flag",
  "payload": ...
}
```

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.

The key can be rotated by clicking the **Edit** button next to the generic webhook and then **Rotate key** in the “Generic webhook integration” form.

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

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)
        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)

print(ok)
```

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="$2"
  if [ "$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 was flagged	Get event by ID
expireFlag	An IP flag was manually expired	List events
createCustomRedaction	A custom redaction was created	Create a custom redactions
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	

NGINX

NGINX Module Release Notes

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

- Fixed backward compatibility issue

1.2.8 2019-06-10

- 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 env 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

- 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

1.1.3 2016-07-26

- Corrected version number reported by module

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 https://github.com/fperrad/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

- 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 (whitelisted, e.g. “GET”, “POST”). The logic is now inverted to allow all methods not on an ignored list (blacklisted, 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+363 2015-08-24

- Fixed issue of missing server response codes introduced by 361

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 (https://github.com/fperrad/lua-MessagePack) 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

- 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

---

Blocking

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

What is a decision?

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 is sent to the Signal Sciences backend (see the Data Redactions FAQ to learn how this is done in a safe and private manner). The backend aggregates attacks from across all of your agents, and when enough attacks are seen from a single IP, the backend reaches a decision to flag that IP. Agents will pull those 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.

How do I trust the decisions you make?

Our console provides transparency about which IP we flagged, when and why we flagged it, and what action we took (log or block, depending on which mode you’re in).

What is the difference between “blocking” and “not blocking”?

When an IP address is flagged, “blocking” mode takes action by automatically blocking subsequent requests containing attacks for 24 hours after the decision has been reached by the backend. Because “blocking” mode only blocks requests containing attacks, legitimate traffic is still allowed through. Attacks are blocked by returning a unique `HTTP 406` response code. By using the unique `406` response code—as opposed to a `404` or `500`—your operations team won’t get paged thinking there’s an outage or issue with your application.

Agents can also be set to “not blocking” mode. In “not blocking” mode, charts in the console and decisions on flagged IPs appear in the event list and alert notifications to provide visibility into all attacks. Once a decision has been reached, subsequent attacks from flagged IP addresses are only logged, not blocked. Additionally, requests will not be blocked by any custom rules you have created to immediately block requests. If those rules are designed to also tag requests for visibility, requests will continue to be tagged.

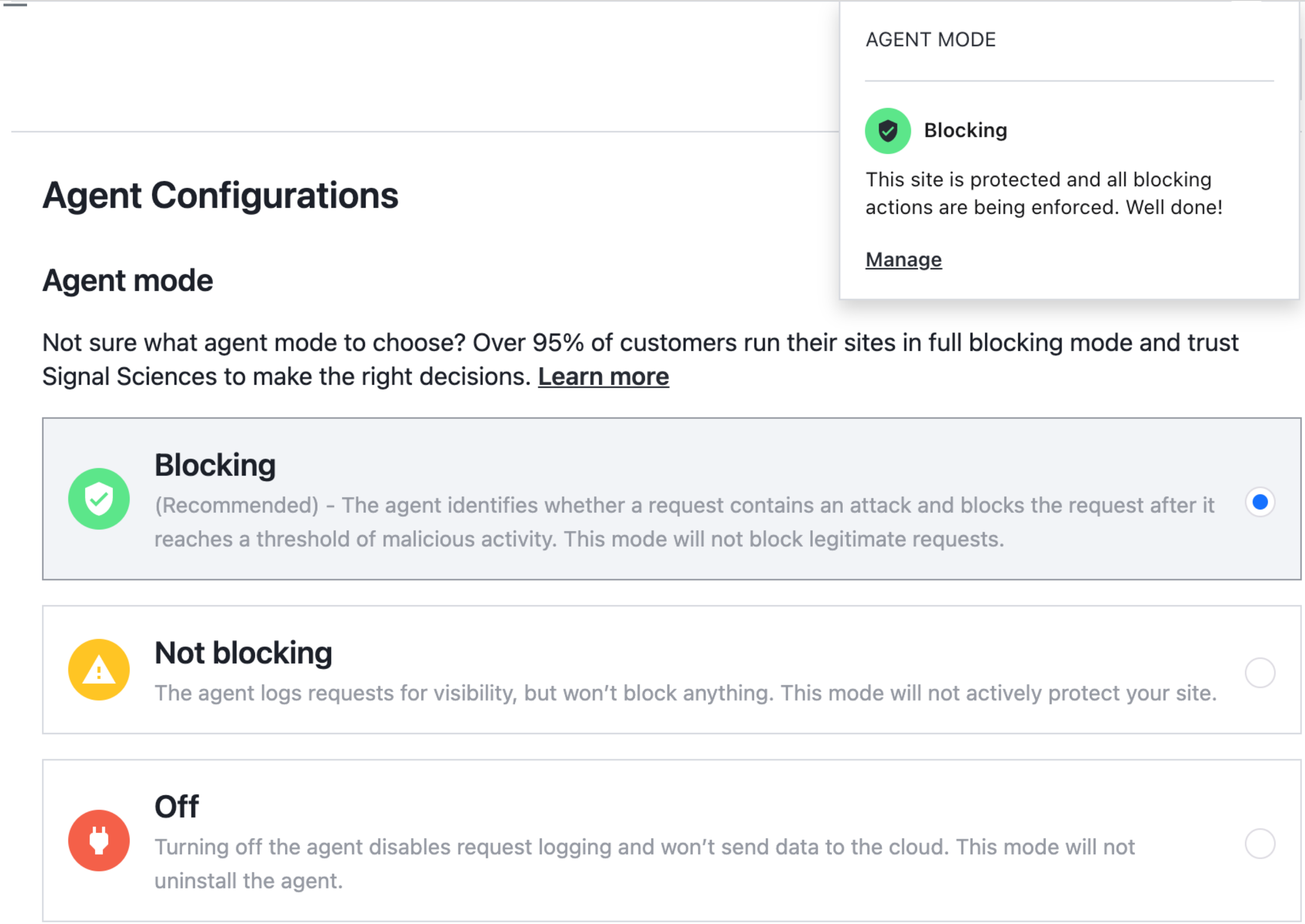
Why would I want to use blocking mode?

You can see the decisions we reach while the agent mode is set to “not blocking”, so you’ll feel comfortable with how we’re identifying attacks before you switch to “blocking”. Additionally, since “blocking” still allows legitimate traffic through (i.e. requests that don’t contain attacks), running in blocking mode doesn’t negatively impact your application.

How do I change agent modes?

To switch agent modes, click on the agent mode in the top navigation and then click on **Manage**. Then select **Blocking**, **Not blocking**, or **Off**.

Owners can change the agent mode for all sites, while Admins and Users can change the agent mode for any sites they are member of. See Corp Management for more information.



AGENT MODE

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. ○

What are the IP address flagging thresholds?

As requests with attack signals are sent to our backend, we track the number of signals that are seen from an IP across all agents.

When the number of malicious requests from an IP reaches one of the following thresholds, the IP will be flagged and subsequent malicious requests will be blocked (or logged if the agent mode is set to "not blocking") for 24 hours:

Interval	Threshold	Frequency of Check
1 minute	50	Every 20 seconds
10 minutes	350	Every 3 minutes
1 hour	1,800	Every 20 minutes

Note: Requests containing only anomaly signals are not counted towards IP flagging thresholds.

How are block rules different than blocking mode?

Block rules block all requests from a given IP address. Block rules are never created automatically by Signal Sciences; all blocking rules are created by the customers themselves.

What are allow rules?

Allow rules give you the ability to allow all requests from certain IP ranges or individual parameters, so they won't show up in the console or affect decisions. Typical use cases are allowlisting an IP range used for scanning, or parameters that might resemble attacks but are actually valid inputs in the application.

What is the precedence of allow and block rules?

When two conflicting rules are created, the allow rules will always take precedence over the block rules. 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.

How do I change the default block duration for flagged IPs?

How do I configure the blocking mute period?

In some cases you may want to disable blocking during a specific time period to accommodate scheduled vulnerability scans of your applications. There are two ways to achieve this.

First, blocking mode can be disabled via our API. Scan automation scripts can include a call to the API to disable blocking mode before scheduled scans start.

Second, if scanner IP addresses are known then these IP addresses can be allowlisted by creating rules to allow them in the console.

How do I configure the time to lift IP flag?

By default a flagged IP will be removed from the flagged IP list in 24 hours. This time period can be configured via our API by setting the `blockDurationSeconds` value when calling the update site by name endpoint.

What if I have a field that looks like SQL? How can I ensure it's not blocked?

You can create signal exclusions to exclude requests matching your parameters from being tagged with certain signals.

Are flagged IPs tracked between customers?

Whenever an IP is flagged by any Signal Sciences customer, we record that IP address as a known potential bad actor and make its status known across our whole network. If that same IP is seen on another customer's workspace, we indicate that it's been identified as a potential threat by tagging it with the `SigSci IP` signal.

What happens when I see false positives?

These are very rare in practice, but we take them seriously. File a support ticket immediately. We can address these quickly on our end, and you won't have to update the agent to see the changes take effect.

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.

To enable or disable two-factor authentication, go to **Profile** > **Account Settings**. There, click **Enable** or **Disable** under "Two-Factor Authentication" and follow the instructions.

> **Note:** Two-factor authentication settings are set at the user-level for a particular corporation. This means that a user only needs to configure two-factor authentication once to access the sites to which they belong.

Error Response Codes

What do "-2", "-1", and "0" agent response codes mean?

The `-2`, `-1`, and `0` 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/response couldn't be matched to the request
- The module timed out waiting for a response from the agent

Request and response mismatch

Error response codes can occur when a post/response couldn't be matched to any actual requests. This is typically the result of NGINX redirecting before the request is passed to the Signal Sciences module.

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)

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 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 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

Metrics for each agent can be viewed by going to **Agents** > **Metrics**. There are a few metrics that can indicate error response codes are occurring due to the module timing out.

Connections dropped

The “Connections dropped” metric indicates number of requests that were allowed through (or “dropped”):

CPU usage

The CPU metrics can indicate the host is overloaded, preventing it from processing requests quickly enough.

- The “Host CPU” metric indicates the CPU percentage for all cores together (100% is maximum).
- 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.

Further help

If you're unable to troubleshoot or resolve this issue yourself, generate an agent diagnostic package by running `sigsci-agent-diag`, which will output a .tar.gz archive with diagnostic information. Reach out to our support team to explain the issue in detail—including console links to the requests and agents affected—and provide the diagnostic .tar.gz archive.

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.

Installation

Download

Download manually

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

Access with Maven

For Java projects using Maven for build or deployment, the Signal Sciences Java modules can be installed by adding the following to the

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

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 just 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.

Getting and Updating the Signal Sciences Agent Container Image

The official `signalsciences/sigsci-agent` container image available from the Signal Sciences account on Docker Hub is the recommended place to get the image. If you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

The documentation references 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 this case you should specify a 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 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. To keep some consistency, 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, then just replace `latest` with the agent version. You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
   image: signalsciences/sigsci-agent:4.1.0
   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 the `latest`), apply a custom tag, then use that custom tag in the configuration. You will want to specify `imagePullPolicy: Never` so that local images are only updated manually. 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:

```
...
```

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 (_) (e.g., 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 (ID and secret key)
- A volume to write temporary files

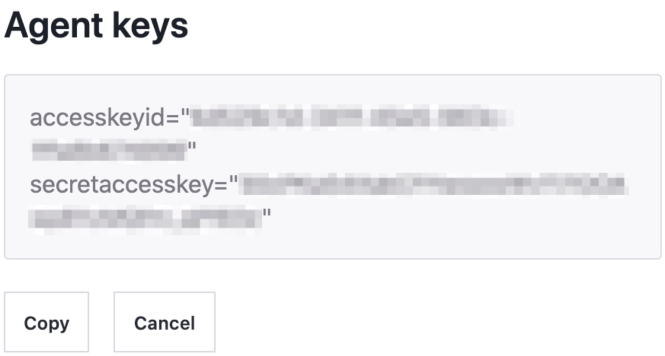
Agent Credentials

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

- SIGSCI_ACCESSKEYID: Identifies the site that the agent is configured against
- SIGSCI_SECRETACCESSKEY: The shared secret key to authenticate and authorize the agent

The credentials can be found by following these steps:

1. Log into the Signal Sciences console.
2. Click on **Agents**. The Agents page appears.
3. On the Agents page click **View Agent Keys**. The agent keys window appears.
4. Copy down the **Access Key** and **Secret Key** for later use.



Agent keys

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

Because of the sensitive nature of these values, it is recommended to use the builtin `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded the 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.

Using secrets via environment variables is done using the valueFrom option instead of the value option such as follows:

```
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 documentation 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 with something like the following YAML:

```
  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:

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

See the documentation on secrets for more details.

Agent Temporary Volume

For added security, it is recommended that the `sigsci-agent` container be executed with the root filesystem mounted read only. The agent, however, 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. Typically this is just configured in the `volumes` section of a deployment.

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

Containers would then typically 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 described earlier 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: See the language/framework module installation documentation. Web server based modules must have the Signal Sciences module added to the container: See the web server module installation documentation.

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:
```

```
      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

The `sigsci-agent` container will default to RPC mode with a Unix Domain Socket (UDS) file at `/sigsci/tmp/sigsci.sock`. There should be a temp volume mounted at `/sigsci/tmp` to capture this socket file and should be shared with the pod. The web application should be configured to communicate with the `sigsci-agent` via this UDS socket. The deployment YAML will need to 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 desired (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 (geodata, etc).

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
      #  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

Finally, the agent temp volume needs to be defined for use by the other containers in the pod. This just 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: {}
```

Ubuntu NGINX 1.9 or lower

Add the Package Repositories

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Ubuntu 18.04 "bionic"

Cut-and-paste the following script into a terminal:

```
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"

Cut-and-paste the following script into a terminal:

```
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"

Cut-and-paste 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"

Cut-and-paste 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
```

Enabling Lua for NGINX

To assist customers, we provide pre-built drop in replacements NGINX packages already built with the `ngx_lua` module. This is intended for customers who prefer not to build from source, or who either use a distribution provided package or an official NGNIX provided package.

Flavors of our NGINX replacement packages

We support three “flavors” of NGINX. These flavors are based on what upstream package we’ve based our builds off of. 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 (Ubuntu 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 Matrix of our NGINX replacement packages

The following versions are contained in the various OS and flavor packages:

OS	Distribution	Stable	Mainline
Ubuntu 12.04 (Precise)	1.1.19	1.8.1	1.9.10
Ubuntu 14.04 (Trusty)	1.4.6	1.8.1	1.9.10
Ubuntu 15.04 (Vivid)	1.6.2	1.8.1	1.9.10
Ubuntu 16.04 (Xenial)	1.10.3	N/A	N/A
Ubuntu 18.04 (Bionic)	1.14.0	N/A	N/A

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

To configure the apt repository on your Ubuntu systems:

1. Add our 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 12.04 (Precise)	`deb https://apt.signalsciences.net/nginx/distro precise main`
Ubuntu 14.04 (Trusty)	`deb https://apt.signalsciences.net/nginx/distro trusty main`
Ubuntu 15.04 (Vivid)	`deb https://apt.signalsciences.net/nginx/distro vivid main`

   Stable Flavor

OS	sigsci-nginx.list content
Ubuntu 12.04 (Precise)	`deb https://apt.signalsciences.net/nginx/stable precise main`
Ubuntu 14.04 (Trusty)	`deb https://apt.signalsciences.net/nginx/stable trusty main`
Ubuntu 15.04 (Vivid)	`deb https://apt.signalsciences.net/nginx/stable vivid main`

   Mainline flavor

OS	sigsci-nginx.list content
Ubuntu 12.04 (Precise)	`deb https://apt.signalsciences.net/nginx/mainline precise main`

Ubuntu 15.04 (Vivid)	`deb https://apt.signalsciences.net/nginx/mainline vivid 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 Signal Sciences NGINX

```
sudo apt-get install nginx
```

Check that Lua is loaded correctly

To verify that Lua has been loaded properly load the following config(ex: `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
end

';

}
```

Example of successfully loading the 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 and Configure the Signal Sciences NGINX Module

1. Install the module

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

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

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

3. Restart the NGINX Service to initialize the new module

   Ubuntu 14.04 and lower

   ```
   sudo restart nginx
   ```

   Ubuntu 15.04 and higher

   ```
   sudo systemctl 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.

Installation

Download via package manager

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

OS	Command
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 agent

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

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-ha.sock`) will be inaccessible to the HAProxy module outside of your specified chroot directory.

After installing the Signal Sciences agent, you will need to create the directory structure for the Unix domain socket under chroot:

```
sudo mkdir -p /haproxy-chroot-directory/var/run/
```

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

```
haproxy-spoa-address=unix:/haproxy-chroot-directory/var/run/sigsci-ha.sock
```

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

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

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

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 easily deployed with Heroku. The installation process is compatible with any of the language buildpacks.

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.

Installation

1. Login `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-buildpac
   ```

   > **Note:** The Signal Sciences buildpack must run first, or before your application's primary buildpack.

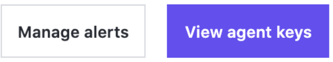
3. Update your `Procfile` file by inserting `sigsci/bin/sigsci-start` so it precedes your existing start command:

   ```
   web: sigsci/bin/sigsci-start <your application's start command>
   ```

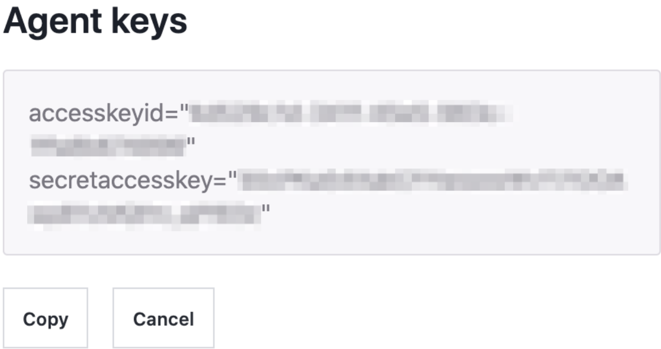
   Example:

```
heroku config:set SIGSCI_ACCESSKEYID=<access key goes here>
heroku config:set SIGSCI_SECRETACCESSKEY=<secret key goes here>
```

- The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:



- The Agent Access Key and Agent Secret Key will be visible within the window:



Agent keys

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

5. Deploy your application, typically with the following commands:

```
git add .
git commit -m "my comment here"
git push heroku master
```

6. You will now see the agent listed in the **Agents** page of the Signal Sciences console.

Additional Configuration Options

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
```

By default the buildpack will install the latest version of the Signal Sciences agent. Which agent version to install can be specified 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

Templated Rules

Templated Rules enable you to gain visibility into registrations, logins, and virtual patches within your application by configuring simple rules.

Enabling and Editing Templated Rules

1. In the Signal Sciences console, go to **Rules** > **Templated Rules** in the navigation bar at the top.

2. Click on the **View** button to the far right of the rule you want to configure.

3. This page features a graph, Event list, and list of requests tagged with the signal associated with this rule.

   Click on **Configure** button in the upper-right corner to enable or edit the rule.

4. You will be taken to a pre-built rule that's ready to set up. You will need to configure the empty value fields with values specific to your application, such as paths, response codes, and headers. It is possible to add and remove conditions in the rule as necessary for your application.

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.

API Protection

> **Note:** API protection signals are not supported on the Essential platform.

With API Protection rules, easily tag 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 Data Storage and Sampling for additional details.

ATO Protection

> **Note:** ATO protection signals are not supported on the Essential platform.

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 Data Storage and Sampling for additional details.

Virtual Patching

With Signal Sciences' virtual patching rules, you have the ability to immediately block or log requests matching specific vulnerabilities. These can be configured to send an alert after a threshold of matching requests.

New virtual patch rules are announced through an optional email subscription. You can subscribe to new virtual patch announcements in your account settings.

Debian Agent Installation

Step 1 - Add the Package Repositories

We’ll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Debian 10 “buster”

Cut-and-paste 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/ buster main
EOF
sudo apt-get update
```

Debian 9 “stretch”

Cut-and-paste 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”

Cut-and-paste 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'
```

Debian 7 “wheezy”

Cut-and-paste 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
```

Step 2 - Install the Signal Sciences Agent Package

1. To install the package, running the following command.

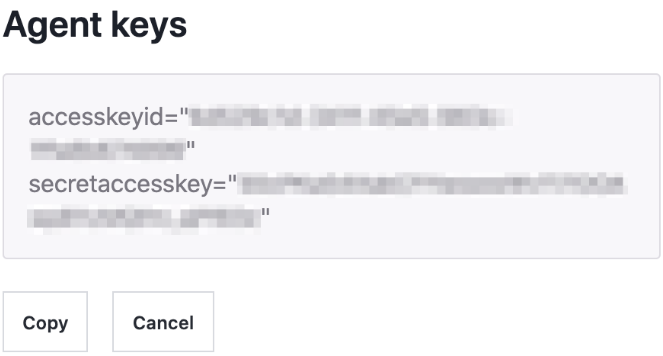
   ```
   sudo apt-get install sigsci-agent
   ```

2. Create the file `/etc/sigsci/agent.conf`

3. Configure the agent by inputting the Agent Access Key and Agent Secret Key into the `/etc/sigsci/agent.conf`.

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

   - The Agent Access Key and Agent Secret Key will be visible within the window:

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

   Example /etc/sigsci/agent.conf

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

   Additional configuration options are listed on the agent configuration page.

4. Start the Signal Sciences Agent

   Debian 8 and higher

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

   Debian 7

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

Next Steps

Install the Signal Sciences Module:

- Explore module options

Debian Apache Module Install

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration file (`apache2.conf` or `httpd.conf`) after the "Dynamic Shared Object (DSO) Support" section:

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

3. Restart the Apache web service.

   ```
   sudo service apache2 restart
   ```

Next Steps

- Verify Agent and Module Installation

Explore other installation options:

- Explore module options

Testing Blocking Mode

Signal Sciences takes a different approach to blocking compared to other products — rather than blocking individual requests that match a particular signature, we look for spikes in 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 for the next 24 hours. This means that requests that don't contain an attack will be allowed, preventing Signal Sciences from breaking normal traffic.

Note, if you completed Scenario 3 from the Testing With Attack Tooling page, you have already verified blocking malicious traffic using an attack tool. To manually verify blocking, complete the two sections below.

Verifying your IP was flagged

After you've run your scan:

1. Verify that your IP is listed under "Events" on the Overview page.
2. Verify that you received an email indicating that your IP was flagged.

From the "Events" module on the Overview page, click on the flagged IP to view additional information. You can also click through to the event from the event email.

From the event page you can view the requests that led to the decision being made as well as any subsequent malicious requests. For information on using the search page see Investigating an attack.

Manually Verifying blocking

1. If your agent mode is set to "not blocking" (the default), you can verify that subsequent malicious requests are allowed by visiting your site with a malicious payload (e.g., `https://www.example.com/?q=<script>alert('xss')</script>`).
2. To test "blocking" mode, click **Not blocking** in the site navigation and then **Manage**. On the next page, switch the agent mode to **Blocking**.
3. After the configuration change has propagated to your agents (it can take up to a minute), visit the same URL. The server should respond with a 406 response code and the request will be blocked.
4. Visit your site normally (e.g., `https://www.example.com/`) and test basic functionality (navigation, search, etc.). Even though the IP is flagged, you should see that normal site traffic is unaffected.

NGINX C Binary

NGINX C Binary Module Release Notes

1.1.5 2021-05-17

- 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 Alpine 3.15 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)

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

1.1.3 2020-11-24

- Improved support for setting headers to HTTP/0.9 request if agent responds with headers

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

- 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

1.0.35 2020-01-21

- Released packages for Nginx 1.17.8

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

1.0.22 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 1.17.1

1.0.19 2019-06-21

- 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

1.0.5 2019-01-29

- Updated package for NGINX Plus with dependency `nginx-plus-module-ndk` - NGINX Plus Release 17 (R17)
- Cleaned up package deinstall 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

- Recertified with latest release - NGINX Plus Release 17 (R17)

1.0.2 2018-12-05

- Recertified with latest release - NGINX Plus Release 16 (R16)

1.0.1 2018-11-28

- Updated config checks for port and time values
- Updated README's for install

1.0.0 2018-11-01

- Built packages for Nginx 1.15.2 and NGINX Plus

IPv6 support

Signal Sciences provides full support for IPv6 in the product, including:

1. Detection and decisioning — Requests are appropriately tagged and IPv6 addresses can be automatically flagged within the product.
2. Blocklist and allowlist support — IPv6 addresses can be blocklisted and allowlisted within the UI.
3. Search — IPv6 addresses can be filtered within search.
4. Country/DNS lookups — IPv6 addressed are resolved and mapped to countries, where possible.

SE Linux Support

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies, including United States Department of Defense–style mandatory access controls (MAC).

All official CentOS Linux builds come pre-configured with SE Linux enabled and set to enforcement mode. There are two approaches to running the agent on a system with SE Linux enabled:

1. Set SELinux to Permissive mode or disable SELinux completely

2. Configure SELinux to allow the module and agent to communicate

Symptoms of SELinux enabled in enforcement mode

Often times system administrators may not be aware that SE Linux is installed until they hit 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 SE Linux, run the command `sestatus` which should produce 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
```

```
Policy MLS status: enabled
Policy deny_unknown status: allowed
Max kernel policy version: 28
```

Set SE Linux to Permissive mode or disable SE Linux completely

The main configuration file for SELinux is `/etc/selinux/config`. We can 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. Verify the new status for SELinux with another `sestatus` command.

