TX Protect

Administration

Version 3.20, 2025-01-10

Table of Contents

Introduction

The ThreatX platform has a number of configuration, user, sensor, and other settings that can be managed by the ThreatX SOC, by your local administrator, or a combination of both.

The administrative settings can be accessed from the ThreatX user interface or from the API. Your ThreatX account must have write access to perform these tasks.

Allow, Deny, and Block lists

An entity in the following lists is denied temporarily blocked, or always allowed to interact with any of your sites.

Lists

Blacklist
An entity in the list is prevented from interacting with any of your sites.

Blocklist
An entity in the list is prevented from interacting with any of your sites. The block lasts for 30 minutes from the time the entity was added to the list. All requests made while the threat is blocked are tracked for valuable threat intelligence.

Whitelist
An entity in the list cannot be blocked or denied.

Once added to the Blacklist or Whitelist, the entity remains there permanently until it is manually removed. An administrator or ThreatX SOC can add an IP address or CIDR range, or manually remove an entity from the list.

See: Blocking

Firewall Settings

You can view the CNAME provided for your tenant. The ThreatX WAF is SNI (Server Name Indication) aware and refers to the hostname provided in each request when visualizing and routing traffic. Request traffic for each of your sites is routed to the backend you defined for that site on the site's details page.

Risk-Based Blocking Settings

Risk-Based Blocking Timeout
Determines the length of time a threat is blocked. Applies only to those threats that are blocked automatically.

Risk-Based Blocking Threshold
Sets the Risk Level score. Any threat that meets or exceeds the score is blocked automatically.

Block Embargoed Countries
When checked, any traffic from a country that is on the USA embargo list is blocked automatically.

Block Tor Exit Nodes
: These are the gateways where encrypted Tor traffic hits the Internet. When configured, all incoming traffic from a TOR Exit node is not allowed.

_ See: Firewall Settings

Site Settings

The ThreatX sensor operates as a reverse proxy and is designed to monitor and act on incoming HTTP and HTTPS request traffic to prevent attacks and unwanted activity from reaching your web application and API servers. The backend you define for each site can be a single CNAME or a list of IPs, wherever traffic can be properly routed to reach your origin servers.

Site Settings

An administrator or ThreatX SOC can configure the following settings.

Listener
: Settings include host name, SSL/TLS, redirect traffic, HTTP2.

Backend
: Backend configuration for the connection of sites to sensors can be specified as a single hostname or CNAME, or a comma-separated list of IP addresses.

Blocking modes
: Determine whether threats can be automatically blocked by risk-based blocking or by rules when it is an obvious hostile attack. Additionally, you can enable users to add IP addresses to the blocked or deny lists.

Caching configuration
: Enable or disable static or dynamic caching. For more information about caching, see Performance

Proxy configuration
: Configure the proxy settings, such as maximum request body size, proxy read timeout, proxy send timeout, set real IP from, and custom response headers.

Site group
: You can assign a site group to limit which users can access the site configuration and its associated data.

See: Managing Sites

Sensors

Sensors can be managed by your local administrator or the ThreatX SOC. If managed locally, you need to provide a Sensor API Key, which is required to authenticate to the ThreatX cloud infrastructure.

The sensor IP addresses are available in the ThreatX user interface. These addresses must be added to the whitelist in your environment to ensure traffic can reach your application.

See: Sensors

Notifications

You can configure users to receive notifications on various events relating to threats, rule matches, changes to the allow, deny, and block lists, and more. Notifications are typically sent by email, but you can configure a webhook notification to another app, such as Slack.

You can subscribe to the Threat X Maintenance and System Status Notifications website for messages regarding scheduled maintenance windows and any issues that might impact your ThreatX services.

See: Notifications

Access Management

User Accounts

To access the ThreatX platform, you need to add user accounts. You can configure analyst accounts to be read-only where the users can access all analytical data. For administrators, you can grant write permission where the users can configure various settings as needed.

As needed, you can restrict users to access a specific site only.

You can add, edit, or remove user accounts from the ThreatX Dashboard or the API.

See: Managing User Accounts

Audit Log

The ThreatX audit feature logs events, such as updating users, updating sites, and adding IP addresses to whitelists and blocked lists. The audit log lists all events by category and actions. As opposed to the Log Emitter, the audit log focuses mostly on user actions.

The audit log is available from the user interface or by using API.

See: Audit Log

API Access

The ThreatX platform uses a RESTful API and supports a full set of application capabilities that can be used ad-hoc, in scripts, and in automation toolsets, such as SOAR. Advanced administrators can use the API to prevent, allow, or block an IP address or CIDR range with an API command. Other common uses include creating and managing user accounts, provisioning new sites to be protected, and managing certificates.

See: API Access

Observability

Analytical events

Notifications

Dashboard	**Settings › Notifications**
ThreatX API	`api.threatx.com/tx_api/v2/subscriptions`

You can configure users to receive notifications on various events relating to threats, rule matches, changes to the IP allow, deny, and block lists. Notifications are typically sent by email, but