Configure SE Linux 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).

- Log in as root to install the SigSci agent and module.
- Restart the web server and start the agent. Also browse the web site to cause the module to invoke communications with the agent. If in permissive mode, things should work but the audit log will get populated with messages of what would be blocked. If in enforced mode, the same log messages will be appended to the audit log.
- Now 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`
- Now install the policy package file with `semodule -i sigscilua.pp`
- Verify policy was installed and loaded with `semodule -l`

At this point you should 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 easily be deployed through Dropwizard.

Installation

Download

Download manually

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

Access with Maven

Dropwizard supports standard Java servlet filters, but you will need to register the filter class.

Additional information about Dropwizard servet filter support can be found here.

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 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 Science supports a native module for Nginx. This enables you to easily wrap the existing ingress-nginx controller to install the Signal Science module.

Wrap the Base nginx-ingress-controller to Install the Signal Science 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.

A prebuilt container can be pulled from Docker Hub with: `docker pull signalsciences/sigsci-nginx-ingress-controller:0.47.0`

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 repo:

   ```
   helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
   ```

2. Add `SIGSCI_ACCESSKEYID` and `SIGSCI_SECRETACCESSKEY` to the sigsci-values.yaml file.

You can specify a namespace with `-n` flag:

```
helm install -n NAMESPACE -f values-sigsci.yaml my-ingress ingress-nginx/ingress-nginx
```

After a few minutes, you should see the agent in your Signal Sciences console.

4. 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:
      paths:
      - pathType: Prefix
        path: /testpath
        backend:
          service:
            name: NAME OF SERVICE
            port:
              number: 80
```

Helm Upgrade with Override File

1. To update the ingress-nginx charts, update the `sigsci-nginx-ingress-controller` to the latest version in the sigsci-values.yaml file:

```
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: "0.47.0"
      pullPolicy: IfNotPresent
```

2. Then run helm upgrade with override file. This example is running helm upgrade against the `my-ingress` release created in step 3 of the previous section:

```
helm upgrade -f sigsci-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
```

Uninstall Release

Uninstall release `my-ingress`:

```
helm uninstall my-ingress
```

If it's not in the default namespace, use `-n` to specify the namesapce:

```
helm uninstall -n NAMESPACE my-ingress
```

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 just 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.

Getting and Updating the Signal Sciences Agent Container Image

The official `signalsciences/sigsci-agent` container image available from the Signal Sciences account on Docker Hub is the recommended place to get the image. If you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

The documentation references 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 this case you should specify a 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 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. To keep some consistency, 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, then just replace `latest` with the agent version. You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
   image: signalsciences/sigsci-agent:4.1.0
   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 the `latest`), apply a custom tag, then use that custom tag in the configuration. You will want to specify `imagePullPolicy: Never` so that local images are only updated manually. 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
```

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 (_) (e.g., 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 (ID and secret key)
- A volume to write temporary files

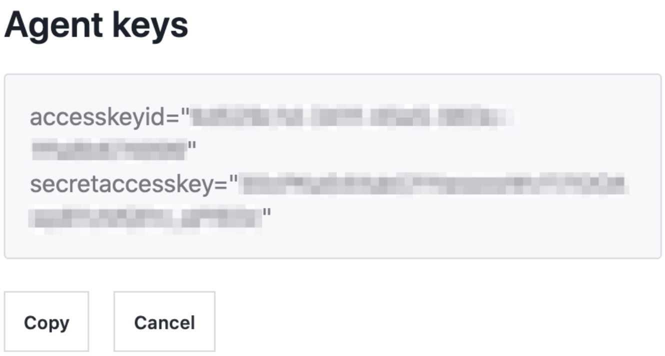
Agent Credentials

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

- SIGSCI_ACCESSKEYID: Identifies the site that the agent is configured against
- SIGSCI_SECRETACCESSKEY: The shared secret key to authenticate and authorize the agent

The credentials can be found by following these steps:

1. Log into the Signal Sciences console.
2. Click on **Agents**. The Agents page appears.
3. On the Agents page click **View Agent Keys**. The agent keys window appears.
4. Copy down the **Access Key** and **Secret Key** for later use.



Agent keys

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

Because of the sensitive nature of these values, it is recommended to use the builtin `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded the 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.

Using secrets via environment variables is done using the valueFrom option instead of the value option such as follows:

```
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 documentation 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 with something like the following YAML:

```
apiVersion: v1
kind: Secret
```

```
  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:

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

See the documentation on secrets for more details.

Agent Temporary Volume

For added security, it is recommended that the `sigsci-agent` container be executed with the root filesystem mounted read only. The agent, however, 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. Typically this is just configured in the `volumes` section of a deployment.

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

Containers would then typically 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 at https://kubernetes.github.io/ingress-nginx/deploy/#prerequisite-generic-deployment-command. The main additions are:

- Changing the ingress container to load the custom Signal Sciences Module/ingress container and adding 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:0.47.0
        ...
        volumeMounts:
          - name: sigsci-tmp
            mountPath: /sigsci/tmp
...
```

- Loading the Signal Sciences Module in nginx.conf via ConfigMap:

```
kind: ConfigMap
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
```

- Adding 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
...
```

- And defining 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:
  - 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
```

```
          name: sigsci.my-site-name-here
          key: secretaccesskey
...
```

2. Pull or build the Nginx ingress + Signal Sciences Module container. Set whatever registry & repository name you'd like here, just be sure to set the image to match in `mandatory.yaml`:

   ```
   docker pull signalsciences/sigsci-nginx-ingress-controller:0.47.0
   ```

3. Deploy using modified Generic Deployment:

   ```
   kubectl apply -f mandatory.yaml
   ```

4. Create service to expose Ingress Controller. The steps necessary are dependent on your cloud provider. Official instructions can be found at https://kubernetes.github.io/ingress-nginx/deploy/#provider-specific-steps.

   Here 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 Ingress Resource

   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
   ```

Ubuntu NGINX-Plus

Add the Package Repositories

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

```
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”

Cut-and-paste the following script into a terminal:

```
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”

Cut-and-paste 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”

Cut-and-paste 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 the module with apt

Then install the module by running the following command for your NGINX version:

NGINX+ 19

```
sudo apt-get install nginx-module-sigsci-nxp=1.17.3*
```

NGINX+ 18

```
sudo apt-get install nginx-module-sigsci-nxp=1.15.10*
```

NGINX+ 17

```
sudo apt-get install nginx-module-sigsci-nxp=1.15.7*
```

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

```
sudo service nginx restart
```

Amazon Linux Agent Installation

Add the Package Repositories

Amazon Linux 2

Amazon Linux 2 is most similar to CentOS 7 and reuses the same configuration.

Cut-and-paste the following script:

```
sudo tee /etc/yum.repos.d/sigsci.repo <<-'EOF'
[sigsci_release]
```

```
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

Amazon Linux 2015.09.01 is most similar to CentOS 6 and reuses the same configuration.

Cut-and-paste 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 the Signal Sciences Agent Package

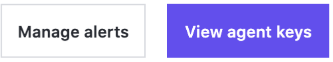
1. To install the package, running the following command.

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

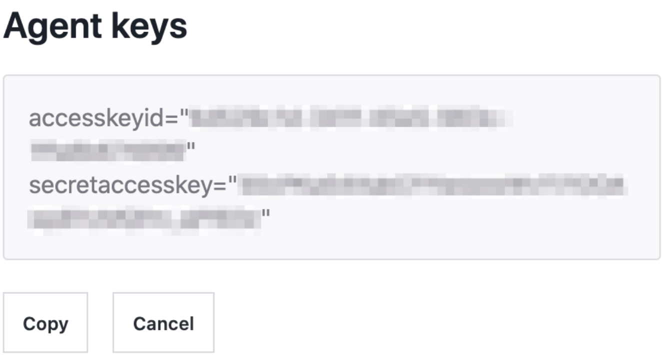
2. Create the file `/etc/sigsci/agent.conf`

3. Configure the agent by inputting the Agent Access Key and Agent Secret Key into the `/etc/sigsci/agent.conf`.

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

   - The Agent Access Key and Agent Secret Key will be visible within the window:

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

   Example /etc/sigsci/agent.conf

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

   Additional configuration options are listed on the agent configuration page.

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

Amazon Linux 2015.09.01

```
start sigsci-agent
```

Next Steps

Install the Signal Sciences Module:

- Explore module options

Amazon Linux Apache Module Install

Amazon Linux 2

1. First install the Signal Sciences Apache Module using `yum`.

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

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration after the “Dynamic Shared Object (DSO) Support” section:

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

3. Restart Apache httpd.

   ```
   sudo systemctl restart httpd
   ```

Amazon Linux 2015.09.01

1. Amazon Linux 2015.09.01 can have two versions of Apache:

   Apache 2.2 Install Command:

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

   Apache 2.4 Install Command:

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

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration after the “Dynamic Shared Object (DSO) Support” section:

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

3. Restart Apache httpd.

   ```
   sudo service httpd restart
   ```

Next Steps

- Verify Agent and Module Installation

Explore other installation options:

- Explore module options

Lists

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,

- Countries
- IP addresses
- Strings
- Wildcards

> **Note:** Lists support CIDR notation for IP address ranges.

Creating a List

Corp Lists

1. Go to **Corp Rules** > **Corp Lists** and click **Add corp list**
2. Select the type of data the list will contain
3. Name the list
4. Provide an optional description for the list
5. Input the items that will comprise the list, each entry must be on its own line
6. Click **Create corp list**

After creating the Corp List, use it on specific sites by selecting the site from the dropdown menu at the top of the console and using it in a rule.

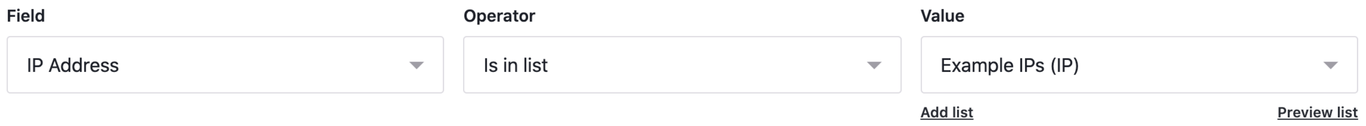
> **Note:** Only Owner users can create, edit, and delete Corp Lists. This is because Corp Lists have the ability to manipulate traffic across every site and other user types can only manage Rules and Lists for sites they have access to.

Site Lists

1. Go to **Rules** > **Site Lists** and click **New list**
2. Select the type of data the list will contain
3. Name the list
4. Provide an optional description for the list
5. Input the items that will comprise the list, each entry must be on its own line
6. Click **Save 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)
		Add list Preview list

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

Install the Agent

Install the Signal Sciences Agent by following the instructions for your environment here: https://docs.signalsciences.net/install-guides/#step-1-agent-installation

Add Agent TCP Listener Config Option

`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>
```

Download the NGINX Module

Download and extract the latest Signal Sciences NGINX module by running the following commands:

```
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
```

Update the Kong Plugin Config Options

As with the agent configuration file, you will also need to edit the Kong plugin's config options to reflect the host IP address and the port used for communication with the agent. Edit the following lines in `/opt/sigsci/nginx/kong/plugins/signalsciences/handler.lua` to replace `"localhost"` and `12345` with the host IP address and port:

```
sigsci.agenthost = "localhost"
sigsci.agentport = 12345
```

Update the Kong Configuration File

Add the following lines to the Kong configuration file at `/etc/kong/kong.conf`:

```
plugins=signalsciences
lua_package_path=/opt/sigsci/nginx/?.lua
```

Enable the Kong Plugin

Enable the Kong plugin by running the following command, after replacing `<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 easily 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

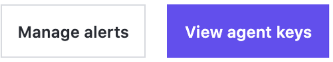
1. Application developers will need to specify the buildpack with the `cf push` command:

   ```
   cf push YOUR-APP -b https://github.com/signalsciences/sigsci-cloudfoundry-buildpack.git -b APP_BUILDPACK
   ```

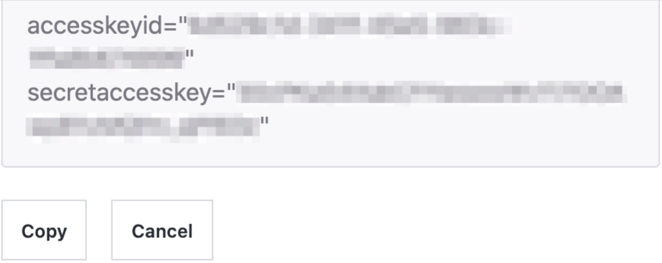
2. Set your agent's access key and secret using the `cf set-env` command. Example:

   ```
   cf set-env <application name> SIGSCI_ACCESSKEYID <key>
   cf set-env <application name> SIGSCI_SECRETACCESSKEY <secret>
   ```

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

   - The Agent Access Key and Agent Secret Key will be visible within the window:



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

3. 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`:

```
cf set-env YOUR-APP <variable name> "<value>"
```

To have these changes take effect, you must at least re-stage your app:

```
cf restage YOUR-APP
```

Server Hostname

Each time you deploy your application, IBM Cloud will automatically assign a new random name for the agent. If you'd prefer to specify an agent name for each deployment, set the `SIGSCI_SERVER_HOSTNAME` environment variable:

```
cf set-env <application name> SIGSCI_SERVER_HOSTNAME <agent name>
```

Reverse Proxy Upstream

If you would like to define upstream host(s) that the Agent will proxy requests to, use the `SIGSCI_REVERSE_PROXY_UPSTREAM` option. This variable is optional with a default value of `127.0.0.1:8081`:

```
cf set-env <application name> SIGSCI_REVERSE_PROXY_UPSTREAM <ip:port>
```

Access Logs

If you would like to enable the agent's access logging, set the `SIGSCI_REVERSE_PROXY_ACCESSLOG` environment variable:

```
cf set-env <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. If you prefer to specify which agent version to install, set the `SIGSCI_AGENT_VERSION` environment variable:

```
cf set-env <application name> SIGSCI_AGENT_VERSION 4.7.0
```

Health Checks

Currently, IBM Cloud does not support HTTP health checks native to Cloud Foundry so 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.

`SIGSCI_HC` - set this to "true" to enable health checking. Example:

```
cf set-env <application name> SIGSCI_HC true
```

`SIGSCI_HC_CONFIG` - set this environment variable to configure the health check. If you do not set this environment variable the default settings will be used.

Default health check settings: check the "/" path every 5 seconds, if the agent listener returns a 502 for 5 sequential checks then the health check fails. Additionally, if the application process does not return a 200 response for 3 sequential tries 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 `:`.

Field	Description
`frequency`	How often to perform the check in seconds, example: 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 this: `5:/:502:5:200:3`

An example custom setting: check the "`/health.html`" path every 10 seconds, if the agent listener returns a 502 for 10 sequential tries the health check fails or if the application process does not return a 200 for 5 sequential tries the health check fails - looks like this: `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. If you prefer to ensure that your application never starts with out being protected by the Signal Sciences agent, use the `SIGSCI_REQUIRED` environment variable. Example:

```
cf set-env <application name> SIGSCI_REQUIRED true
```

Additional configuration options are listed on the agent configuration page

Making Security Visible

The teams that we've seen most successful with Signal Sciences are the ones that share their security data with the developers and operations engineers responsible for their web applications. Now that you've successfully verified that data is being sent to Signal Sciences and blocking mode is working, here are some ways that you can share that data with your wider organization:

1. Setting up the Monitor View on a TV
2. Inviting members as Observers
3. Setting up integrations

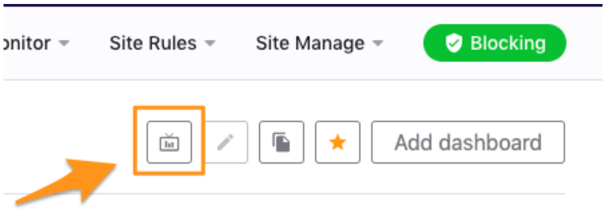
Setting up the Monitor View on a TV

We've found that one of the best ways to get other teams interested in security is by putting up security dashboards on a TV. You can do this easily by using our read-only URL on the Monitor View page.

The Monitor View will reflect the Overview page as you've customized it. In the default grid view, the Monitor will simultaneously show up to the first six cards on the Overview page. Users can customize the cards and their arrangement from the Overview page. In the carousel view, the Monitor will cycle through all cards on the Overview page.

1. Go to the Overview Page for the site by selecting the site in the site-selection dropdown menu, or clicking the name of the site on the left of the navigation bar.

2. Click the "Monitor View" icon near the upper-right corner:



3. Click **Read-only URL**.

4. Click **Enable**.

5. Copy the link and open it on the TV you'd like to display it on.

If necessary, you can invalidate and generate a new URL or disable the read-only URL altogether.

site (for example, to dig in to a spike they saw on the Monitor View), but they can't make any changes (e.g., allowlisting or blocklisting IPs or expiring flags). To invite members as Observers:

1. On the Site Members page, click **Add Member**.
2. Enter the email address of the member you'd like to add.
3. Choose **Observer**.
4. Click **Invite User**.

They'll be sent an invitation which expires in 24 hours.

Setting up integrations

We add new integrations all the time, so if you don't see something you're looking for, let us know. In particular, these are some of the integrations we encourage teams to set up:

1. Integrating with your messaging app.
2. Integrating with your incident response flow.
3. Integrating with your other systems.

Integrating with your messaging app

If your team uses a chat client, you can be alerted when any activity occurs (e.g., an IP being flagged, when the agent mode is changed, an IP is allowlisted, etc...). We currently support Slack, and if you use IRC, you can also create your own integration using our generic webhook.

Integrating with your incident response flow

If you have an existing incident response flow, you can be alerted or we can create a ticket when an IP is flagged is malicious. We currently support PagerDuty, VictorOps, and JIRA.

Integrating with other systems

If you have another use case that we don't currently support, you can also use our generic webhook to be notified when any activity occurs. That said, let us know if there's another integration you'd like to see!

More Details On Integrations

For detailed instructions on how to configure integrations see the Integrations page.

Setting up Agent Alerts

You can set up alerts to inform you when the product isn't functioning properly. 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) for whenever 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.

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.

> **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.

Apache

Apache Module Release Notes

Unreleased

1.9.0 2022-01-18

- Improved `Content-Type` header inspection
- Added Debian 11 (bullseye) support

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 prerequest to agent

1.7.10 2019-05-07

- Added support for Apache 2.4 for Windows

1.7.9 2019-04-23

- Updated internal tooling

1.7.8 2019-03-25

- Added `ServerName` field to agent messages

1.7.7 2019-02-15

- 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 runlist 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. ie. 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
- Added .tar.gz packages for CentOS

1.5.4 2017-10-12

- Improved error logs
- Added debugging for specific customer issue

1.5.3 2017-09-11

- Standardized defaults across modules and document

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 millisecs to microsecs
- 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

0.318 2016-03-21

- Brought all version numbering in sync with the new packages

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`)

- 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
- Removed rare debug messages that were incorrectly going to stderr

0.139 2015-06-14

- 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

JIRA

Our JIRA issue integration creates an issue when IPs are flagged on Signal Sciences.

Adding a JIRA issue integration

JIRA issue integrations are configured per project.

1. Within JIRA, go to **Settings** > **Site Administration** > **User management**.
2. Create a user for the integration to use (e.g., "Create Ticket User").
3. Confirm the user's account and set a password.
4. Create an API token for that user
5. On Signal Sciences, go to **Manage** > **Site Integrations**.
6. Click **Add site integration** and select the **Jira Issue** integration.
7. Enter the host for your JIRA instance and the username of the user and API Token you created.
8. Enter the key for the project you'd like to create the ticket in.
9. Click **Add**.

Activity types

Activity type	Description
flag	An IP was flagged
agentAlert	An agent alert was triggered

IP Anonymization

What is IP Anonymization?

IP Anonymization is a site-level customization that changes the way Signal Sciences stores and uses remote client IP addresses. By default IPs are not anonymized. When a customer chooses to enable IP Anonymization, agents for a specific site will anonymize an IP before sending it to the cloud. Signal Sciences will convert IPs into the anonymized IPv6 by performing a one-way hash. As a result, Signal Sciences databases will not have knowledge of the actual IP and it will appear anonymized throughout the console.

Actual IPs are converted to anonymous IPv6 using rfc7343.

The IP is anonymized in all headers and data fields with the anonymized IPv6. In addition, the actual IP 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)

How do I enable IP Anonymization?

IP Anonymization can be enabled by navigating to **Manage** > **Site Settings**. IP Anonymization will be listed as disabled by default. To enable it, select the **Active** radio button. You will have to acknowledge and consent that some functionality will not work with IP Anonymization enabled, as explained in the note above.

Installing the Java Module on Weblogic

The Signal Sciences Java module can easily be deployed on WebLogic servers.

Compatibility

The Signal Sciences Java module is compatible with WebLogic version 12c (12.2.1) and higher.

Installation

To deploy the Signal Sciences Java module on Weblogic servers, you will first need to deploy 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

tcp://127.0.0.1:9999	
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 IPs to be excluded from filter processing.
excludeIpRange optional, no default	A comma-delimited list of IP ranges or specific IPs 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 are 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>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<!-- end Signal Sciences Filter -->
```

Node.js Module Install

Compatibility

This is compatible with Node 0.10 to 12.X. All dependencies are explicitly specified in the `npm-shrinkwrap.json` file.

Installation

One may install the latest version from npmjs.com using:

```
npm install sigsci-module-nodejs
```

For specific releases prior to 1.5.3, installation can be performed from the release archive:

```
npm install https://dl.signalsciences.net/sigsci-module-nodejs/<VERSION>/sigsci-module-nodejs-<VERSION>.tgz
```

See the package archive for a list of versions.

Usage For Native Applications

2. Creating a `SigSci` object, adding or overriding any parameters.
3. Wrapping your native dispatcher

```
// Import sigsci module
var Sigsci = require('sigsci-module-nodejs')

// your code

// 2. Create a SigSci object
var sigsci = new Sigsci({
 path: '/var/run/sigsci.sock'
 // other parameters here
})

// 3. Wrap dispatcher with sigsci.wrap

// WAS http.createServer(dispatcher).listen(8085, '127.0.0.1')

http.createServer(sigsci.wrap(dispatcher)).listen(8085, '127.0.0.1')
```

Usage For Node.js Express

The Node.js express module is exposed as a express middleware and is typically inserted as the first middleware, right after the `var app = express()` statement . See the express Using Middleware documentation for more details.

In particular, adding Signal Sciences involves:

1. Importing the `sigsci` module
2. Creating a `SigSci` object, adding or overriding any parameters.
3. Inserting the SigSci module

```
// Import sigsci module
var Sigsci = require('sigsci-module-nodejs')

// your code

// 2. Create a SigSci object
var sigsci = new Sigsci({
path: '/var/run/sigsci.sock'
// other parameters here
})

// 3. Insert the SigSci module middleware right after the express app is created.

// WAS
// var app = express()
//
// Other routes and middleware
// app.use(...)
// app.get('/route', ...)


// NEW
var app = express()
app.use(sigsci.express())    // NEW

// continue with existing routes and middleware
app.use(...)
app.get('/route', ...)
```

results, Signal Sciences should hook into the `NotFound` event. See the Restify node server api for more details.

Usage for Node.js Hapi v14

Add to the top of the file:

```
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)
})
```

Usage for Node.js Hapi v17 & v18

Add to the top of the file:

```
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())
 // Add SigSci request lifecycle methods, e.g.
 // server.route({
 //   method: ['POST', 'PUT', 'PATCH', 'DELETE'],
 //   config: {
 //     payload: {
 //       parse: false,
 //       maxBytes: 10 * 1024 * 1024,
 //       output: 'data'
 //     }
 // })
};
init();
```

Usage for Node.js KOA

Add to the top of the file:

```
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
//     let res = ctx.res
       // add your code here
// }

// setup your endpoints here
// router.all('/response', dispatcher)

server.use(sigsci.koa())
server.use(router.routes())

server.listen(8085);
```

Parameters

One can pass various parameters to `SigSci` object:

```
var sigsci = new Sigsci({
path: '/var/run/sigsci.sock'
// other parameters here
})
```

The most important ones are listed here. See the file SigSci.js for more details and default values.

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))`

Next Steps

- Verify Agent and Module Installation

Explore other installation 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 just 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.

Getting and Updating the Signal Sciences Agent Container Image

The official `signalsciences/sigsci-agent` container image available from the Signal Sciences account on Docker Hub is the recommended place to get the image. If you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

The documentation references 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 this case you should specify a 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 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. To keep some consistency, 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, then just replace `latest` with the agent version. You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
   image: signalsciences/sigsci-agent:4.1.0
   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 the `latest`), apply a custom tag, then use that custom tag in the configuration. You will want to specify `imagePullPolicy: Never` so that local images are only updated manually. You will need to periodically update the local image to keep the agent up-to-date.

For example:

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 (_) (e.g., 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 (ID and secret key)
- A volume to write temporary files

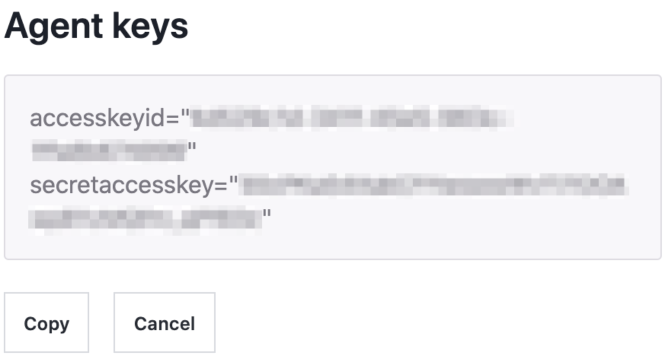
Agent Credentials

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

- SIGSCI_ACCESSKEYID: Identifies the site that the agent is configured against
- SIGSCI_SECRETACCESSKEY: The shared secret key to authenticate and authorize the agent

The credentials can be found by following these steps:

1. Log into the Signal Sciences console.
2. Click on **Agents**. The Agents page appears.
3. On the Agents page click **View Agent Keys**. The agent keys window appears.
4. Copy down the **Access Key** and **Secret Key** for later use.



Agent keys

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

Because of the sensitive nature of these values, it is recommended to use the builtin `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded the 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.

Using secrets via environment variables is done using the valueFrom option instead of the value option such as follows:

```
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 documentation 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 with something like the following YAML:

```
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:

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

See the documentation on secrets for more details.

Agent Temporary Volume

For added security, it is recommended that the `sigsci-agent` container be executed with the root filesystem mounted read only. The agent, however, 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. Typically this is just configured in the `volumes` section of a deployment.

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

Containers would then typically 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

In addition to the general deployment types, the Signal Sciences Agent can be deployed for integration with the Envoy Proxy via the External Authorization (`ext_authz`), HTTP filter. This filter will communicate with the `sigsci-agent` via gRPC.

Generic Envoy Proxy

Configuration for envoy and the `sigsci-agent` are documented with the other modules in the envoy install guide. The following documentation is for deploying the sigsci-agent as a sidecar to your existing envoy configuration. Deploying sigsci-agent container as a sidecar to envoy is similar to a typical module based deployment, but configuration is slightly different.

To do this, you must:

• Modify your existing envoy configuration as noted in the envoy install guide • 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

Modify the Envoy Proxy Configuration

Modify your existing envoy configuration as noted in the envoy install guide.

Add the Signal Sciences Agent as an Envoy gRPC Service:

```
   - 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
```

Adding the Signal Sciences Agent Temp Volume Definition to the Deployment

Finally, the agent temp volume needs to be defined for use by the other containers in the pod. This just uses the builtin `emptyDir: {}` volume type:

```
...
   volumes:
   # Define a volume where sigsci-agent will write temp data and share the socket file,
```

Red Hat NGINX 1.14.1+

Add the Package Repositories

Red Hat CentOS 8

Cut-and-paste 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

Cut-and-paste 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.

Cut-and-paste 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 the module with yum

Then install the module by running the following command, replacing “NN.NN” with your Nginx version number:

```
sudo yum install nginx-module-sigsci-nxo-1.NN.NN*
```

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

RHEL 7/CentOS 7 and higher

```
systemctl restart nginx
```

RHEL 6/CentOS 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.

Download

The Apache module is delivered as a zip archive. The package contains a DLL that you will extract and configure.

The module can be downloaded from:

https://dl.signalsciences.net/sigsci-module-apache/sigsci-module-apache_latest.zip

Install

1. First install the Signal Sciences Apache Module.

   ```
   unzip sigsci-module-apache_latest.zip
   copy mod_sigsci.so <path to Apache>\modules\
   ```

2. Enable the Signal Sciences module for Apache by adding the following line to your Apache configuration file (`httpd.conf`) after the “Dynamic Shared Object (DSO) Support” section:

   ```
   LoadModule signalsciences_module modules/mod_sigsci.so
   ```

3. Test that the configuration is correct:

   ```
   httpd.exe -n "MyServiceName" -t
   ```

4. Start the Apache service as normal, for example:

   ```
   net start Apache2.4
   ```

   Or restart the Apache service with the following example command:

   ```
   httpd.exe -k restart -n "MyServiceName"
   ```

Next Steps

- Verify Agent and Module Installation

Explore other installation options:

Alpine Linux Agent Installation

Run the Alpine Docker Container

If Alpine is being run in a Docker container, start the container. For example:

```
docker run -it -p 80:80 alpine:3.11 /bin/sh
```

Add the Package Repositories

We'll first add in the Signal Sciences apk repositories as this simplifies the installation process.

Alpine in Container

On the running Alpine container, cut-and-paste the following script into a terminal:

Alpine in VM or bare-metal

If running Alpine on a VM or bare-metal, cut-and-paste the following script into a terminal:

Verify the downloaded key contains the proper key by running this command:

```
openssl rsa -pubin -in /etc/apk/keys/sigsci_apk.pub -text -noout
```

Expected modulus output:

```
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 Signal Sciences Agent Package

1. To install the package, running the following command:

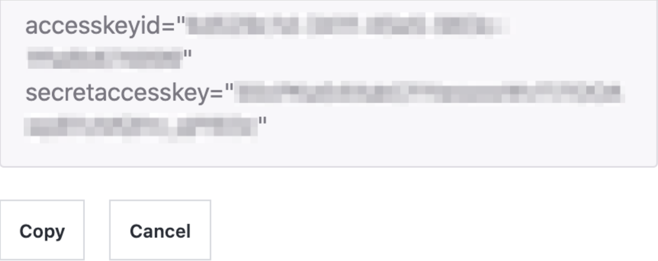
   ```
   sudo apk add sigsci-agent
   ```

2. Create the file `/etc/sigsci/agent.conf`

3. Configure the agent by inputting the Agent Access Key and Agent Secret Key into the `/etc/sigsci/agent.conf`.

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

   - The Agent Access Key and Agent Secret Key will be visible within the window:



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

Example /etc/sigsci/agent.conf

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

Additional configuration options are listed on the agent configuration page.

4. Start the Signal Sciences Agent

Alpine in Container

Start the Signal Sciences Agent running in Docker:

Alpine in VM or bare-metal

The following is required to have the agent start on reboot:

Next Steps

Install the Signal Sciences Module:

- Explore module options

Custom Signals

> **Note:** Custom Signals are not supported on the Essential platform.

About 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.

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.

> **Note:** Only Owner users can create, edit, and delete Corp Signals.

Creating Signals

Corp Signals

1. Go to **Corp Rules** > **Corp Signals** > **Add corp signal**

2. Assign a name to the new signal

3. Provide an optional description for the signal

4. Click **Create corp signal**

1. Go to **Rules** > **Site Signals** > **Add site signal**
2. Assign a name to the new signal
3. Provide an optional description for the signal
4. 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.

OpenShift Install

Signal Sciences is Primed for Openshift! The Signal Sciences agent can be easily 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 install. 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. In the examples below we are using a directory that a non root user would have access to. You may specify a different location, but ensure your non root user account has the read/write permissions to that location.

> **Note:** For agent install instructions see Red Hat Agent Install

Set the `SIGSCI_RPC_ADDRESS` environment variable in your Dockerfile:

```
ENV SIGSCI_RPC_ADDRESS unix:/tmp/sigsci.sock
```

Export the `SIGSCI_RPC_ADDRESS` environment variable in a script when your container starts:

```
export SIGSCI_RPC_ADDRESS=unix:/tmp/sigsci.sock
```

Set the `rpc-address` configuration option in your `agent.conf` file:

```
rpc-address="unix:/tmp/sigsci.sock"
```

Additional configuration options are listed on the agent configuration page.

Configuring the Module

Apache

Add the `AgentHost` directive to your `httpd.conf` file. For module install instructions see Red Hat Module Install.

```
# This line must be after the Signal Sciences module is loaded
AgentHost "/tmp/sigsci.sock"
```

Nginx

Update the `sigsci.agenthost` directive in the module's configuration file, `/opt/sigsci/nginx/sigsci.conf`. Note, you will need to remove the `--` to uncomment the line. For module install instructions see NGINX Module Install.

```
sigsci.agenthost = "unix:/tmp/sigsci.sock"
```

Example Dockerfile

```
...

# 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

# Configure your web server with the Signal Sciences module
# Here we enable the module with Apache, and configure Signal Sciences
# module by specifying a
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

...
```

IIS

SignalSciences IIS Module Release Notes

Unreleased

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

- Added HTTP redirect support
- Removed restrictions on HTTP methods
- Fixed an issue where Windows eventlog 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 env. 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

- Improved the installer, working on older versions of Windows back to Server 2008r2
- Changed the default behavior to install as per-machine (instead of per-user). Because of this, previous installs may need to be 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

- 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
- Fixed outdated module detection

1.6.5 2017-11-08

- 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.

Mailing List

Our mailing list integration allows you to receive email notifications for certain activity on Signal Sciences.

Adding a mailing list integration

- For a Corp Integration, navigate to **Corp Manage** > **Corp Audit Log** > **Manage corp integrations** > **Add corp integration** and select the **Mailing List** integration.
- For a Site Integration, navigate to **Manage** > **Site Integrations** > **Add site integration** and select the **Mailing List** integration.

1. Enter the email address or alias you want notifications to be sent to.
2. Select if you want email notifications for all activity or specific activity.

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
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 was flagged
expireFlag	An IP 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

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
weeklyDigest	Weekly digest sent

Real Remote (Client) IP Addresses

Often the server being protected is behind a load balancer or other proxy. In this case, the server will see this load balancer or proxy IP address as the remote (client) IP address. To get around this common issue, most load balancers or proxies offer the ability to record the real 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

Alternative client IP headers for the agent to source the real remote IP address from can be set directly in the console by going to **Manage** > **Site Settings**.

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. Headers are not case sensitive. 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.

Setting Alternative Headers Directly in the Agent

Alternative HTTP Header

If your environment uses a different HTTP header to pass the real remote address, such as using `X-Real-Ip`, you will need to configure the agent by adding a line to the `/etc/sigsci/agent.conf` file specifying the correct header name to use.

```
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, however, 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. To do this, you will need to configure the agent by adding a line to the `/etc/sigsci/agent.conf` to specify that *no header* should be used. If this is not done, then the agent may misinterpret the remote address.

```
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.

The agent will report:

```
127.0.0.1
```

To ensure that the true IP address is being identified in the above case, the agent can be configured to read XFF IP addresses from right to left instead. Use the `local-networks` directive by adding the following line to your agent configuration file (`/etc/sigsci/agent.conf`):

```
local-networks = "private"
```

By setting the `local-networks` directive 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.

See the documentation on this module for more details: http://nginx.org/en/docs/http/ngx_http_realip_module.html

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;
```

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.

See the documentation on this module for more details: https://httpd.apache.org/docs/2.4/mod/mod_remoteip.html

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
RemoteIPInternalProxy 192.0.2.54
RemoteIPHeader X-Forwarded-For
```

> **Note:** On Debian/Ubuntu, you will typically use the `a2enmod` command to enable the module vs. adding the LoadModule directive directly. For example:

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:** https://github.com/mpyatishev/hosting_tools/blob/master/mod_realip2.c
- **mod_extract_forwarded:** http://www.cotds.org/mod_extract_forwarded2/
- **mod_rpaf:** https://github.com/y-ken/mod_rpaf

Known Issues

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 HTTP(S) load balancer. Because of this, the real remote address cannot be obtained. The HTTP(S) 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

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 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/

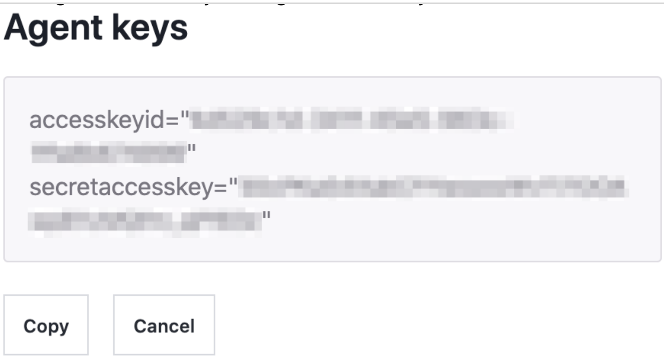
1. In the Azure Portal, go to **App Services** and select your web app

2. Set environment variables

   - Click on **Configuration** > **Application settings** > **New application setting** and set 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>
     ```

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:



Agent keys

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

- Click on **Save** after adding the application settings

3. Restart the web app by clicking on **Overview** in the side bar and then clicking on the **Stop** and **Start** buttons

Install the Signal Sciences Site Extension

1. In the Azure Portal, go to **App Services** and select your web app
2. Stop the web app by clicking on **Overview** in the side bar and then clicking on the **Stop** button
3. Add the site extension by going to **Extensions** in the sidebar and clicking on **Add** > **Choose Extension** > **Signal Sciences WAF** > **OK**
4. Start the web app again by clicking on **Overview** in the side bar and then clicking on the **Start** button

   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.

Managing the Signal Sciences Site Extension

Uninstalling the Signal Sciences Site Extension

1. In the Azure Portal, go to **App Services** and select your web app
2. Stop the web app by clicking on **Overview** in the side bar and then clicking on the **Stop** button
3. Delete the site extension by clicking on **Extensions** in the sidebar and clicking on **Signal Sciences WAF** > **Delete**.

Upgrading the Signal Sciences Agent and Module

There are two methods for upgrading the Signal Sciences agent and module that are downloaded when the site extension is first installed:

- Uninstall and reinstall the site extension. When the extension is reinstalled, the latest version of the Signal Sciences agent and IIS module will be downloaded and installed.

OR

- Open the Azure CLI and run the `install.cmd` script in the site extension directory. This method could also be used in a PowerShell script for automating the upgrade of multiple agents.

  1. Open the windows `cmd` shell by clicking on **Console** in the sidebar
  2. 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** > **eventlog.xml**

Introduction

In this example, the Signal Sciences runs in a Docker sidecar and integrates directly with an Istio service mesh deployed on the application. In this configuration, Signal Sciences can be configured 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 just 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.

Getting and Updating the Signal Sciences Agent Container Image

The official `signalsciences/sigsci-agent` container image available from the Signal Sciences account on Docker Hub is the recommended place to get the image. If you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

The documentation references 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 this case you should specify a 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 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. To keep some consistency, 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, then just replace `latest` with the agent version. You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
  image: signalsciences/sigsci-agent:4.1.0
  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 the `latest`), apply a custom tag, then use that custom tag in the configuration. You will want to specify `imagePullPolicy: Never` so that local images are only updated manually. You will need to periodically update the local image to keep the agent up-to-date.

For example:

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 (_) (e.g., 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 (ID and secret key)
- A volume to write temporary files

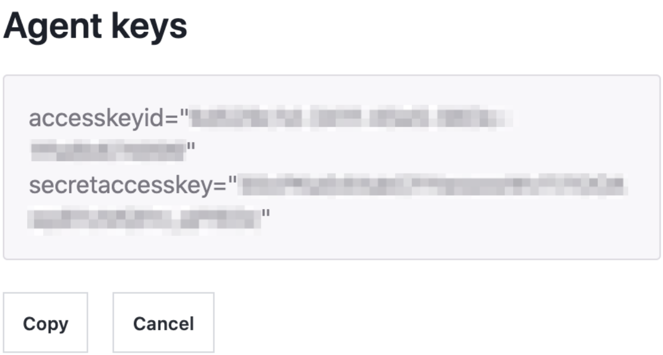
Agent Credentials

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

- SIGSCI_ACCESSKEYID: Identifies the site that the agent is configured against
- SIGSCI_SECRETACCESSKEY: The shared secret key to authenticate and authorize the agent

The credentials can be found by following these steps:

1. Log into the Signal Sciences console.
2. Click on **Agents**. The Agents page appears.
3. On the Agents page click **View Agent Keys**. The agent keys window appears.
4. Copy down the **Access Key** and **Secret Key** for later use.



Agent keys

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

Because of the sensitive nature of these values, it is recommended to use the builtin `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded the 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.

Using secrets via environment variables is done using the valueFrom option instead of the value option such as follows:

```
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 documentation 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 with something like the following YAML:

```
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:

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

See the documentation on secrets for more details.

Agent Temporary Volume

For added security, it is recommended that the `sigsci-agent` container be executed with the root filesystem mounted read only. The agent, however, 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. Typically this is just configured in the `volumes` section of a deployment.

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

Containers would then typically 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 Istio Service Mesh

Istio uses envoy proxy under its hood. Because of this, Istio can use the `sigsci-agent` in gRPC mode in the same you as with a generic envoy install. Installing and configuring the `sigsci-agent` are similar to a generic envoy install except the envoy proxy is automatically deployed as a sidecar. Envoy is then configured using Istio's `EnvoyFilter`. Full Istio integration is only possible in Istio v1.3 or later due to the required extensions to `EnvoyFilter`.

To add Signal Sciences support to an Istio based application deployment:

- 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
```

```
    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

Add the agent temp volume needs to be defined for use by the other containers in the pod. This just 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: {}
```

Adding the Istio EnvoyFilter Object to Inject the Required Envoy Config into the Istio Proxy

Istio has a feature rich way of customizing the envoy configuration for the `istio-proxy`. This is done via the `EnvoyFilter` object.

You will need to modify the `EnvoyFilter metadata.name` field and the `spec.workloadSelector.labels.app` field to be set to the application 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:
    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:
      name: "envoy.http_connection_manager"
      typed_config:
        "@type": "type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectio
        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:
                  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. Something like:

```
$ 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 pods
NAME                          READY   STATUS    RESTARTS   AGE
helloworld-7954bb57bc-pfr22   3/3     Running   2          33s
$ 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 4.0.0 starting as user sigsci with PID 1, Max open files=1048576,
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
```

Red Hat NGINX 1.10-1.14

Add the Package Repositories

Red Hat CentOS 8

Cut-and-paste 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

Cut-and-paste 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.

Cut-and-paste 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
```

Enabling Lua for NGINX

For older versions of NGINX, we require NGINX to be built with Lua and LuaJIT support. It is recommended to first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

The first step is to install the dynamic Lua NGINX Module appropriate for your NGINX distribution:

Red Hat distribution

NGINX 1.12.2 and higher

1. Next we will modify the `nginx.conf` (default `/etc/nginx/nginx.conf`) to load the dynamic Lua NGINX module. Directly below the line that starts with `pid` add:

   ```
   load_module /usr/lib64/nginx/modules/ndk_http_module.so;
   load_module /usr/lib64/nginx/modules/ngx_http_lua_module.so;
   ```

   An alternative option is to 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

   RHEL 7/CentOS 7 and higher

   ```
   systemctl restart nginx
   ```

   RHEL 6/CentOS 6

   ```
   restart nginx
   ```

Check that Lua is loaded correctly

To verify that Lua has been loaded properly load the following config(ex: 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
```

```
  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 successfully loading the 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 and Configure the Signal Sciences NGINX Lua Module

1. Install the Signal Sciences NGINX Lua module

   ```
   yum install sigsci-module-nginx
   ```

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

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

3. Restart the NGINX Service to initialize the new module

   RHEL 7/CentOS 7

   ```
   systemctl restart nginx
   ```

   RHEL 6/CentOS 6

   ```
   restart nginx
   ```

Site Alerts

Site alerts allow you to define thresholds for when to flag an IP address and how to treat subsequent requests from that IP.

How do system alerts work?

As requests with attack signals are sent to our backend, we track the number of signals that are seen from an IP across all agents.

Interval	Threshold	Frequency of Check
1 minute	50	Every 20 seconds
10 minutes	350	Every 3 minutes
1 hour	1,800	Every 20 minutes

When the number of malicious requests from an IP reaches one of these thresholds, the IP will be flagged and subsequent malicious requests will be blocked (or logged if your agent mode is set to “not blocking”) for 24 hours.

> **Note:** Requests containing only anomaly signals are not counted towards IP flagging thresholds.

How do site alerts work?

Site Alerts can be used to set lower or higher thresholds to alert and optionally block requests from an IP.

How do I configure a site alert?

If your role is User or above, configure a site alert by going to **Rules** > **Site Alerts** and clicking **New alert**.

Choose any attack or anomaly signal and set a threshold and interval for when to flag an IP. Once an IP is flagged, for attack signals, choose to either log subsequent requests or block subsequent malicious requests from that IP. Anomaly signals can only log subsequent requests.

What is the precedence of alerts?

The alert (either system or custom) with the lowest threshold and smallest interval for a given action (“block” or “log”) will be checked first. If an IP is flagged, it won’t be reflagged by any other alerts until that flag is lifted (in 24 hours).

> **Note:** “Blocking” and “logging” alerts are considered different types of alerts. This means that you can log (but not block) if Signal Sciences sees 25 SQLi in a minute, while we’ll still block subsequent requests from an IP if we see over 50 SQLi in a minute.

.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. Extract the contents of `sigsci-module-dotnet-x.x.x.zip` to your application’s `bin` directory.
2. 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>
   ```

3. Restart the web site service (recommended).

> **Note:** Make sure the agentEndPoint value is set to the same IP and port configured with the Signal Sciences agent’s rpc-address value. See details on configuring the Windows agent here.

.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	default	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: 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.

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 agent configuration file with any text editor:

   ```
   C:\Program Files\Signal Sciences\Agent\agent.conf

   accesskeyid = "AGENTACCESSKEYHERE"
   secretaccesskey = "AGENTSECRETACCESSKEYHERE"
   rpc-address = "127.0.0.1:737"
   ```

2. Configure the agent by inputting the Agent Access Key and Agent Secret Key into the `C:\Program Files\Signal Sciences\Agent\agent.conf`.

   - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

   - The Agent Access Key and Agent Secret Key will be visible within the window:

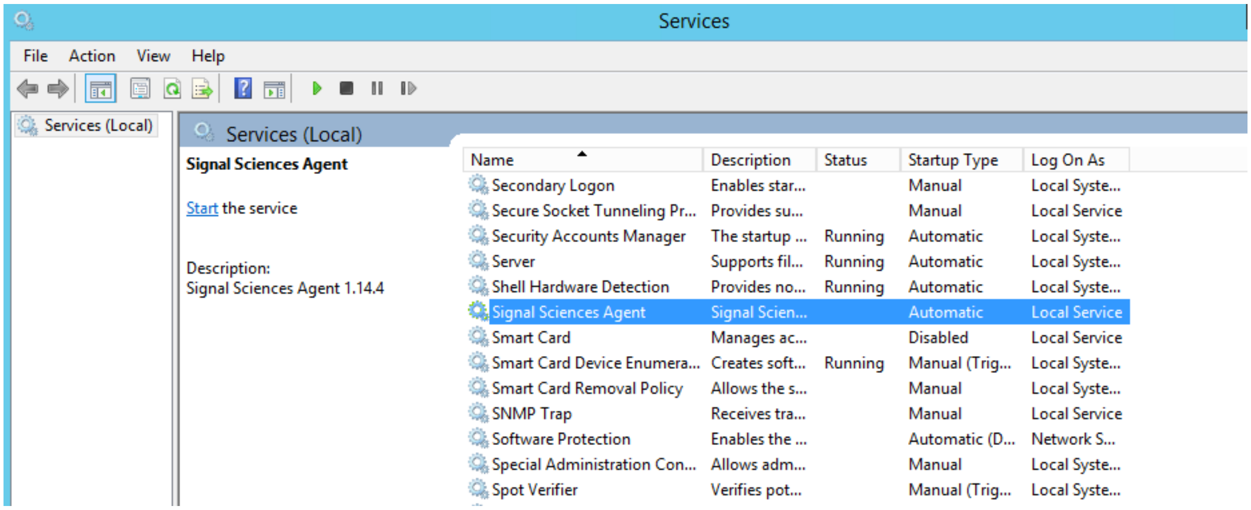
```
accesskeyid="[redacted]"
secretaccesskey="[redacted]"
```

Copy Cancel

- 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.

Additional configuration options are available on our Agent Configuration Page.

3. Download the latest Signal Sciences Windows Agent from https://dl.signalsciences.net/?prefix=sigsci-agent/.
   - Running the MSI will 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 and has valid accesskeyid and secretaccesskey settings.
   - The installed service name is `sigsci-agent` and can be controlled with PowerShell commandlets:
     - `Start-Service sigsci-agent`
     - `Restart-Service sigsci-agent`
     - `Stop-Service sigsci-agent`
   - The zip file contains the agent binary, which can be run from any location you prefer. Installing in this way requires the user 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

Install the Signal Sciences Module:

- Explore module options

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)

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
- Fixed outdated module detection
- Fixed handling of xml content type

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

Microsoft Teams

Our Teams integration allows you to be notified when certain activity occurs on Signal Sciences.

Adding Teams integration

1. Get started by adding a custom incoming webhook within Microsoft Teams: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/connectors#setting-up-a-custom-incoming-webhook

   > **Note:** Ensure that you copy the webhook URL to your clipboard from Microsoft Teams.

2. In Signal Sciences:

   - For a Corp Integration, navigate to **Corp Manage** > **Corp Audit Log** > **Manage corp integrations** > **Add corp integration** and select the **Microsoft Teams** integration.

   - For a Site Integration, navigate to **Manage** > **Site Integrations** > **Add site integration** and select the **Microsoft Teams** integration.

3. Enter the email address or alias you want notifications to be sent to.

4. Select if you want email notifications for all activity or specific activity.

5. Click **Add**.

6. Paste the webhook URL into the dialog box.

7. Click **Add**.

   Note: We will send all activity to Teams by default. To limit the integration to only send specific types of activity, select **Specific activity** and choose which activity types you'd like to trigger the integration.

Activity types

Corp

Activity type	Description
releaseCreated	New release notifications

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
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 was flagged
expireFlag	An IP 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

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

Response Codes

What is a “200” agent response code?

The Signal Sciences agent returns a “200” response code when a request is allowed through (similar to an `HTTP 200 OK` response).

What is a “406” agent response code?

By default, the Signal Sciences agent returns a “406” response code when a request is blocked (similar to an `HTTP 406 NOT ACCEPTABLE` response). You can configure rules to return alternative custom response codes other than 406 when a request is blocked.

What is a “499” agent response code?

A “499” response code indicates the client closed the connection mid-request.

What is an HTTP 504 response code?

A 504 response code is a timeout error which indicates that the gateway did not receive a response from the user’s upstream origin in the allotted time specified.

How are 504s and 499s related?

If a client is making a request and the Cloud WAF 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 longstanding request through a browser, will receive a 504 error in the browser, while the SigSci Console will show a 499 for the request.

Troubleshooting 504s correlated with 499s

The longstanding request will need to be optimized to meet the 60 second threshold. If the request cannot be optimized, reach out to our support team to explain the issue in detail and we will gladly help.

Relevant timeouts in the Cloud WAF architecture

- The Cloud WAF agent has 60 seconds to start sending a response to the Application Load Balance (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

What do “-2”, “-1”, or “0” agent response codes mean?

The `-2`, `-1`, and `0` response codes are error response codes that are applied to requests that weren’t processed correctly.

See the error response codes troubleshooting guide for additional information about these response codes.

Kubernetes Ambassador

Installing with Ambassador Edge Stack (AES)

This example illustrates the integration of Signal Sciences 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 just 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

Getting and Updating the Signal Sciences Agent Container Image

The official `signalsciences/sigsci-agent` container image available from the Signal Sciences account on Docker Hub is the recommended place to get the image. If you want to build your own image or need to customize the image, then follow the sigsci-agent build instructions.

The documentation references 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 this case you should specify a 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 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. To keep some consistency, 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, then just replace `latest` with the agent version. You may also want to change `imagePullPolicy: IfNotPresent` in this case as the image should not change.

```
- name: sigsci-agent
   image: signalsciences/sigsci-agent:4.1.0
   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 the `latest`), apply a custom tag, then use that custom tag in the configuration. You will want to specify `imagePullPolicy: Never` so that local images are only updated manually. 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 (ID and secret key)
- A volume to write temporary files

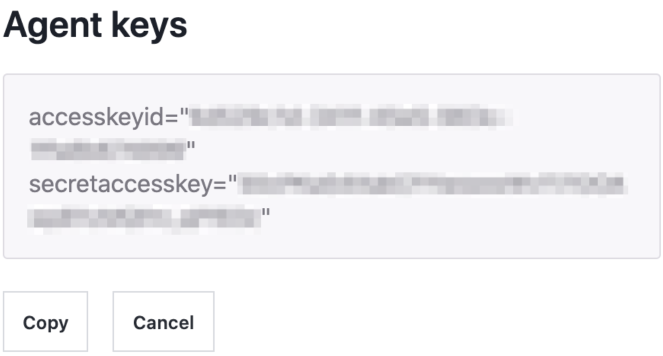
Agent Credentials

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

- SIGSCI_ACCESSKEYID: Identifies the site that the agent is configured against
- SIGSCI_SECRETACCESSKEY: The shared secret key to authenticate and authorize the agent

The credentials can be found by following these steps:

1. Log into the Signal Sciences console.
2. Click on **Agents**. The Agents page appears.
3. On the Agents page click **View Agent Keys**. The agent keys window appears.
4. Copy down the **Access Key** and **Secret Key** for later use.



Agent keys

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

Because of the sensitive nature of these values, it is recommended to use the builtin `secrets` functionality of Kubernetes. With this configuration, the agent will pull the values from the secrets data instead of reading hardcoded the 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.

Using secrets via environment variables is done using the valueFrom option instead of the value option such as follows:

```
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 documentation 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 with something like the following YAML:

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

See the documentation on secrets for more details.

Agent Temporary Volume

For added security, it is recommended that the `sigsci-agent` container be executed with the root filesystem mounted read only. The agent, however, 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. Typically this is just configured in the `volumes` section of a deployment.

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

Containers would then typically 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 (as of v4.5.0) 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 document is designed so the information 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 docs 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 some care must be taken during configuration to use the correct namespaces and 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 (Filter, FilterPolicy, LogService, Mapping, etc.)
- Used for the sigsci-agent when running as a sidecar

default

- Used for all applications and services running behind AES
- Used for the agent when run in standalone mode

Agent: 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 much easier if you are using Helm as this is directly supported in the Helm values file. Running as a sidecar has the distinct 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.

The agent must be configured with its ID and 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 docs 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 docs for installing with helm.

To install the agent as a sidecar, you should add the following to your custom values file, then install or upgrade AES with this values file. Refer to the Ambassador helm chart docs for a reference on the values file. This will add the container with the correct configuration to the AES pod as a sidecar.

Add 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 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
  - 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
  - name: sigsci-tmp
    mountPath: /sigsci/tmp
volumes:
- name: sigsci-tmp
  emptyDir: {}
```

Example upgrading AES with helm:

```
helm upgrade ambassador \
  --values /path/to/ambassador-sigsci_values.yaml \
  --namespace ambassador \
  datawire/ambassador
```

Alternatively 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 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

To sidecar the agent into the AES pod manually is a bit more involved. It is instead recommended to use Helm to render the manifests (see the Helm section above).

Refer to the AES installation guide for more details. You will need to modify the aes.yaml file (download here: https://www.getambassador.io/yaml/aes.yaml) and append the container and volumes described above in the helm docs to the ambassador deployment resource. Refer to the Kubernetes and envoy documentation for more details.

This is the correct resource to modify:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    product: aes
  name: ambassador
  namespace: ambassador
…
  containers:
  …
  volumes:
  …
```

The container will need to be added to the `containers` section and the volume to the `volumes` section.

Standalone

For a standalone agent, you just need to add a `Deployment` and `Service` resource for the agent such as follows. Refer to the Kubernetes and envoy documentation for more details.

Example SigSci Agent Service and Deployment:

```
apiVersion: v1
kind: Service
```

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

Sending Traffic to the Agent

Three Ambassador resources need to be configured for AES to send data to the agent. Refer to the envoy configuration docs for more detailed information on what each of these configures in the underlying Envoy install. The following documentation 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 (see below).

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`.

One item to note here is the namespace that needs to be used for the auth_service configuration. This is the namespace that the agent is deployed to. For this documentation we have used the `ambassador` namespace for sidecar agents and `default` namespace for standalone agents. The format for the auth_service URL should 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: ""
    tls: false
    proto: grpc
    include_body:
      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. This can be mapped 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 as handled the request.

Example routing all traffic to the agent:

```
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: {}
```

Routing subsets of traffic to the agent is possible with multiple rules. However every rule must go to the same agent due to 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
          arguments: {}
    - host: "*"
      path: "/app2/*"
      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 there was a signal detected. It is also used for anomaly signals such as HTTP_4XX, HTTP_5XX, etc.

agents. The format for the `service` URL should 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"
      during_request: false
      during_response: true
      during_trailer: true
    ### 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
```

Add the Package Repositories

Red Hat CentOS 8

Cut-and-paste 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

Cut-and-paste 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.

Cut-and-paste 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
```

Enabling Lua for NGINX

For older versions of NGINX, we require NGINX to be built with the third party `ngx_lua` module. As older versions of NGINX do not support dynamically loadable modules you would typically be required to rebuild from source.

To assist customers, we provide pre-built drop in replacements NGINX packages already built with the `ngx_lua` module. This is intended for customers who prefer not to build from source, or who either use a distribution provided package or an official NGNIX provided package. These pre-built packages are built to support much older distributions and are not gpg signed.

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 Red Hat based Linux distributions we've based them off the EPEL packages as neither Red Hat nor 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 Matrix of our NGINX replacement packages

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:

   Distro (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
   baseurl=https://yum.signalsciences.net/nginx/stable/el7/$basearch
   repo_gpgcheck=1
   gpgcheck=0
   enabled=1
   gpgkey=https://yum.signalsciences.net/nginx/gpg.key
   ```

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 sigsci repository:

```
yum -q makecache -y --disablerepo=* --enablerepo=sigsci_*
```

3. Install the Signal Sciences provided NGINX

```
yum install nginx
```

Yum repository setup for Red Hat and CentOS EL6 systems

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:

Distro (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
repo_gpgcheck=1
```

```
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 sigsci repository:

   ```
   yum -q makecache -y --disablerepo=* --enablerepo=sigsci_*
   ```

3. Install the Signal Sciences provided NGINX

   ```
   yum install nginx
   ```

Check that Lua is loaded correctly

To verify that Lua has been loaded properly load the following config(ex: 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
```

```
  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 successfully loading the 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 and Configure the Signal Sciences NGINX Lua Module

1. Install the Signal Sciences NGINX Lua module

   ```
   yum install sigsci-module-nginx
   ```

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

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

3. Restart the NGINX Service to initialize the new module

   RHEL 7/CentOS 7

   ```
   systemctl restart nginx
   ```

   RHEL 6/CentOS 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.)

Install

1. Download the latest SigSci HTTP middleware from https://dl.signalsciences.net/?prefix=sigsci-module-dotnetcore/, or get it via Nuget

2. Add the SigSci HTTP middleware to 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.

3. Add the following sections to your application's `appsettings.json` file:

```
{
    "SigsciOptions": {
        "AgentEndPoint": "127.0.0.1:2345"
    }
}
```

4. Configure HTTP request pipeline in Configure

```
Configure(IApplicationBuilder app, IHostingEnvironment env) {
        var sigsciOptions = Configuration.GetSection("SigsciOptions").Get<SigSciOptions>();
        app.UseSigSciHandler(sigsciOptions);
}
```

5. Restart the web site service (recommended).

   Note: Make sure the `AgentEndPoint` value is set to the same IP and port configured with the Signal Sciences agent's `rpc-address` value.

.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.

Sample advanced .NET Core module configuration:

```
{
        "SigsciOptions": {
                "AnomalySize": 200000,
                "AgentRPCTimeoutMillis": 200,
                "MaxPostSize": 50000,
                "AnomalyDurationMillis": 1000,
                "AgentEndPoint": "127.0.0.1:2345"
        }
}
```

Dotnet Core

SignalSciences .NET Core Module Release Notes

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

1.2.3 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

OpsGenie

Our OpsGenie issue integration creates an alert when IPs are flagged on Signal Sciences.

Adding a OpsGenie integration

1. Within OpsGenie, go to **Integrations**.
2. Find the **API** card and click the **Add** button.
3. Add your notification recipients and teams, and click **Save Integration**.
4. Copy the provided **API Key**.
5. On Signal Sciences, go to **Manage** > **Site Integrations**.
6. Click **Add site integration** and select the **OpsGenie Alert** integration.
7. Enter the **API Key** in the API key field.
8. Click **Add**.

Activity types

Activity type	Description
flag	An IP was flagged
agentAlert	An agent alert was triggered

Events

About events

Events are actions that Signal Sciences takes as the result of regular threshold-based blocking, templated rules, and site alerts.

Viewing Events

Events can be viewed on the Events page of the console by going to **Monitor** > **Events**.

Alternatively, a short list of the most recent Flagged IP events is available in the **Flagged IPs** dashboard card. Clicking on the **View** button for any Flagged IP in the list will take you to the Events page.

The Events page

The Events page of the console shows a historical record of all flagged IP addresses within the last 30 days. This page provides detailed information about the event associated with this IP address, including:

- A timeline illustrating the actions that occurred during the event. This includes:

  - When the IP was flagged.
  - How many requests were blocked or logged.
- A “Details” section providing additional, detailed information regarding the event. Depending on the nature of the attack, this can include the host, user agents, file paths, and country of origin.
- A “Sample Request” highlighting a single request received during the event, including the request itself and the signals applied to it. Clicking on **View this request** will take you to the request details page for that request.

This page also provides controls for managing IP addresses that have been flagged, including:

- Removing the IP address from the flag list.
- Creating request rules to allow specific IP addresses.
- Creating request rules to block specific IP addresses.

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:

```
lua_shared_dict sigsci_conf 12k;
lua_use_default_type off;
```

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.

Agent Scaling and Running as a Service

Scaling the Agent

If the `sigsci-agent` is installed as a sidecar into a pod, then 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 this case you may consider installing the agent as a service to be used by the application pods. Doing so, however, has some limitations and challenges of its own.

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 state 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, so load balancing agents requires affinity, which does make the service more complex to scale.
- Using the `sigsci-agent` as a service means configuring the communication channel as TCP vs a Unix domain socket and this is slightly less efficient.

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 the service, it must be configured so that the same agent will process both the request and response data for a transaction. To do this, you need to configure the service to use affinity based on the pod that is sending data to the agent. This is done by setting the affinity to use the Client IP.

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 would then be backed by a deployment with any number of replicas. The `sigsci-agent` container would be configured as in a typical sidecar install, but would use TCP instead of a shared Unix domain socket. This is done by setting the `SIGSCI_RPC_ADDRESS` configuration option. Note that for using this with envoy, you would 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.my-site-name
              key: secretaccesskey
        # Use RPC via TCP instead of default Unix Domain Socket
        - 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 can 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 would be defined by the service as well as the `SIGSCI_RPC_ADDRESS` (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 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/or 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, however. When scaling (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 is hit (`sessionAffinityConfig.clientIP.timeoutSeconds`). 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, etc. Similarly, when an agent pod is removed from the service due to scaling down, the application pods that were assigned 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

```
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.

Cut-and-paste 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 the module with yum

Then install the module by running the following command for your NGINX version:

NGINX+ 19

```
sudo yum install nginx-module-sigsci-nxp-1.17.3*
```

NGINX+ 18

```
sudo yum install nginx-module-sigsci-nxp-1.15.10*
```

NGINX+ 17

```
sudo yum install nginx-module-sigsci-nxp-1.15.7*
```

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

RHEL 6/CentOS 6

```
restart nginx
```

RHEL 7/CentOS 7

```
systemctl restart nginx
```

SOC 2

Signal Sciences has completed our SOC 2 Type II audit of the company's operational and security processes for our service. Signal Sciences will continue to undergo a regular third-party audit to certify our services against this standard.

What is SOC 2?

SOC 2 is a report based on AICPA's existing Trust Services principles and criteria. The purpose of the SOC 2 report is to evaluate an organization's information systems relevant to security, availability, processing integrity, and confidentiality or privacy.

How can I obtain the SOC 2 report?

Prospects can request the report through a sales representative. Customers can request the report through a support ticket.

GDPR

Signal Sciences is aligned with GDPR.

What is GDPR?

The General Data Protection Regulation (GDPR) is a regulation in EU law on data protection and privacy for all individuals within the European Union. It aims to give control back to EU residents over their personal data.

Who does GDPR apply to?

GDPR applies to any organization handling personal data of an EU resident, regardless of where it is based.

What is personal data?

GDPR defines "personal data" very broadly. By definition, personal data includes information relating to an identifiable person who can be directly or indirectly identified in particular by reference to an identifier. Common examples of "personal data" include name and address. However, GDPR's definition also includes, but is not limited to, log-in credentials, IP addresses, and cookies.

How does GDPR apply to Signal Sciences' services?

While Signal Sciences' services are not intended to process highly sensitive personal information, Signal Sciences is subject to GDPR as we process information regarding our customers, which may include personal data of EU residents (i.e. IP addresses).

How has Signal Sciences prepared for GDPR?

Signal Sciences is committed to being aligned with GDPR with respect to the services we provide and the client data we process. We have worked to build features that give customers more control over their data, like IP anonymization and data redactions. We have also updated our privacy policy to provide more transparency to our customers on how we intend to use their data.

How can Signal Sciences assist customers in meeting their obligations under GDPR?

Signal Sciences ("Processor") can assist customers ("Controllers") in fulfilling their obligations as data controllers by:

- supporting customers in complying with requests from Data Subjects
- maintaining security best practices for safeguarding personal data
- providing a list of our sub-processors, upon request

If you have any requests related to the above, please reach out to support.

How can Signal Sciences help address requests from Data Subjects?

Signal Sciences has implemented IP anonymization as a product feature to give customers more control over personal data. Please refer to IP anonymization for guidance on how to enable IP anonymization.

If you have any other requests from Data Subjects, please reach out to support.

Where can I learn more about security and privacy efforts?

Signal Sciences' privacy policy can be referenced here: https://www.fastly.com/privacy/

Does Signal Sciences have a Data Processing Agreement (DPA) for their customers?

Yes, Signal Sciences has a standard DPA for all new contracts. If you are a current customer and need a DPA, please reach out to support.

Who are the sub-processors authorized to process customer data for signal sciences services?

Signal Sciences engages certain sub-processors in connection with the provision of the Solution. A sub-processor is a third-party service provider engaged by Signal Sciences to process personal data on behalf of Signal Sciences's customers.

Entity Name	Entity Type	Entity Location
Amazon Web Services, Inc.	Third-party sub-processor	United States
MongoDB Atlas	Third-party sub-processor	United States

Python Module Install

Compatibility

Module is compatible with latest Python 2.X and 3.X

Installation via pip

```
pip install https://dl.signalsciences.net/sigsci-module-python/sigsci-module-python_latest.tar.gz
```

Add to Flask Application

1. Reference the module in `setup.py`

```
# Reference module in `setup.py`
packages = ['flask', '......', 'sigscimodule']
```

2. Import the `sigscimodule` and apply Middleware in `app.py`

```
# Import sigsci module
from sigscimodule import Middleware

# To apply SignalSciences middleware, wrap the application object
app.wsgi_app = Middleware(app.wsgi_app)
```

Java

Java Module Release Notes

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

2.2.1 2020-09-9

- Improved logging when module fails to communicate to the agent

2.2.0 2020-08-17

- 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 unncessary 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

1.1.2 2019-02-19

- Added ability for Java module to work without any dependencies
- 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

1.1.0 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 sigsci 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

0.1.4 2016-09-20

- Changed it to send all headers to agent for inspection

0.1.3 2016-09-19

- 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

PagerDuty

Our PagerDuty issue integration creates an incident when IPs are flagged on Signal Sciences.

Adding a PagerDuty integration

PagerDuty issue integrations are configured per project.

1. Within PagerDuty, go to **Configuration > Services**.
2. Click **Add New Service** button.
3. Name your new service, and set your escalation policy and integration method to **Use Our API Directly**.
4. Click **Add Service** and copy the newly created **Service API Key**.
5. On Signal Sciences, go to **Manage** > **Site Integrations**.
6. Click **Add site integration** and select the **PagerDuty Trigger** integration.
7. Enter the **Service API Key** in the Service key field.
8. Click **Add**.

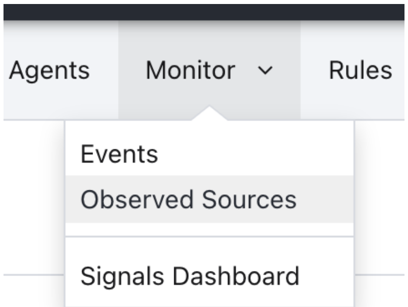
Activity types

Activity type	Description
flag	An IP was flagged
agentAlert	An agent alert was triggered

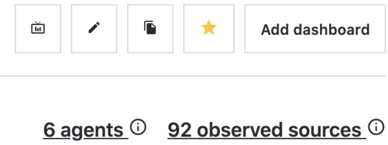
Observed Sources

The Observed Sources page provides an overview of all IPs that have been—or soon will be—flagged on your site. The Observed Sources page is divided into separate tabs for Suspicious IPs, Flagged IPs, and Rate Limited Sources.

The Observed Sources page can be viewed by going to **Monitor** > **Observed Sources** in the console:



The Observed Sources page can also be viewed by clicking on the combined count of all Observed Sources on the site overview page:



Suspicious IPs

The Suspicious IPs tab shows IP addresses from which requests containing attack payloads have originated, but the volume of attack traffic from these IPs has not exceeded the decision threshold. Once the threshold is met or exceeded, an IP address will be flagged and added to

- The IP
- Country of origin
- The signal for which the IP is approaching a threshold
- The threshold being approached
- How long ago the IP was added to the Suspicious IPs list
- If the IP was flagged by another Signal Sciences customer

Clicking on an IP in the Suspicious IPs list will take you to a Requests page search for that IP.

Flagged IPs

The Flagged IPs tab shows all IP flagging events. IP addresses can be flagged through regular threshold-based blocking, templated rules, and site alerts.

The Flagged IPs tab lists:

- The IP
- Country of origin
- The signal the IP was flagged on
- How long ago the IP was flagged
- If the IP is still currently flagged

Clicking on an IP in the Flagged IPs list will take you to a Requests page search for that IP.

Rate Limited Sources

> **Note:** Rate Limit rules are only included with the Premier platform. They are not included as part of our Professional or Essential platforms.

The Rate Limited Sources tab shows all sources that have been rate limited.

The Rate Limited Sources tab lists:

- The source
- The signal the source was rate limited on
- When the source will stop being rate limited

This page also provides controls for managing sources that have been rate limited, including:

- Removing all entries from the rate limited sources list
- Removing specific sources from the rate limited sources list
- Creating request rules to allow specific sources
- Creating request rules to block specific sources

Custom Response Codes

> **Note:** Custom Response Codes are not supported on the Essential platform.

Custom response codes allow you to specify which HTTP status code is returned by Signal Sciences when a request is blocked. By default, Signal Sciences will return a 406 response code when a request is blocked. With custom response codes enabled on a rule, you can select an alternative response code to be returned instead of 406.

Custom response codes can facilitate additional actions at the edge depending on the rule triggered. For example, a specific custom response code can be used to tell your CDN to redirect the request to a CAPTCHA. The Fastly CDN supports custom response codes in VCL to redirect requests to other pages, such as custom error pages.

Limitations

- Custom response codes can only be set on individual rules that block requests.
- Each site may have up to 5 unique response codes across all rules at any time.
- There is no limit to the total number of rules that use custom response codes.
- Custom response codes require a minimum agent and module version.
- Custom response codes are limited to numbers between 400 and 499.

What happens when a rule with the default response code and a rule with a custom response code both block a request?

The request is blocked and the custom response code is returned.

What happens when two rules with different custom response codes both block a request?

The request is blocked and the oldest custom response code is returned, based on when the response codes were first created.

For example, if Rule A had a custom response code created one week ago and Rule B had a custom response code created yesterday, the custom response code of Rule A would be used because that response code was created earlier.

How to set a custom response code

When creating or editing a rule:

1. From the **Action type** menu, select **Block**.
2. Beneath the **Action type** menu, click **Change response**. The **Response code (optional)** field appears.
3. In the **Response code (optional)** field, enter the custom response code to return when the rule blocks a request. You can only use codes between 400 and 499.
4. Click **Create site rule** or **Update site rule** at the bottom of the rule editor.

Minimum version support

The following agent and module versions support custom response codes:

Name	Minimum Version
Agent	4.10+
Apache	1.8.0+
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+
PHP	2.0.0+
Python	1.3.0+

> **Note:** If an unsupported module version is told to block a request due to a rule that uses a custom response code, that request will **not be blocked**.

Unsupported agents and modules handle requests that should be blocked by rules with custom response codes in the following ways:

Agent	Module	Result
Supported	Supported	Blocked with custom 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 response code
Unsupported (Reverse Proxy)	N/A	Blocked with default response code of 406

Pivotal Container Services (PKS) Setup

RASP detects and stops web layer attacks, maintains site reliability, and provides visibility that empowers DevOps teams, all at a lower cost of ownership than legacy WAF solutions. Read our blog post for more information about the Pivotal PKS and Signal Sciences partnership.

Getting Started

- https://tanzu.vmware.com/kubernetes-grid
- https://docs.pivotal.io/pks/

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

Setup the Environment

First setup your environment.

```
# Credentials filename
export KUBECONFIG=pks-creds.yaml
```

Login to PKS

You then need to login to PKS using your URL and username/password.

```
pks login -a <your-url> -u <user> -p <password> -k
```

Create the Credentials File

You then create the credentials file (from KUBECONFIG).

```
pks get-credentials <cluster-name>
```

Set the Context to the Remote Cluster

Set the context to the remote cluster so that all local commands are run on that remote cluster.

```
kubectl config use-context <cluster-name>
```

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
```

To set up Signal Sciences, follow the normal Kubernetes instructions to integrate Signal Sciences into a Kubernetes pod.

Debian NGINX 1.14.1+

Add the Package Repositories

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Debian 10 “buster”

Cut-and-paste 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/ buster main
EOF
sudo apt-get update
```

Debian 9 “stretch”

Cut-and-paste 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
```

Debian 8 “jessie”

Cut-and-paste 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”

Cut-and-paste 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 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.

Then install the 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*
```

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

```
sudo service nginx restart
```

PHP Module Install

> **Note:** Our PHP module requires a minimum version of PHP 5.3.

Installing the PHP module with PEAR

1. Download the Signal Sciences PHP PEAR module package

```
curl -O -L https://dl.signalsciences.net/sigsci-module-php/sigsci-module-php_latest.tgz
pear install sigsci-module-php_latest.tgz
```

2. Validate your PHP include_path

Once you download the PEAR package, it will be installed in the default location configured by your system, commonly `/usr/share/php`. The following command will help you identify the PEAR directory:

```
pear config-show | grep "PEAR directory"
```

Next, we will check the PHP configuration for the PEAR directory’s existence within the `include_path`:

```
php -i | grep include_path
```

> **Note:** If the PEAR directory is not within your PHP `include_path`, the next step would be to modify your PHP file.

3. Include the module within your application

```
require_once('SigSci/sigsci.php');
```

4. Use the module within your application

After installing the module you'll need to call the `SigSciModule` class, and capture the response to make a decision on whether or not to block. Learn more about configuring and using the PHP module here.

Installing the PHP module from a tarball

1. Download the Signal Sciences PHP module tarball

```
curl -O https://dl.signalsciences.net/sigsci-module-php/sigsci-module-php_latest.tar.gz
```

2. Extract the PHP module

After downloading the package, extract it to the current directory:

```
tar -xvzf sigsci-module-php_latest.tar.gz
```

3. Include the module within your application

After you extract the PHP package, you will need to include it within your application. Depending on your application structure, you may want to move the `msgpack.php` and `sigsci.php` files to a new directory.

Once the module is within your application tree, you will need to require it at the beginning of your application.

```
require_once('sigsci.php');
```

4. Use the module within your application

After installing the module you'll need to call the `SigSciModule` class, and capture the response to make a decision on whether or not to block. Learn more about configuring and using the PHP module here.

Using the PHP Module

The Signal Sciences PHP module class is named `SigSciModule`. This module contains several methods used for communicating with the Signal Sciences agent in addition to the following methods which the customer can safely access:

```
__construct()
block()
agentResponseCode()
agentRequestID()
agentMeta()
agentTags()
preRequest()
postRequest()
```

Basic Usage

To get started, we will call the `SigSciModule` class:

```
$sigsci = new SigSciModule();
$sigsci->preRequest(); // Gathers request details for the agent
if ($sigsci->block()){
http_send_status(406);
echo "Invalid Request Detected";
$sigsci->postRequest();
exit();
}

// Your application code ....
$sigsci->postRequest();
?>
```

Once `$sigsci->preRequest()` has completed, you will now have access to `$sigsci->block()`.

Detected attack types such as `SQLI` and `XSS`, which are returned to the module from the agent, can be pulled by calling the `$sigsci->agentTags()` method.

You will also need to add `$sigsci->postRequest()` to the end of the application. If your application exits anywhere in your application code, you should make the `$sigsci` object available to that calling method to call `$sigsci->postRequest()`.

Simplified Configuration

To simplify this process and to take advantage of the `__construct()` and `__destruct()` magic methods, instantiate the `SigSciModuleSimple()` class, which extends `SigSciModule()` and automatically calls `preRequest` and `postRequest` within `__construct()` and `__destruct()` respectfully.

This simplifies implementation into the following snippet:

```
block()){
http_send_status(406);
echo "Invalid Request Detected";
exit();
}

// Your application code ....
?>
```

Advanced Configuration

We provide the ability to configure the module via an `array()`. The following attributes are set by default, but may need to be modified to provide support for different environments.

```
$config = array(
'max_post_size' => 100000, /* ignore posts bigger than this */
'timeout_microseconds' => 500000, /* fail open if agent calls take longer than this */
'socket_domain' => AF_UNIX, /* INET or UNIX */
'socket_address' => "/tmp/sigsci-lua",
'socket_port' => 0,
'allowed_methods' => array("GET", "POST", "PUT", "DELETE", "PATCH"),
'body_methods' => array("POST", "PUT", "PATCH"),
'filter_header' => array("cookie", "set-cookie", "authorization", "x-auth-token"), /* headers never sent to agent
'anomaly_size' => 524288, /* if output is bigger size than this, send info to SigSci */
'anomaly_duration' => 1000, /* if request length is greater than this (millisecond), report back */
);
```

For example, on a SystemD-based system, the socket cannot run in `/tmp/sigsci-lua`. As a result, we need to update the agent configuration to point to `/var/tmp/sigsci-lua`. To ensure the module can communicate with the agent, we must match the socket during module instantiation.

```
$sigsci_conf = array('socket_address' => '/var/tmp/sigsci-lua');
$sigsci = new SigSciModuleSimple($config);
```

Next Steps

- Verify Agent and Module Installation

Explore other installation options:

- Explore module options

Upgrading the Agent

> Check the Agent Changelog to see what's new in the agent.

Our Agent 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-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 and higher:

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

Upgrading the Agent on Red Hat-CentOS systems

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

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`

Heroku

SignalSciences Buildpack for Heroku Release Notes

0.2.1 2020-11-09

- Added server-flavor option to distinguish buildpack.

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`

0.1.11 2020-05-19

- Fixed upstream URL

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

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.

3. Within the Signal Sciences console, go to **Manage** > **Site Integrations**.
4. Click **Add site integration** and select the **PivotalTracker Story** integration.
5. Enter the **Project ID** and **API Token** into their respective fields and choose your preferred triggers for this event.
6. Click **Add**.

Activity types

Activity type	Description
flag	An IP was flagged
agentAlert	An agent alert was triggered

Search Syntax

Free Text

In many cases, you can "just type" a free-text query.

example	description
`/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

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)

Absolute time

Absolute time is also allowed using

- 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, example: 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`), or IPv4 range (`ip:198.51.100.0..198.51.100.255`)
`method`	string	HTTP Method, example: 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

`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, example: "example.com" if http://example.com/name
`tag`	string	A particular signal on a request, example: SQLI, XSS, etc.
`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)

AWS Elastic Container Service (ECS) Setup

Introduction

This article 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.

Instructions

1. Create a new task definition. Select either **Fargate** or **EC2**.

2. Add the Shared Volume for the containers to use for the Unix Socket file by clicking **Add volume** under "Volumes".

3. In the **Add Volume** window, enter a name, select the type of **Bind Mount**, and click **Add**.

4. On the main Task page, click **Add Container**.

5. Specify the following:

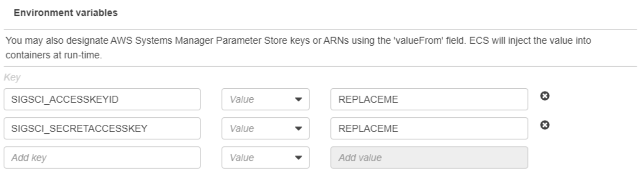
   - **Container Name:** This can be any Display Name you would like, such as "example-app".
   - **Image:** This will be the Docker Image, i.e. `username/example-app:latest`.
   - **Port Mappings:** Add any ports that should be available for the App.

6. Scroll down in the Container Definition to **Storage and Logging** and select the volume created earlier in the **Mount Points**.

7. Create the container.

8. Add a second container for the Signal Sciences Agent:

   - **Container Name:** `sigsci-agent`
   - **Image:** `signalsciences/sigsci-agent:latest`
   - **Port Mappings:** Add any ports that should be available for the App.
   - **Environment Variables:**
     - Enter the Agent Access Key and Agent Secret Key for your site as environment variables named `SIGSCI_ACCESSKEYID` and `SIGSCI_SECRETACCESSKEY`:



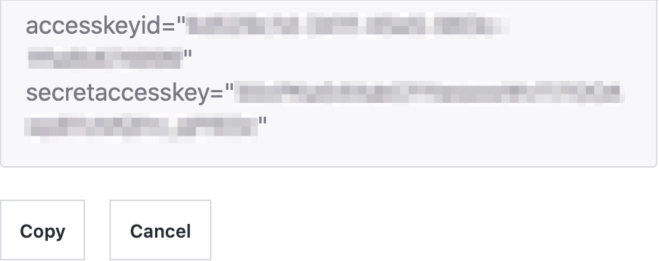
Environment variables

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

     - The Agent Access Key and Agent Secret Key for your site are listed within the Signal Sciences console by going to **Agents** > **View agent keys**:

     - The Agent Access Key and Agent Secret Key will be visible within the window:



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

  - **Mount Points:** Select the same mount point as the previous Container.

9. Create the container

10. Finish creating the task definition

11. After the task definition is created, in 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` and `REPLACE_ME` 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"
                }
            },
            "entryPoint": null,
            "portMappings": [
                {
                    "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"
                }
            ],
            "ulimits": null,
            "dnsServers": null,
            "mountPoints": [
                {
                }
            ],
            "workingDirectory": null,
            "secrets": null,
            "dockerSecurityOptions": null,
            "memory": null,
            "memoryReservation": null,
            "volumesFrom": [],
            "stopTimeout": null,
            "image": "trickyhu/sigsci-apache-alpine: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": {
                    "awslogs-group": "/ecs/sigsci-example",
                    "awslogs-region": "us-west-1",
                    "awslogs-stream-prefix": "ecs"
                }
            },
            "entryPoint": null,
            "portMappings": [],
            "command": null,
            "linuxParameters": null,
            "cpu": 0,
            "environment": [
                {
                    "name": "SIGSCI_ACCESSKEYID",
                    "value": "REPLACEME"
                },
                {
                    "name": "SIGSCI_SECRETACCESSKEY",
                    "value": "REPLACEME"
                }
            ],
            "ulimits": null,
            "dnsServers": null,
            "mountPoints": [
                {
                }
            ],
            "workingDirectory": null,
            "secrets": null,
            "dockerSecurityOptions": null,
            "memory": null,
            "memoryReservation": null,
            "volumesFrom": [],
            "stopTimeout": null,
            "image": "trickyhu/sigsci-agent-alpine: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"
        }
    ],
    "memory": "4096",
    "taskRoleArn": "arn:aws:iam::REPLACEME:role/EcsServiceRole2",
    "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

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

```
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/ buster main
EOF
sudo apt-get update
```

Debian 9 “stretch”

Cut-and-paste 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”

Cut-and-paste 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”

Cut-and-paste 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
```

Enabling Lua for NGINX

For older versions of NGINX, we require NGINX to be built with Lua and LuaJIT support. It is recommended to first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

Nginx.org distribution

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

Debian distribution

Enable Lua by installing the `nginx-extras` package with the following command:

Check that Lua is loaded correctly

To verify that Lua has been loaded properly load the following config(ex: 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
# 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 successfully loading the 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 and Configure the Signal Sciences NGINX Module

1. Install the module

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

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

Debian 8 and 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:

- Counters are counts since last update
- Gauges are point in time or lifetime metrics

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 decision time 50th percentile
sigsci.agent.waf.perf.decision_time.95pct	gauge	The decision time 95th percentile
sigsci.agent.waf.perf.decision_time.99pct	gauge	The decision time 99th percentile
sigsci.agent.waf.perf.queue_time.50pct	gauge	The queue time 50th percentile
sigsci.agent.waf.perf.queue_time.95pct	gauge	The queue time 95th percentile
sigsci.agent.waf.perf.queue_time.99pct	gauge	The queue time 99th percentile
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

Installation

Install the Agent

Install the Signal Sciences Agent by following the instructions for your environment here: https://docs.signalsciences.net/install-guides/agent-installation/agent-install-intro/

Download Prerequisites

The Golang module requires two prerequisite packages to be installed: MessagePack Code Generator and the Signal Sciences custom tlstext package.

The easiest way to install these packages is by 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 Golang Module

Download the latest version of the Golang module:

```
curl -O -L https://dl.signalsciences.net/sigsci-module-golang/sigsci-module-golang_latest.tar.gz
```

```
sudo mkdir -p $GOPATH/src/github.com/signalsciences
sudo tar -xf sigsci-module-golang_latest.tar.gz -C $GOPATH/src/github.com/signalsciences
```

Wrapping 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 by adding this line:

   ```
   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:

   ```
   muxname.HandleFunc(“/hello“, hellofunc)
   muxname.HandleFunc(“/example”, examplefunc)
   ```

4. Wrap your ServeMux in the Signal Sciences Golang Module by adding these 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()
```

```
	wrappername, err := sigsci.NewModule(muxname)
	if err != nil {
		log.Fatal(err)
	}

	http.ListenAndServe("127.0.0.1:80", wrappername)
}
```

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

1.0.0 2016-08-01

- Initial release

Slack

Our Slack message integration allows you to be notified when certain activity occurs on Signal Sciences.

Adding a Slack message integration

1. If you don’t currently have any incoming webhooks set up, go to https://my.slack.com/services/new/incoming-webhook/.
    - If you have already set up incoming webhooks, then within the main menu of Slack, go to **Configure Apps**.
    - Go to **Custom Integrations**, and click on the **Incoming WebHooks** integration.
    - Click on **Add Configuration**.
2. Choose the channel you want the webhook to post in and create the webhook.
3. Copy the **webhook URL** on the following page.
4. In Signal Sciences:
    - For a Corp Integration, navigate to **Corp Manage** > **Corp Audit Log** > **Manage corp integrations** > **Add corp integration** and select the **Slack Message** integration.
    - For a Site Integration, navigate to **Manage** > **Site Integrations** > **Add site integration** and select the **Slack Message** integration.
5. Enter the email address or alias you want notifications to be sent to.
6. Select if you want email notifications for all activity or specific activity.
7. Click **Add**.
8. Paste the webhook URL into **Webhook URL** and choose which types of activity you want to trigger the webhook.
9. Click **Add**.

Activity types

Corp

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
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 was flagged
expireFlag	An IP 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

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

Corp Management

Signal Sciences provides you a set of tools, depending on your permission level, to easily manage sites, users, and members in your corporation.

Glossary

1. **Corporation:** A corporation is a set of sites and users. Users are authenticated against a corporation and can be members of different sites in that corporation.
2. **Site:** Sites belong to a corporation and consist of a set of requests and configurations. Requests come from agents configured with the site's access and secret keys. Configurations include agent rules (e.g., tagging requests as XSS, blocklist and allowlist rules, blocking rules, etc.), the list of members, integrations, and other configuration options.

- Logically think of a site as a mapping to a particular application or domain (e.g., `app1.example.com` vs. `app2.example.com`), but you could have multiple apps share the same site keys, or split one app into different sites (e.g., example.com and example.com/admin).

1. **User:** A user belongs to a particular corporation and is identified by an email address and password. A user can be a member of one or more site.
2. **Member:** A member is a user's membership in a particular site.

How do permissions work?

A user has a role of either Owner, Admin, User, or Observer:

1. **Owners** have access to all corp features, can edit settings on every site, and can make changes to user accounts.
2. **Admins** have limited access to corp features, access to specific sites and site-level settings, and can invite new users to specific sites.
3. **Users** have access to specific sites and site-level settings.
4. **Observers** have access to specific sites.

	Owner	Admin	User	Observer
Corp Management				
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				
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				
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 Display Name and Short Name	All sites	Specific sites	Specific sites	No sites
Site Configurations				
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
Create or edit redactions	All sites	Specific sites	Specific sites	No 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				
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

Corp management

Owner users can manage the sites and users of their corporation.

Site management

The Site Management page enables you to add, remove, and edit sites on your corp. This page lists all the sites in your corporation, along with their agent mode and number of members. To access the Site Management page:

1. Log into the Signal Sciences console.
2. From the **Corp Manage** menu, select **Sites**. The Site Management page appears.
3. Under the **Corp Manage** menu, click **Sites**.

Adding a site

To add a site, click **New site**. Choose a display name, a short name to be used in the URL, and the agent mode. Once you’ve added the site, set up the agent and module by following the installation process.

> **Note:** By default, your corporation has a limited number of sites. If you need more, contact support for assistance.

Editing a site

Edit any site by clicking the pencil icon to the far right of the site. The site configuration page allows you to:

- Change the display name
- Change the short name
- Change the agent mode
- Toggle IP anonymization

Deleting a site

A site can be deleted by selecting the **Delete** button next to the site. Only Owners have the ability to delete sites.

A site cannot be deleted if it:

- Is the current active console
- Is the last site remaining for the corp
- Has users that aren’t members of any other sites

> **Note:** If you would like to delete a site meeting any of the conditions listed above, reach out to our support team.

Removing an agent

Once an agent has been offline for 3 days, it will disappear from the agents list automatically.

Under the **Corp Manage** menu, click **Manage Users**. This page lists all the users in your corporation, along with their roles, site memberships, and whether they have 2FA enabled, as well as the list of pending invited users.

Adding a user

Click the **Add user** button. Enter their email and choose a role and site memberships.

> **Note:** A user must belong to at least one site.

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 or deleting a user

Click the user's row to change their role as well as delete the user from the corporation.

Resetting 2FA for a user

To reset 2FA for a user, click the pencil icon next to the user. Click the **Disable** button next to their 2FA status. The user will then be able to sign into their account and reconfigure 2FA.

Auditing two-factor authentication

Audit two-factor authentication (2FA) usage via the "2FA" column in the users list. We don't currently support 2FA enforcement.

Single sign-on

See Single Sign-On for more information about enabling Single Sign-On.

Bypassing SSO

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.

Select **Allow this user to bypass Single Sign-On (SSO)** to set the user to bypass SSO.

API Access Tokens

See Using Our API for information about personal API access tokens.

Assigning or removing a user from a site

Admins

Assign a user to a site by navigating to that specific site, clicking **Manage** > **Site Settings** from the navigation, and selecting the **Users** tab. From there, click **Manage site users** and select either **Invite new user** to invite an entirely new user or **Assign existing users** to choose an existing user in the corp. If the user doesn't already belong to the corp they'll be provisionally added to the site and receive an invitation email to join your corp.

Owners

In addition to the method described above, Owner users can also assign users by going to **Corp Manage** > **Corp Users** in the navigation bar at the top. On that page, Owners can select specific sites from the dropdown menu on the left and assign users to that site by clicking **Assign existing users to this site**. Alternatively, Owners can select a User's row, and from the User Edit page, select which sites that user should be assigned to manage. In this case, they will have their same role across every site membership.

For more information on member roles see How do permissions work?

Console Timeout

The default duration for a validated session is 30 days. To set a custom duration your corp:

1. Log into 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.

Example Helloworld Test Web Application

Helloworld Test Web Application

Files

Dockerfile

Dockerfile to build the `signalsciences/example-helloworld` container:

```
docker build . -t signalsciences/example-helloworld:latest
```

```
FROM golang:1.13

# Image metadata
LABEL com.signalsciences.sigsci-module-golang.examples="helloworld"
LABEL maintainer="Signal Sciences <support@signalsciences.com>"

# 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
```

```
        # Address for the app to listen on
        - localhost:8000
        ports:
        - containerPort: 8000
```

Debian NGINX 1.9 or lower

Add the Package Repositories

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Debian 10 "buster"

Cut-and-paste 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/ buster main
EOF
sudo apt-get update
```

Debian 9 "stretch"

Cut-and-paste 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"

Cut-and-paste 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"

Cut-and-paste 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
```

Enabling Lua for NGINX

For older versions of NGINX, we require NGINX to be built with the third party `ngx_lua` module. As older versions of NGINX do not support dynamically loadable modules you would typically be required to rebuild from source.

To assist customers, we provide pre-built drop in replacements NGINX packages already built with the `ngx_lua` module. This is intended for customers who prefer not to build from source, or who either use a distribution provided package or an official NGNIX provided package.

Flavors of our NGINX replacement packages

modules.

Our provided flavors are:

- **distribution** - The distribution flavor is based off the official distribution provided NGINX packages. For Debian-based Linux distributions (Ubuntu 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 relases.

Flavor Version Matrix of our NGINX replacement packages

The following versions are contained in the various OS and flavor packages:

OS	Distribution	Stable	Mainline
Debian 7 (Wheezy)	1.2.1	1.8.1	1.9.10
Debian 8 (Jessie)	1.6.2	1.8.1	1.9.10

The versions are dependent on the upstream package maintainer’s supported version.

Apt repository setup for Debian systems

To configure the apt repository on your Debian system:

1. Add our 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
Debian 7 (Wheezy)	`deb https://apt.signalsciences.net/nginx/distro wheezy main`
Debian 8 (Jessie)	`deb https://apt.signalsciences.net/nginx/distro jessie main`

   Stable Flavor

OS	sigsci-nginx.list content
Debian 7 (Wheezy)	`deb https://apt.signalsciences.net/nginx/stable wheezy main`
Debian 8 (Jessie)	`deb https://apt.signalsciences.net/nginx/stable jessie main`

   Mainline flavor

OS	sigsci-nginx.list content
Debian 7 (Wheezy)	`deb https://apt.signalsciences.net/nginx/mainline wheezy main`
Debian 8 (Jessie)	`deb https://apt.signalsciences.net/nginx/mainline jessie 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 Signal Sciences NGINX

   ```
   sudo apt-get install nginx
   ```

Check that Lua is loaded correctly

To verify that Lua has been loaded properly load the following config(ex: sigsci_check_lua.conf) with nginx:

```
  # Config just to test for lua jit support
#
# Test from commandline as follows:
```

```
# 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 successfully loading the 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 and Configure the Signal Sciences NGINX Module

1. Install the module

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

3. Restart the NGINX Service to initialize the new module

   Debian 8 and higher

   ```
   sudo systemctl unmask nginx && sudo systemctl restart nginx
   ```

   Debian 7

   ```
   sudo service nginx restart
   ```

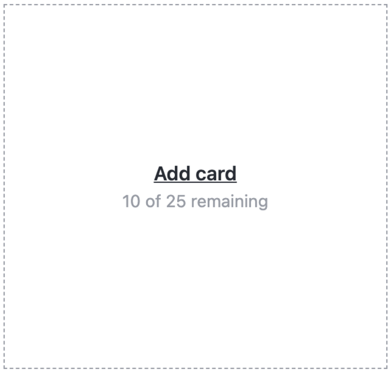
Overview Page

Customizable Overview Page

Signal Sciences provides the ability to customize the overview page experience. These customizations include creating and arranging cards into a preferred layout, as well as editing custom cards to display specific signals. Create multiple dashboards to easily switch between saved arrangements of cards and signals.

Adding a new Card

To add a new card, scroll to the bottom of the Overview Page and select "Add card".



Add the title of the new card, a description and begin adding signals by using either the dropdown list or typing in the name of the signal. Save the card and it will now show on the overview page.

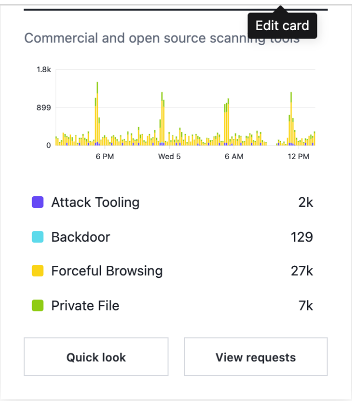
Resetting the overview page

To reset the overview page back to its original settings, click on "Edit dashboard" in the top right corner, and click "Reset Dashboard".

> **Note:** By resetting the dashboard all previous customizations will be deleted.

Edit an existing card

To edit an existing card, hover over the card and the top right corner of the card will display **Edit**. Click **Edit** to change the parameters as necessary.

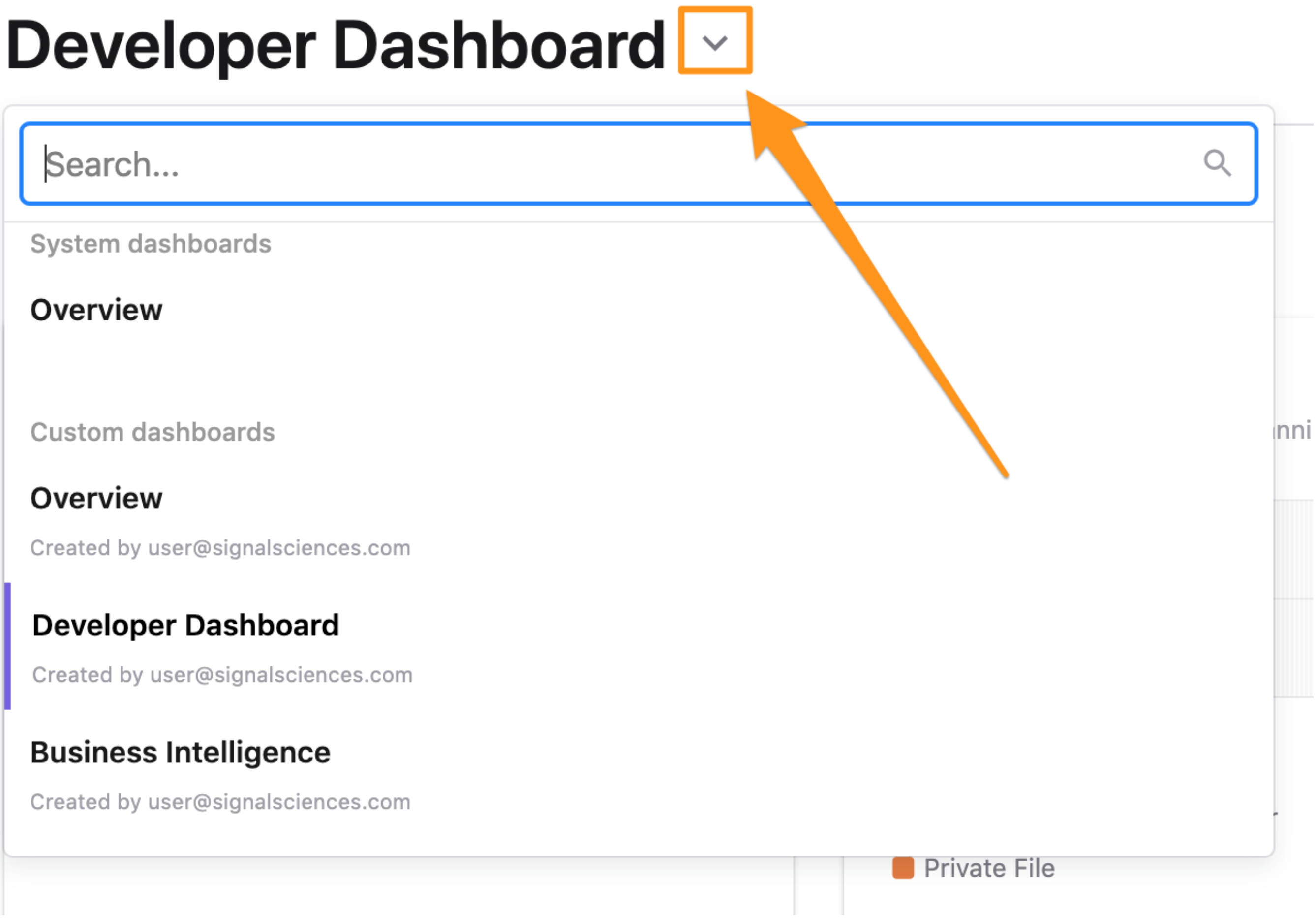


🟪 Attack Tooling	2k
🟦 Backdoor	129
🟨 Forceful Browsing	27k
🟩 Private File	7k

Dashboards

Selecting a dashboard

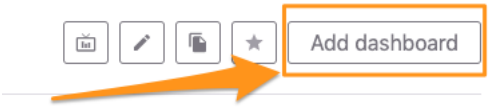
Switch between dashboards by clicking the chevron next to the name of the current dashboard:



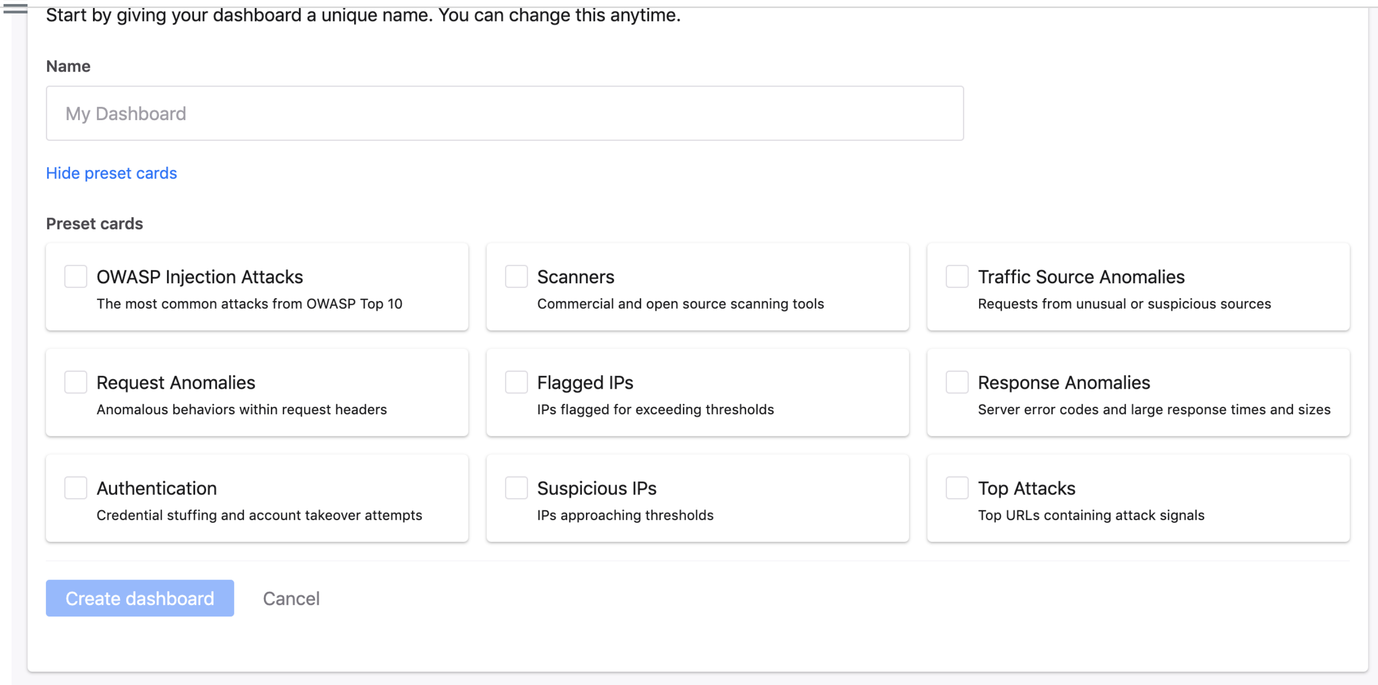
Choose which dashboard you want to switch to by selecting it from the menu. Narrow down the list using the search field at the top.

Creating a new dashboard

To create an entirely new dashboard, click on the “Add Dashboard” button in the upper-right corner:



You will then be able to name the new dashboard, as well as select which cards the dashboard will start with. You will be able to further customize the dashboard as described above after it has been created.



Start by giving your dashboard a unique name. You can change this anytime.

Name

My Dashboard

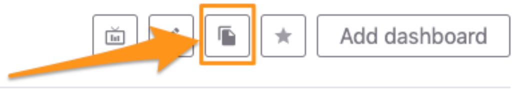
Hide preset cards

Preset cards

□ OWASP Injection Attacks The most common attacks from OWASP Top 10	□ Scanners Commercial and open source scanning tools	□ Traffic Source Anomalies Requests from unusual or suspicious sources
□ Request Anomalies Anomalous behaviors within request headers	□ Flagged IPs IPs flagged for exceeding thresholds	□ Response Anomalies Server error codes and large response times and sizes
□ Authentication Credential stuffing and account takeover attempts	□ Suspicious IPs IPs approaching thresholds	□ Top Attacks Top URLs containing attack signals

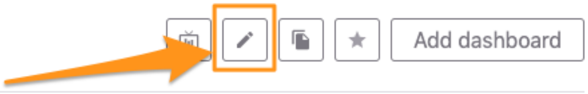
Duplicating dashboards

Duplicate existing dashboards by switching to the dashboard you wish to duplicate and clicking the “Make a copy of this dashboard” icon to the far right:



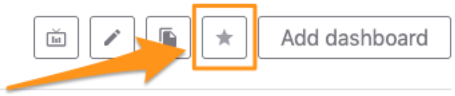
Managing dashboards

Rename and delete dashboards by switching to the dashboard you wish to manage and clicking the “Edit dashboard” icon to the far right:



Setting a default dashboard

Select a dashboard to be your default dashboard. This dashboard will automatically be selected when you first log into the Signal Sciences console. To set a default dashboard, switch to the dashboard you wish to make your default and then click the “Set as your default dashboard” button on the far right:



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.

- 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.

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. Within VictorOps, go to **Settings** > **Integrations**.
2. Under **Incoming Alerts**, choose **REST Endpoint**.
3. Click **Enable Integration** if you have not already generated an API key.
4. On Signal Sciences, go to **Manage** > **Site Integrations**.
5. Click **Add site integration** and select the **VictorOps Alert** integration.
6. Enter the **Post URL** in the Webhook URL field and choose what activity you want to trigger notifications.
7. Click **Add**.

Note that your VictorOps `Post URL` will be in the format of:

```
https://alert.victorops.com/integrations/generic/XXXXXXXXX/alert/XXXXXXXXXXXXX/$routing_key
```

Change `$routing_key` to your target group who should be notified of the alert. Failure to do so may result in missed notifications.

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 was flagged
expireFlag	An IP 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

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

System Signals

Attacks

Long name	Short name	Search/URL name	Description
Attack Tooling	Attack Tooling	`USERAGENT`	Attack Tooling is the use of automated software to identify security vulnerabilities or to attempt to exploit a discovered vulnerability
AWS SSRF	AWS SSRF	`AWS-SSRF`	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	`BACKDOOR`	A backdoor signal is a request which attempts to determine if a common backdoor file is present on the system
Command Execution	CMDEXE	`CMDEXE`	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	`XSS`	Cross-Site Scripting is the attempt to hijack a user's account or web-browsing session through malicious JavaScript code
Directory Traversal	Traversal	`TRAVERSAL`	Directory Traversal is the attempt to navigate privileged folders throughout a system in hopes of obtaining sensitive information
SQL Injection	SQLI	`SQLI`	SQL Injection is the attempt to gain access to an application or obtain privileged information by executing arbitrary database queries

Anomalies

Long name	Short name	Search/URL name	Description
Abnormal Path	ABNORMALPATH	`ABNORMALPATH`	Abnormal Path indicates the original path differs from the normalized path (e.g /foo/./bar is normalized to /foo/bar)
Bad Hop Headers	BHH	`BHH`	Bad Hop Headers indicate an HTTP smuggling attempt through either a malformed Transfer-Encoding (TE) and/or Content-Length (CL) header, or a well-formed TE and CL header
Blocked Requests	Blocked Request	`BLOCKED`	Requests blocked by Signal Sciences
Code Injection PHP	Code Injection	`CODEINJECTION`	Code Injection is the attempt to gain control or damage a target system through arbitrary application code commands by means of user input. Note, this signal only covers PHP code and is currently in an experimental phase. Contact support if you encounter any issues with this signal.
Datacenter Traffic	Datacenter	`DATACENTER`	Datacenter Traffic is non-organic traffic originating from identified hosting providers. This type of traffic is not commonly associated with a real end user. Datacenter IP ranges are sourced from ipcat.
Double Encoding	Double Encoding	`DOUBLEENCODING`	Double Encoding checks for the evasion technique of double encoding html characters
Forceful Browsing	Forceful Browsing	`FORCEFULBROWSING`	Forceful Browsing is the failed attempt to access admin pages
HTTP 403 Errors	HTTP 403	`HTTP403`	Forbidden. This is commonly seen when the request for a url has been protected by the server's configuration.
HTTP 404 Errors	HTTP 404	`HTTP404`	Not Found. This is commonly seen when the request for a page or asset does not exist or cannot be found by the server.

Errors			the number of active connections to a server.
HTTP 4XX Errors	HTTP4XX	`HTTP4XX`	4xx Status Codes commonly refer to client request errors
HTTP 500 Errors	HTTP 500	`HTTP500`	Internal Server Error. This is commonly seen when a request generates an unhandled application error.
HTTP 503 Errors	HTTP 503	`HTTP503`	Service Unavailable. This is commonly seen when a web service is overloaded or sometimes taken down for maintenance.
HTTP 5XX Errors	HTTP5XX	`HTTP5XX`	5xx Status Codes commonly refer to server related issues
HTTP Response Splitting	Response Splitting	`RESPONSESPLIT`	Identifies when CRLF characters are submitted as input to the application to inject headers into the HTTP response
Invalid Encoding	Invalid Encoding	`NOTUTF8`	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 Encoding Error	`JSON-ERROR`	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	`MALFORMED-DATA`	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. Requires agent 3.2 or higher.
Malicious IP Traffic	Malicious IP	`SANS`	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	`SIGSCI-IP`	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 IPs that contain any additional signal for the duration of the flag.
Missing "Content-Type" request header	No Content Type	`NO-CONTENT-TYPE`	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	No UA	`NOUA`	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	Null Byte	`NULLBYTE`	Null bytes do not normally appear in a request and indicate the request is malformed and potentially malicious
Private Files	Private File	`PRIVATEFILE`	Private files are usually confidential in nature, such as an Apache .htaccess file, or a configuration file which could leak sensitive information
Scanner	Scanner	`SCANNER`	Identifies popular scanning services and tools
SearchBot Impostor	Impostor	`IMPOSTOR`	Search bot impostor is someone pretending to be a Google or Bing search bot, but who is not legitimate
Tor Traffic	Tor Traffic	`TORNODE`	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	Weak TLS	`WEAKTLS`	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 Encoding Error	`XML-ERROR`	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 often related to a programming error or an automated or malicious request.

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

In reverse proxy mode, the Agent will start one or more listeners and proxy all traffic received on the listener(s) to the configured upstream server. Both HTTP, HTTPS (TLS) listeners can be enabled. Note that configuring the Agent in reverse proxy mode will disable the RPC listener and 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 host(s) that the Agent will proxy requests to. The upstream hosts are a comma separate list of URLs. The scheme of the URLs specify the protocol that will be used for listening and proxying to the upstreams.

> **Note:** If your load balancer is configured for sticky session load balancing, 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:{...}; ...
```

```
SIGSCI_REVPROXY_LISTENER="example1:{listener=http://203.0.113.13:80,upstreams=http://192.168.1.2:80}"
```

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

Bad Request error response. This may affect some simple monitoring from older monitors and load balancers.

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 Agent and Module Installation

Explore other installation options:

- Explore module options

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 3rd 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

The Signal Sciences CTR integration is a native integration that's easy to install in minutes. The integration is available within the SecureX console:

> **Note:** The user setting up the CTR integration must have permission to create API Access Tokens.

1. Log into the Signal Sciences Console
2. Create an API Access Token for your user
3. You will need to generate an **Authorization Bearer Token** from this API Access Token:
    - The **Authorization 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 of this in Javascript is:

      ```
      btoa("user@example.com:api-access-token") = "YW5keUBleGFtcGxlY29ycC5jb206ZXhhbXBsZXRva2Vu"
      ```
4. Log into your SecureX console
5. Click on the **Integrations** tab at the top
6. In the navigation bar on the left, select **Integrations** > **Available Integrations**
7. In the list of available modules, locate the **Signal Sciences Next-Gen WAF** module and click on **Add New Module**
8. Complete the form by entering the following:
    - **Module Name** - Leave the default name or enter a name that is meaningful to you (for example, if you plan to have multiple integrations for several cloud instances)
    - **URL** - `https://dashboard.signalsciences.net/api.v0/corps/<your-corp-name>/ctr` (your corp name is the string that appears in the URL after logging into the Signal Sciences console)
    - **Authorization Bearer Token** - The base64-encoded token you generated in Step 3
9. Click the **Save** button to finish setting up the integration

Using the Cisco Threat Response Integration

The Sighting will show when the IP was flagged, the URL that was targeted, and a link back to the flagged IP event within the SigSci console. The Indicator will describe the attack signal that was associated with the flagged IP (i.e. XSS).

Debian NGINX-Plus

Add the Package Repositories

We'll first add in the Signal Sciences apt repositories as this simplifies the installation process.

Debian 9 "stretch"

Cut-and-paste 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"

Cut-and-paste 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"

Cut-and-paste 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 the module with apt

Then install the module by running the following command for your NGINX version:

NGINX+ 19

```
sudo apt-get install nginx-module-sigsci-nxp=1.17.3*
```

NGINX+ 18

```
sudo apt-get install nginx-module-sigsci-nxp=1.15.10*
```

NGINX+ 17

```
sudo apt-get install nginx-module-sigsci-nxp=1.15.7*
```

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

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.
Anomalies	Abnormal requests that, although not attacks, may still be notable. Examples include malformed request data and 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 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.
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 IPs.
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. Events are automatically system generated.
Flagged IPs	An IP that has been flagged for containing attack traffic that has exceeded 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	IPs are converted to anonymous IPv6 so that Signal Sciences will not know the actual IP, which causes the IP 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, IPs, 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 define arbitrary conditions and automatically begin to block or tag requests that pass a user-defined threshold.

Redactions	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	A descriptive tag about a request.
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	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.
Suspicious IPs	IPs 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 IPs 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.
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.

Performance & Reliability

Performance

How does your architecture ensure high performance and reliability?

One of the key reasons for the architectural split between the module and the agent is to optimize for maximum performance and reliability. If the agent ever crashes, your application does not go down because the module fails open if it doesn't hear back from the agent within a set time limit. This claim is simple to verify in a deployment, as the module can be enabled without running the agent, and the site will continue to load as normal. From the performance side, this set time limit is also the worst case latency that Signal Sciences could introduce to a request.

Can I see the actual performance impact of Signal Sciences on my systems?

Yes. We provide graphs and data on resources used by the agent in the agent details page in the console.

How much memory does Signal Sciences consume?

Most clients see median memory usage of 1024mb (1GB) in production deployments.

How much CPU does Signal Sciences consume?

CPU varies by machine size. By default the number of available cores determines the maximum cores the agent can use.

The agent scales using the following by default (overridable - see below):

Available Cores	Agent Core Limit
1	1
2 - 3	2
4 or more	50% of available

Agent and Module

How much time does the agent spend processing a request?

Most clients see a median time of 0.6ms to 2.0ms in production deployments.

How often does the agent poll for new decisions?

What measures are in place to ensure agent updates are from an authorized source?

Agent updates, such as new decisions made by the back end, are encrypted by the back end and then decrypted by the agent using the agent keys.

What impact does the agent to backend communication have on my egress bandwidth?

Impact to egress bandwidth is minimal. Every 30 seconds, we compress any data we have collected and send it to our backend. In other words, it's a ratio of n inbound attacks to one outbound request to our backend.

Are my production systems impacted if the Signal Sciences backend goes down?

No. All agent communication with the backend is asynchronous. 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 perform detections of attacks, anomalies and any custom rules/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, 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

How can I disable the agent?

The agent can be disabled in two ways: 1) by clicking the agent mode toggle at the top of the consoles and selecting "off," and 2) by stopping agents via configuration management.

How do I increase the number of CPUs available to the agent?

By default the agent is configured to scale proportionally based on the number of available CPUs on a system (see above). This is typically a reasonable number, but cases of extremely high throughput can lead to resource contention, which manifests as higher latency and increased memory utilization with a slightly elevated decision time.

To change the number of cores available to the agent, edit the agent configuration file (typically `/etc/sigsci/agent.conf`) to include the line `max-procs = n` where "n" is the number of CPU cores to use. You must then restart the agent for this change to take effect.

What's the difference between the "Host CPU" and "Agent CPU" metrics?

The "Host CPU" metric indicates the CPU percentage for the full host wherein 100% is all cores.

The "Agent CPU" metric on the other hand doesn't use a scale of 100%. The Agent CPU metric is the CPU by core.

For example, take a machine with 8 cores: the maximum Agent CPU percentage would be 800%. However, if the agent has been configured to be limited to only 4 cores, the maximum Agent CPU percentage would instead be 400%. In this example, if the agent is shown to be taking about 50% CPU, it's actually only using 6% CPU (50% Agent CPU / 800% total CPU).

Updates

How frequently do you release updates to the agent and module?

See:

- Release Notes for the Agent
- Release Notes for the NGINX Module
- Release Notes for the Apache Module
- Release Notes for the PHP SDK Module
- Release Notes for the Python Module

How are updates to the agent/module tested?

Our testing process includes:

- Unit tests
- Integration tests
- Security tests
- Signal detection tests (quality test)
- Module correctness tests
- Packaging / install tests
- Performance tests (load tests)

Most of these are completely automated and run regularly, if not constantly.

Golang Module Release Notes

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
- Updated timeout to include time to connect to the agent
- Cleaned up docs/code/examples

- 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

Corp Overview Report

- Which of your sites is attacked the most.
- Which of your sites is seeing the most blocked traffic.
- Which of your sites has the most flagged, malicious IPs.

In addition to high-level stats, the Corp Overview Report also provides attack type and source breakdowns, enabling you to better understand how your sites are being attacked.

How do I access the report?

Access the report by clicking on the name of your corp in the upper left corner of the console.

What data is being shown in the report?

The data being shown in the report is the set of all malicious requests (requests containing 1-or-more attack signals).

Malicious requests

This is a count of all requests with 1-or-more attack signals.

Blocked requests

This is the subset of malicious requests which were blocked. Learn more about how our product decides to block requests.

Malicious IPs

This is the set of IPs whose subsequent malicious requests were blocked due to a threshold of malicious requests being exceeded.

Top attack types

This is the breakdown of malicious signals observed.

Top attack sources

This is the breakdown of attack sources, such as specific countries or private IPs.

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. Go to **Manage** > **Site Integrations**.
4. Click **Add site integration** and select the **Generic Webhook** integration.
5. Paste in the **HTTP Source Address** for the hosted collector.
6. Choose whether to receive notifications for all activity or specific activity.
7. Click **Add**.

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 was flagged	Get event by ID
expireFlag	An IP flag was manually expired	List events
createCustomRedaction	A custom redaction was created	Create a custom redactions
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	

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	

Amazon Linux NGINX 1.14.1+

Add the Package Repositories

First, set up the key and package sources for the Signal Sciences repository:

> **Note:** Our distribution release depends on the EPEL repository, you will need to ensure your system also has it installed.

Nginx version 1.18.0+ running either Amazon Linux 2 / Amazon Linux 2018.03

Cut-and-paste the following script:

Nginx version 1.14.1 < 1.17.9 on Amazon Linux 2

Red Hat CentOS 7

Nginx version 1.14.1 < 1.17.9 on Amazon Linux 2018.03

Red Hat CentOS 6

Install the Nginx module using yum

Nginx version 1.18.0+ running either Amazon Linux 2 / Amazon Linux 2018.03

Install the module by running the following command, replacing "NN.NN" with your Nginx version number:

Nginx version 1.14.1 < 1.17.9 on Amazon Linux 2 / Amazon Linux 2018.03

Install the module by running the following command, replacing "NN.NN" with your Nginx version number:

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

Amazon Linux 2

```
systemctl restart nginx
```

PHP

PHP SDK Module Release Notes

2.1.0 2021-08-11

- Standardized release notes
- Added module testing capability

2.0.1 2021-07-29

- Added support for content-type application/graphql

2.0.0 2021-02-11

- Added support to block on HTTP codes 300-599
- Added support for OPTIONS and CONNECT methods
- Added redirect support

1.2.3 2018-06-29

- Standardized release notes
- Fixed pear packaging

1.2.2 2018-01-31

- Added support for multipart/form-data post
- Added ability to send all HTTP headers to agent for inspection

1.2.1 2017-08-23

- Fixed module type

1.2.0 2017-03-21

- Added ability to send XML posts to agent

1.1.1 2016-07-20

- No operational changes
- Added new download option https://dl.signalsciences.net/sigsci-module-php/sigsci-module-php_latest.tar.gz

1.1.0 2016-07-14

- Improved error handling
- Switched to SemVer version numbers

1.0.0.52 2016-02-16

- Improved and simplified networking calls
- Improved error messages
- Upgraded MessagePack library
- Added support for detection of open redirects
- Configuration change: Originally HTTP methods that were inspected where explicitly listed (allowlisted, e.g. “GET”, “POST”) using the `allowed_methods` configuration parameter. The logic is now inverted, and one lists methods that should be ignored (blocklisted, e.g. “OPTIONS”, “CONNECT”) using the `ignore_methods` parameter. This allows for the detection of invalid or malicious HTTP requests.
- Added more detailed PHP version information sent to the agent for better identification and debugging

1.0.0.48 2015-10-26

Privacy

What data gets sent to the Signal Sciences backend?

Not all traffic is sent to the Signal Sciences backend, but the agent does pre-filtering locally to determine if the request contains an attack. When the agent identifies an attack or anomaly in the request, it only sends parameters with identified attacks to the platform backend. The entire request is never sent to the Signal Sciences backend and certain portions of the request are explicitly never sent to the backend, such as session tokens or tracking cookies. Additional information regarding data redaction can be found here.

What if I have other fields that are sensitive to my application?

We provide a configuration mechanism in the console to add additional fields which will always be filtered. For example, if your password field is named “foobar” instead of “password,” we will redact that field in the agent before it’s sent to our backend. Instructions for specifying additional fields to be redacted can be found here.

How long does Signal Sciences retain the data it collects?

For searching purposes, data is retained for 30 days. Data can only be extracted within 24 hours.

How does Signal Sciences use the data it collects?

We use the data to provide visibility and make decisions about blocking attacks to your application.

Can the data be attributed back to me or any of my users?

No. We’ll never attribute any data back to your organization or end users.

What happens if I want to scrub something after the fact?

See something in the raw data that you’d rather delete? We can delete the data for entire days from our database. Submit a support request with the date range you want to delete and we’ll scrub our database of your requested data.

What response data does the Signal Sciences backend see?

Signal Sciences only collects the response’s metadata, i.e. response codes, sizes, and times.

Using Single Sign-on

Single sign-on (SSO) is a means of allowing your users to authenticate against a single identity provider to access your corporation. We support both SAML 2.0 and Google Apps SSO (OAuth 2.0).

How do I enable 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

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 publish metadata at present, but may in the future.

> **Note:** If using PingFederate as your SSO provider, you will need to deselect “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.

In Signal Sciences

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?

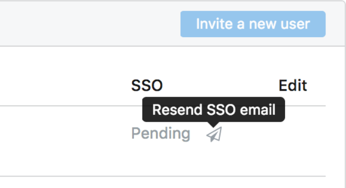
If the email from your identity provider doesn't match the email in your Signal Sciences account, you will be alerted that your Signal Sciences email will be changed to your identity provider's email when you enable SSO.

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

Once you enable SSO, the passwords/2FA tokens for any existing users will be deleted, and they'll be sent an email to set up SSO on their accounts. This email will be valid for 3 days.

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.

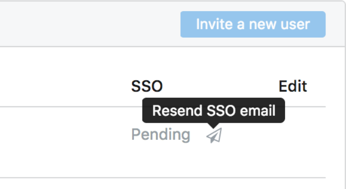
What if an existing user authenticates with an email address in their identity provider that doesn't match the email in their Signal Sciences account?

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 corporation.

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?

Single sign-on can be disabled by Owners on the **User Authentication** page under the **Corp Manage** menu. Under **Built-in Auth** in the **Authentication** section, click **Switch to built-in auth**.

You will be required to set up a new password to continue. Once you disable single sign-on, all other users in your corporation will have their active sessions expired and will receive an email telling them that SSO has been disabled with a link to set a new password.

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.

Linking Fastly Accounts

You can link your Fastly and Signal Sciences accounts, allowing you to sign in using your Fastly account login credentials and freely switch between the Signal Sciences and Fastly consoles. After linking your accounts, you will only be able to log into the Signal Sciences console using your Fastly account credentials.

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.

Before you begin

Before you begin linking your Fastly and Signal Sciences accounts, understand the following:

- You can not unlink your Fastly and Signal Sciences accounts once they have been linked.
- Linked accounts do not currently support SAML authentication. Linked accounts authenticate using your Fastly email address and password, rather than through your identity provider.
- 2FA is supported, but must be enabled on both your Fastly and Signal Sciences accounts before you will be able to link them.
- Signal Sciences accounts set to bypass SSO can not be linked.

How to link your Fastly and Signal Sciences accounts

1. Log into the Signal Sciences console.
2. From the **Profile** menu, select **Account Settings**. The account settings management 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.

IDP Provisioning

In addition to SAML SSO support for authentication, Signal Sciences also supports automated user management through Okta.

Features

The following features are supported:

- 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.

Provisioning enables you to automatically synchronize user access to Signal Sciences sites as well as their role (such as an Owner or Admin).

> **Note:** A user that is provisioned by Okta cannot be modified or deleted in Signal Sciences. All changes must happen inside of Okta.

Requirements and Preparation

1. In your Signal Sciences account, enable Single Sign On to use Okta as your SSO provider.

2. If you do not have one already, create a Signal Sciences application in Okta. Follow the instructions listed in the Okta Signal Sciences application, which provide specific configuration information.

3. Create an API Access Token in Signal Sciences and store it in a secure location for use later in this guide.

Step-by-Step Configuration Instructions

Enter configuration information

In the **Provisioning** tab of the Signal Sciences Okta application, enable provisioning. Enter the following information:

- **SCIM connector base URL:**
  - This will be `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:** Check the boxes for "Push New Users" and "Push Profile Updates"
- **Authentication Mode:** Select "HTTP Header"
- **Authorization:**
  - You will need to 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:

Confirm your connection was configured correctly by clicking **Test Connector Configuration**. If everything is configured correctly, you will see "Signal Sciences was verified successfully!":

Click the green **Save** button to save this configuration and proceed.

Enable provisioning features

After the settings are saved, check the following **Enable** checkboxes under **Provisioning to App**:

- Create Users
- Update User Attributes
- Deactivate Users

Click the green **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`.

Assign 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**. Then click **Assign** > **Assign to Groups**
2. Select a group of users to provision to Signal Sciences
3. A window will appear requesting additional attributes
4. Add 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**

   Note: Signal Sciences only accepts email addresses with letters that are lowercase. Email addresses with uppercase letters will result in erroneous behavior.

What happens to existing (SAML) users when Okta user provisioning is set up for the first time?

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.

User Management

User Updates

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.

User Deletion

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.

Amazon Linux NGINX 1.10-1.14

Add the Package Repositories

First, set up the key and package sources for the Signal Sciences repository:

> **Note:** Our distribution release depends on the EPEL repository, you will need to ensure your system also has it installed.

> **Note:** We are currently supporting Amazon Linux 2018.03 or earlier RHEL6 based OS.

Red Hat CentOS 7

Cut-and-paste 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.

Cut-and-paste 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
```

Enabling Lua for NGINX

For older versions of NGINX, we require NGINX to be built with Lua and LuaJIT support. It is recommended to first ensure that Lua is installed and enabled for NGINX before enabling the Signal Sciences NGINX module.

Install the Lua NGINX Module

The first step is to install the dynamic Lua NGINX Module appropriate for your NGINX distribution:

Nginx.org distribution

NGINX 1.10

Amazon distribution

NGINX 1.10

Enable the Lua NGINX Module

```
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

To verify that Lua has been loaded properly load the following config(ex: `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
```

```
end

';

}
```

Example of successfully loading the 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 and Configure the Signal Sciences NGINX Module

1. Install the module

   ```
   sudo yum install sigsci-module-nginx
   ```

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

   ```
   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
   ```

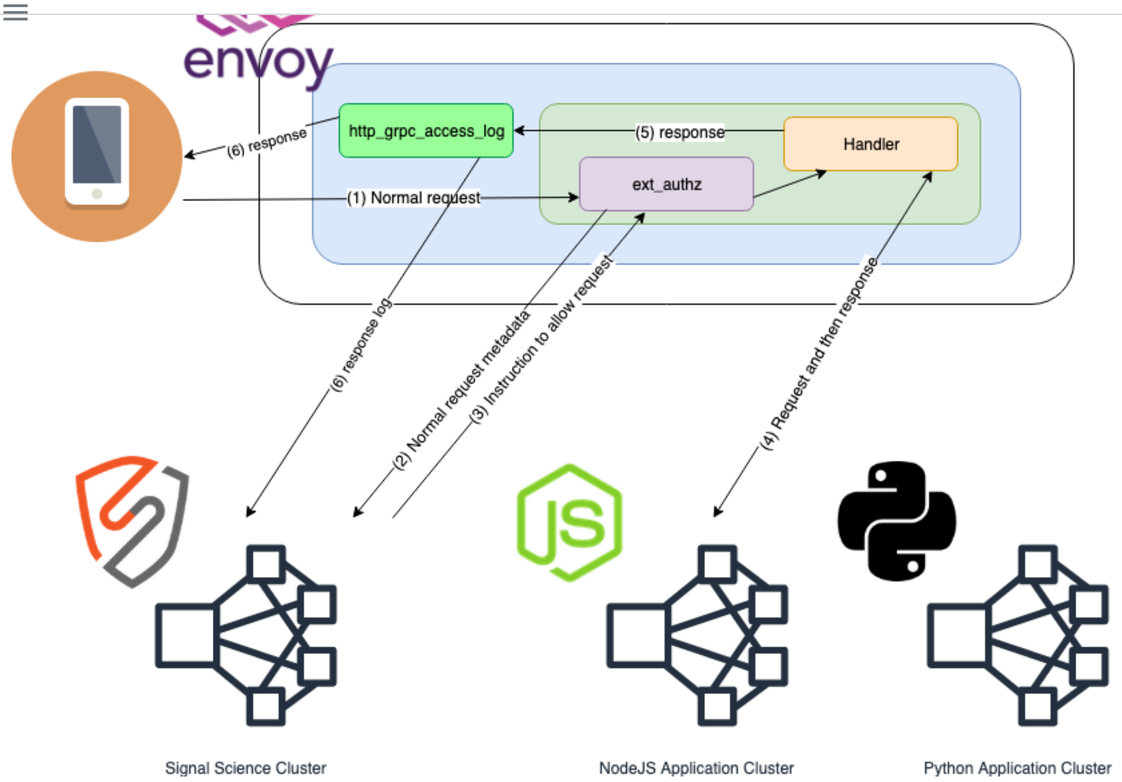
Envoy Proxy gRPC Authorization Mode

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.

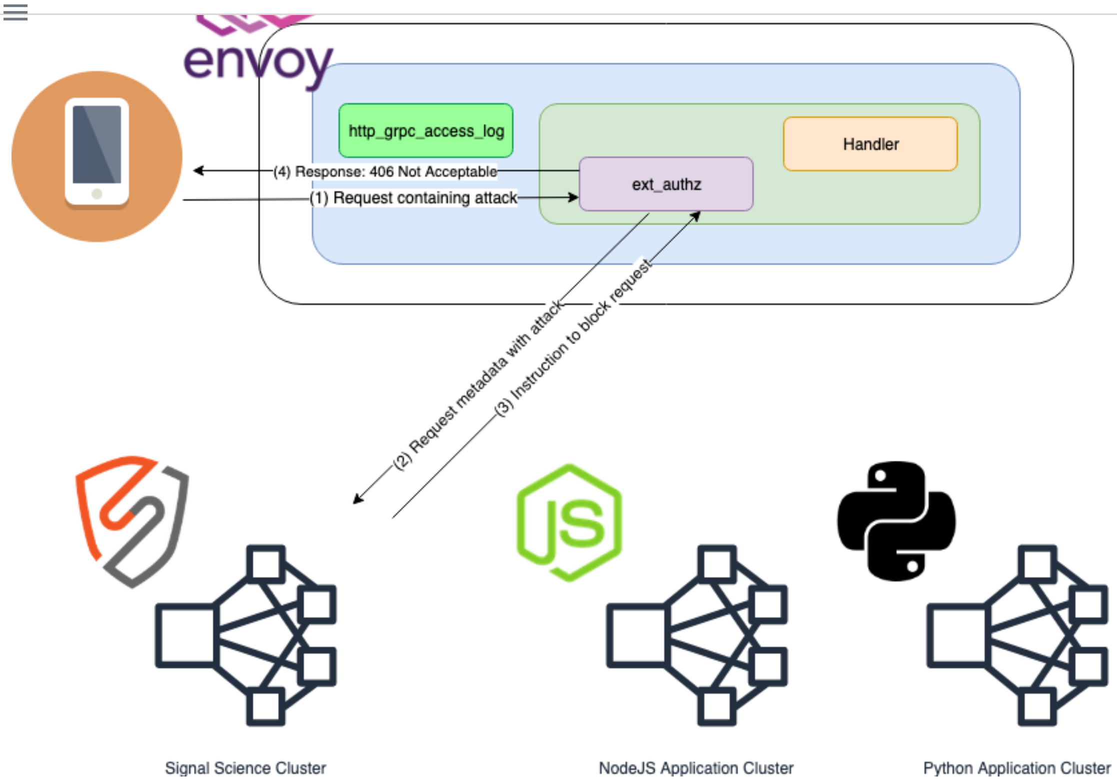
Request Allowed (normal) Processing


envoy
http_grpc_access_log
(5) response
Handler
ext_authz
(6) response
(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

This is the flow for normal requests that the `sigsci-agent` allows 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 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:

```
SIGSCI_ENVOY_GRPC_ADDRESS=0.0.0.0:8000
```

Optionally, the `sigsci-agent` can be configured with TLS enabled. To do this, set the certificate and key files in the `sigsci-agent` 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 available in the `sigsci-agent` version 3.18.0

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
```

As of `sigsci-agent` version 3.24.0 and later, 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.

> **Note:** The `connect_timeout` is the timeout to connect to the `sigsci-agent` (but not to process the data) and can be adjusted if required. The `tls_context` option must be defined if TLS is to be used. TLS can be configured in the `sigsci-agent` config 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.ext_authz
```

```
        cluster_name: sigsci-agent-grpc
      timeout: 0.2s
    failure_mode_allow: true
    ### Include the request body data (Envoy v1.10+ only, see limitations and
    ### workarounds for older versions)
    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
- name: envoy.router
  config: {}
```

> **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
  config:
    common_config:
      log_name: "sigsci-agent-grpc"
      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

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, as of `sigsci-agent` v4.3.0, 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 (`sigsci-agent` v4.3.0+): 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)
             if len > 0 and len <= 8192 then
               -- Triggers buffering
               reqbody = req:body():getBytes(0, len)
               -- 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: {}
```

small snippet of Lua code must be added to extract the body and add it to the request as follows.

> **Note:** The following Lua workaround may cause 503 responses to be returned if HTTP/2 is enabled. If you must enable HTTP/2 support, then you should use envoy 1.10 or newer so that the body can be extracted using the native method via the `with_request_body` option instead of using this workaround. If you cannot use envoy v1.10.0 or greater, then it is recommended that you upgrade to `sigsci-agent` v4.3.0 or later and use the previous Lua workaround which will work in more cases.

Example of including the request body data if it is <= 8KB (adjust the limit if required):

```
http_filters:
- name: envoy.lua
  config:
    inline_code: |
      -- Add an internal :body header to pass the body if <= 8KB
      function envoy_on_request(req)
        len = 0
        cl = req:headers():get("content-length")
        if cl ~= nil then
          len = tonumber(cl)
        end
        if len > 0 and len <= 8192 then
          reqbody = req:body():getBytes(0, len)
          req:headers():add(":body", reqbody)
        end
      end
- name: envoy.ext_authz
  config:
    grpc_service:
      envoy_grpc:
        cluster_name: sigsci-agent-grpc
    failure_mode_allow: 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:
```

```
- "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.

```
additional_response_headers_to_log:
- "date"
- "server"
- "content-type"
- "content-length"
```

Next Steps

- Verify Agent and Module Installation

Explore other installation options:

- Explore module options

NodeJS

Node.js Module Release Notes

Unreleased

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

- Added support for Hapi v17

1.5.3 2020-05-28

- Fixed an issue where form post data wasn’t read fully

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

- Corrected code to conform to standard
- Made no other functional changes

1.2.1 2016-07-20

- 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

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.

To view the Corp Audit Log:

1. Log into the Signal Sciences dashboard.
2. From the **Corp Manage** menu, select **Corp Audit Log**. The Corp Audit Log page appears.

Activity types

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
User email marked undeliverable	A user’s email address bounced

corp	
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

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.

To view the Sit Audit Log:

1. Log into the Signal Sciences dashboard.
2. From the **Manage** menu, select **Site Audit Log**. The Site Audit Log page appears.

Activity types

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

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 flag was manually expired
New agent online	A new agent was detected
Site integration created	A new site-level integration was created
Site integration updated	A site-level integration was updated
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

Amazon Linux NGINX 1.9 or lower

Add the Package Repositories

First, set up the key and package sources for the Signal Sciences repository:

> **Note:** Our distribution release depends on the EPEL repository, you will need to ensure your system also has it installed.
>
> **Note:** We are currently supporting Amazon Linux 2018.03 or earlier RHEL6 based OS.

Red Hat CentOS 7

Cut-and-paste 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.

Cut-and-paste 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
```

Enabling Lua for NGINX

For older versions of NGINX, we require NGINX to be built with the third party `ngx_lua` module. As older versions of NGINX do not support dynamically loadable modules you would typically be required to rebuild from source.

To assist customers, we provide pre-built drop in replacements NGINX packages already built with the `ngx_lua` module. This is intended for customers who prefer not to build from source, or who either use a distribution provided package or an official NGNIX 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 off of. 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:

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 Matrix

The following version 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

```
    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 sigsci repository:

    ```
    yum -q makecache -y --disablerepo=* --enablerepo=sigsci_*
    ```

3. Install the Signal Sciences provided NGINX

    ```
    yum install nginx
    ```

Check that Lua is loaded correctly

To verify that Lua has been loaded properly load the following config(ex: 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
```

```
    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 successfully loading the 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 and Configure the Signal Sciences NGINX Module

1. Install the module

```
sudo yum install sigsci-module-nginx
```

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

```
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

Installation

> **Note:** These steps assume:
> • IHS is installed in `/opt/IBM/HTTPServer`. If IHS is installed in a different path, use the appropriate path for your IHS installation.
> • IHS is installed on CentOS, if assistance is needed with another platform, contact Support.

1. Install the Signal Sciences agent:

https://docs.signalsciences.net/install-guides/agent-installation/agent-install-intro/

2. Download the Signal Sciences module package for your version of IHS:

> **Note:** Replace <VERSION> with the latest module version found here: https://dl.signalsciences.net/?prefix=sigsci-module-apache/

- For IHS version < 9.0, use the `apache22` Signal Sciences module:
  - Download:

    ```
    wget https://dl.signalsciences.net/sigsci-module-apache/<VERSION>/centos/el6/sigsci-module-apache-<VERSIO
    ```

  - Extract:

    ```
    tar -xzf sigsci-module-apache-<VERSION>.el6-1.x86_64.tar.gz
    ```

```
wget https://dl.signalsciences.net/sigsci-module-apache/<VERSION>/centos/el7/sigsci-module-apache-<VERSIO
```

- Extract:

```
tar -xzf sigsci-module-apache-<VERSION>.el7-1.x86_64.tar.gz
```

3. Copy mod_signalsciences.so to the IBM HTTP Server modules directory:
   - `cp mod_signalsciences.so /opt/IBM/HTTPServer/modules`
4. Add the LoadModule directive to the IBM HTTP Server httpd.conf file (`/opt/IBM/HTTPServer/conf/httpd.conf`):
   - `LoadModule signalsciences_module modules/mod_signalsciences.so`
5. Restart the IBM HTTP Server:
   - `/opt/IBM/HTTPServer/bin/apachectl restart`

HAProxy

HAProxy Module Release Notes

Unreleased

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 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

1.1.5 2019-02-07

- Added a default timeout for network operations (set `sigsci.timeout` to override)
- 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

Amazon Linux NGINX-Plus

Add the Package Repositories

First, set up the key and package sources for the Signal Sciences repository:

Red Hat CentOS 7

Cut-and-paste 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.

Cut-and-paste 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 the Nginx module using yum

Then install the module by running the following command for your NGINX version:

NGINX+ 19

```
sudo yum install nginx-module-sigsci-nxp-amzn-1.17.3*
```

NGINX+ 18

```
sudo yum install nginx-module-sigsci-nxp-amzn-1.15.10*
```

NGINX+ 17

```
sudo yum install nginx-module-sigsci-nxp-amzn-1.15.7*
```

Update the Nginx configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`.

Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

Amazon Linux 2018.03

```
restart nginx
```

Verifying Data Privacy

To learn more about how Signal Sciences filters and sanitizes requests, see our Data Privacy page. To verify our agents are correctly filtering and sanitizing requests, we provide a raw log of data that's sent from our agents:

1. Go to the Agents page
2. Click on an agent
3. Click the **Requests** tab
4. Verify that data is being correctly redacted

If you have a field which we're not filtering out, we also allow you to add custom field redactions:

1. Go to **Rules** > **Redactions**
2. Click on **New redaction**
3. Enter the field name and type you'd like to redact
4. Click **Create redaction**

Python

Python Module Release Notes

1.4.0 2021-06-04

- Added qraphql support
- Updated module to use independent python 2 & 3 implementations

1.3.2 2021-05-24

- Fixed missing agent code in anomalous requests
- Fixed missing field causing agent rpc errors

1.3.1 2020-02-25

- Added cryptographic signatures to released RPM packages

1.3.0 2020-08-04

- Added support for setting redirect location
- Added support for blocking on response code range 300 - 599

1.2.2 2019-06-06

- Fixed handling of xml content type

1.2.1 2019-05-22

- Fixed incompatibility with gunicorn

1.2.0 2018-02-16

- Fixed incompatibility with sigsci agent rpc/msgp

1.1.1 2018-02-16

- Added support for multipart/form-data post
- Standardized release notes
- Added ubuntu 18.04 packaging

1.1.0 2017-09-21

- Improved performance and correctness

1.0.1 2017-09-08

1.0.0 2017-07-11

Initial release

Alpine Linux NGINX 1.15.3+

> **Note:** The Nginx Module for Alpine Linux requires Nginx v1.15.3 or higher.

Add the Package Repositories

We'll first add in the Signal Sciences `apk` repositories as this simplifies the installation process:

Alpine 3.12 - Container

```
apk update && apk add wget
```

Alpine 3.12 - Bare Metal

```
sudo apk update && sudo apk add wget
```

Alpine 3.11 - Container

```
apk update && apk add wget
```

Alpine 3.11 - Bare Metal

```
sudo apk update && sudo apk add wget
```

Optional: Verify Signing Key

Verify the downloaded key contains the proper key:

```
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 Module With apk

Update the Nginx Configuration

Edit your `nginx.conf` file located by default at `/etc/nginx/nginx.conf`. Add the following lines to the global section. For example after the `pid /run/nginx.pid;` line add:

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

Restart the Nginx web service

Restart Nginx running on a VM and bare-metal:

```
sudo service nginx restart
sudo rc-service nginx restart
```

Verifying Performance and Reliability

To learn more about our philosophy on performance and reliability, see Performance & Reliability. We provide a number of metrics to understand the performance impact on your infrastructure:

1. On the Agents page, click on an agent
2. Click on the **Graphs** tab
3. Change the time range via the time selector on the top right

The **Graphs** tab shows you information on the number of requests we've processed, any errors observed, memory usage, CPU percentage, decision times, and more. For reference, these are some baseline numbers we've seen with other customers:

1. Median memory usage of 100–400 MB in production
2. Median decision time of 0.6–2.0ms in production

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

- Upgrade to 1.10.3

2.2.1 2016-12-23

- 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

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

Header Links

Header links are a means of easily cross-referencing our data with your own internal systems via a hyperlink. We currently support linking either request or response headers to any system (e.g., Kibana, Splunk, etc…).

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 into the Signal Sciences Console

2. Go to **Manage** > **Header Links**

3. Click **Add header link**

    - **Type:** specifies whether you want to link a request or response header.
    - **Header Name:** is the name of the header (e.g., X-Request-ID).
    - **Link Template:** is how we generate the link to your internal system. The link name is used for display purposes, (e.g., “View in Kibana”).
    - **Link Name:** Is the display name for the header link

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 should see a header link (e.g., “View in Splunk").

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**

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.

Upgrading the Signal Sciences NGINX module on Ubuntu/Debian systems

1. Upgrade the NGINX Lua module package

   ```
   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 and 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**

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

1.1.1 2018-05-21

- added debian 7 (wheezy) package

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

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.

Configuration Options

The following are the current configuration options (as of v4.26.0 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"

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"

`config`=*string*

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-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"

How often to check for a new configuration

Default: "30s"

`download-url`=*string* [EXPERIMENTAL]

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"

Maximum in-flight RPC connections (by default scaled with rpc-workers)

Default: "0"

`max-inspecting=`*integer* [DEPRECATED]

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%". See https://docs.fastly.com/signalsciences/how-it-works/performance-reliability/#how-much-cpu-does-signal-sciences-consume for defaults.

`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 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`=*time-duration* [DEPRECATED]
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 available TLS cipher suites and exit

`statsd-address`=*string*

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

`--version` (commandline only option)

Show version information and exit

`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"

`conn-idle-max`=*integer*

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"

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]*

This experimental option replaces ‘close-conn-on-request-smuggling’ functionality. The option will need to be enabled per each reverse proxy listener.

Default: “false”

`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`=*string*

Listener URL [scheme://address:port]

`log-all-errors`*[=true|false]*

Log all errors, not just common

Default: “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 are made

Default: “true”

`upstreams=`*csv-string*

Upstream, comma separated upstream URLs [scheme://address:port]

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=https://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, just add the environment variable(s) configuration 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=https://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
```

- Edit the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sigsci-agent\Environment` value adding an environment variable and value in `var=value` form, one per line Example:

```
HTTPS_PROXY=https://10.0.0.1:8080
```

URLs with the correct proxy URLs:

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sigsci-agent /v Environment /t REG_MULTI_SZ /d "HTTPS
```

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=https://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

Install the Signal Sciences Module:

- Explore module options

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, ex: mod_example.c mod_something.c
SigSciRunAfterModulesList	Signal Sciences module runs after the list of specified modules, ex: 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 - thus 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
AgentPort	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.
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

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 **Nginx.org** package (`nxo`) only, add the following line:

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

> **Note:** For **NGINX Plus** 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	ex: 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.

Examples of configuration

Following is an example of setting SignalSciences module parameters in the `http` section:

```
# 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 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 of configuration

```
sigsci.agenthost = "unix:/var/run/sigsci.sock"
sigsci.agentport = 12345
sigsci.timeout = 100
sigsci.maxpost = 1000000
```

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
   ...
```

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

Typically, configuration changes are 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. Configuring via MSI or `SigsciCtl.exe` utility is recommended.

To set a configuration option when installing the MSI, just specify the option on the commandline in `option=value` format.via as follows:

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

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

```
"%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 such as follows:

```
"%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 -------------------- ------------ -----------------
> ```

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:

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

To reset the configuration to defaults using PowerShell, run:

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

To set a configuration option via the `appcmd.exe` command line tool use the `-section:SignalSciences` option such as follows:

```
"%SYSTEMROOT%\system32\inetsrv\appcmd.exe" set config -section:SignalSciences -agentPort:737
```

To list the configuration using `appcmd.exe`, run (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:

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

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

Language Modules

See language specific module pages for configuration details.

Compatibility & Requirements

The Signal Sciences Agent and Modules are supported on the following Linux distributions:

Distribution	Code Name	Version
Alpine		3.11
Amazon Linux		>2015.09.01
CentOS	Enterprise Linux 6	6.x
	Enterprise Linux 7	7.x
	Enterprise Linux 8	8.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
Ubuntu	Precise	12.04 LTS
	Trusty	14.04 LTS
	Xenial	16.04 LTS
	Bionic	18.04 LTS
	Disco	19.04 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, Python). Specific details for some of the more commonly deployed platform are listed below:

NGINX Web Servers

The NGINX Module is offered in two different variations, depending on the platform and what best meets your needs:

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:

- 1.19.0
- 1.18.0
- 1.17.9
- 1.17.8
- 1.17.7
- 1.17.6
- 1.17.5
- 1.17.4
- 1.17.3
- 1.17.2

- 1.16.0
- 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:

- 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
1.2	1.2.7, 1.2.9
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)

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

PHP

Our PHP module is available both as a tarball and a PEAR package to simplify installation. The minimum version of PHP supported is 5.3.

Package Downloads

Agent

https://dl.signalsciences.net/sigsci-agent/sigsci-agent_latest.tar.gz

Apache

https://dl.signalsciences.net/sigsci-module-apache/sigsci-module-apache_latest.tar.gz

NGINX

https://dl.signalsciences.net/sigsci-module-nginx/sigsci-module-nginx_latest.tar.gz

Heroku

https://dl.signalsciences.net/sigsci-heroku-buildpack/sigsci-heroku-buildpack_latest.tgz

IBM Cloud

https://dl.signalsciences.net/sigsci-bluemix-buildpack/sigsci-bluemix-buildpack_latest.tgz

Pivotal Platform & Pivotal Web Services (PWS)

https://dl.signalsciences.net/sigsci-cloudfoundry/sigsci-cloudfoundry_latest.tgz

Java

https://dl.signalsciences.net/sigsci-module-java/sigsci-module-java_latest.tar.gz

.NET

https://dl.signalsciences.net/sigsci-module-php/sigsci-module-php_latest.tar.gz

Node.js

https://dl.signalsciences.net/sigsci-module-nodejs/sigsci-module-nodejs_latest.tgz

Python

https://dl.signalsciences.net/sigsci-module-python/sigsci-module-python_latest.tar.gz

IIS

https://dl.signalsciences.net/sigsci-module-iis/sigsci-module-iis_latest.zip

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.

Upgrading the Apache module on Ubuntu/Debian systems

1. Upgrade the Apache module package

   ```
   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

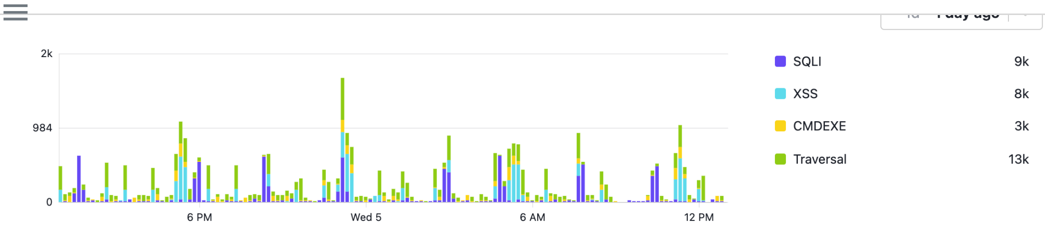
   After upgrading the module you’ll need to restart your Apache service

Data Storage and Sampling

When our agent sends requests to our collectors, we store two types of data: **timeseries data** and **individual request data**.

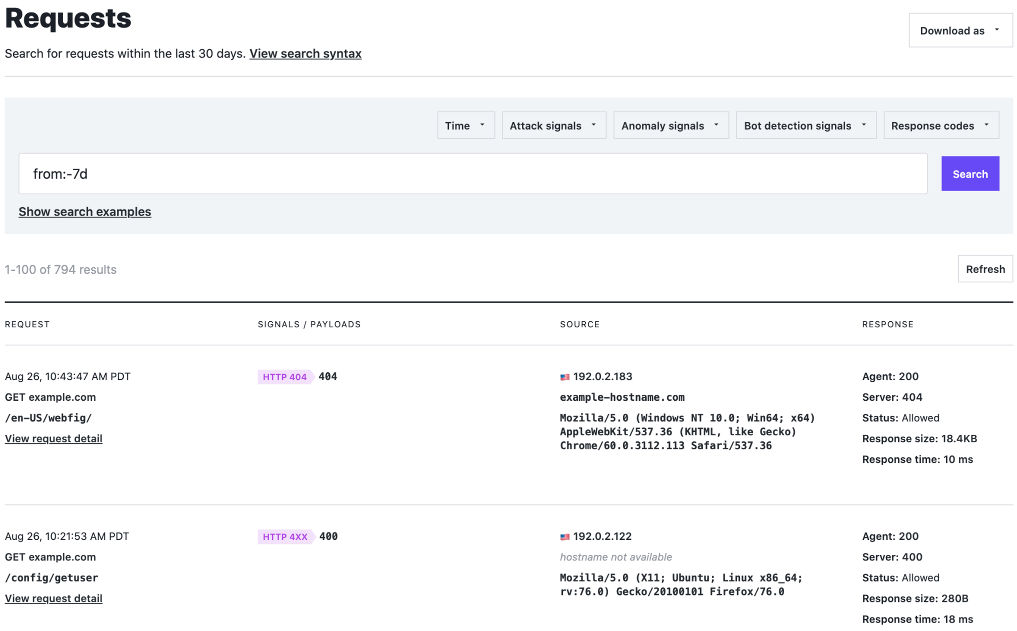
Timeseries data

Timeseries data counts the number of signals (e.g., XSS, SQLi, 404s) observed per minute, while individual request data includes individual records of anonymized requests. Timeseries data powers graphs visible throughout the product, as well as metrics such as tallies of request types.



Individual request data

While all timeseries data is stored and available in the product, a representative sample of individual request data is stored. Individual request data provides detailed information about specific requests, such as the originating IP address and request parameters:



Requests

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

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

What data does Signal Sciences store?

All timeseries data sent to our collectors (powering graphs and metrics throughout the product) is stored.

Our product has three storage categories, depending on the types of signals the requests have been tagged with. The categories are **all**, **sampled**, and **timeseries only**.

All - All requests matching this storage category will be stored and available for reference throughout the console.

Sampled - A random sample of requests matching this storage category will be stored and available for reference throughout the console.

Timeseries only - Requests matching this storage category will not be stored. Timeseries data for all signals tagged to the request will be stored and visible in the dashboards, charts, etc.

> **Note:** Timeseries-only data storage category is only available on agents 3.12 and above. Matching requests processed on earlier agents will be processed according to the Sampled data storage category.

Request signal type	Description	Storage category
Individual requests containing attack signals	Any requests containing 1 or more attack signals (e.g., SQLi, XSS, etc.)	All
Individual requests containing CVE signals	Any requests containing 1 or more CVE signals applied by virtual patching templated rules	All
Individual requests containing only anomaly signals	Requests that contain only anomaly signals (e.g., 404, Tor traffic) but no attack or CVE signals	Sampled
Individual requests containing custom signals	Requests containing custom signals but no attack or CVE signals. See Custom Signals for more information about creating and using signals.	Sampled

Individual requests containing only API and/or ATO templated rules signals, known as informational signals	Requests which are tagged with only a specific set of API and/or ATO templated rules signals, and no custom, anomaly, attack, or CVE signals	Timeseries only

> **Note:** Any requests containing at least one attack or CVE signal will be stored, including requests that also have anomaly, informational, or custom signals.

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

Data Redactions

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 attack or anomaly in a request, the agent sends only the individual parameter of the request which contains the attack payload, as well as a few other non-sensitive or benign portions of the request (such as client IP, user agent, URI, etc.) The entire request is never sent to the Signal 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.

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:

```
foobar=<script>alert(1)</script>
```

Sensitive parameters

If a request contains an attack or anomaly, and also contains sensitive data in commonly-used parameter names, Signal Sciences will redact 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`
- `orpassword_confirmation`

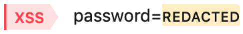
The initial request:

```
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:

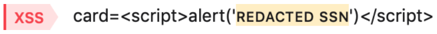
- **Credit card numbers:** values like `4111-1111-1111-1111` become `0000-0000-0000-0000`
- **Social security numbers:** values like `078-05-1120` become `000-00-0000`
- **GUIDs:** values like `3F2504E0-4F89-41D3-9A0C-0305E82C3301` become `0000000-0000-0000-0000-000000000000`
- **Bank account (IBAN) 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:

Within the console we clearly display which patterns have been redacted. Redacted patterns are replaced with the word `REDACTED` highlighted in yellow.


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. Specify additional fields for redaction by following these steps:

1. Go to **Rules** > **Redactions** and click on **New redaction**
2. Enter the field to be redacted
3. Select the type of field to be redacted (Request Parameter, Request Header, or Response Header)
4. Click **Create 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. Go to the Agents page in your console, click on an Agent ID and navigate to the Requests tab. It's also available from the agent itself by running it with the `debug-log-uploads=0|1|2` command line argument. Additional information about agent configuration options can be found in our Agent Configuration guide.

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", "*/*" ],
       [ "Connection", "keep-alive" ],
       [ "Cookie", "" ],
       [ "User-Agent", "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_4) AppleWebKit/600.7.12 (KHTML, like Gecko)
       [ "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
```

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",
  "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",
  "WAFResponse": 200,
  "ResponseCode": 404,
  "ResponseMillis": 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).

```
{
  "WAFResponse": 406,
  "RequestID": "55c24b96ca84c02201000001",
  "RequestHeaders": [
      [ "X-SigSci-Tags", "SQLI" ]
  ]
}
```

The request should be blocked, and at the end of the request, and `UpdateRequest` message.

```
{
  "RequestID": "55c24b96ca84c02201000001",
  "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 (i.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`

See system signals for a full list of 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.

Signal Sciences Help Center

Support

Contacting Support

Our support hours are Monday - Friday, 4:00 am - 7:00 pm PT (except US public holidays.)

For urgent after hours support, mark your priority as “urgent”.

Priority	Severity
Low	General questions, feature requests
Normal	Minor impact, cosmetic issues, non-prod
High	Notable impact on the service
Urgent	Critical, immediate impact to production

Click here to open a new support ticket.

Status Page

Incident information (outages, lack of functionality, etc) is always published as quickly as possible on our status page:

https://status.signalsciences.net/

Subscribe to status page updates via email, text, webhook, and RSS directly on the status page by clicking on **Subscribe to Updates**.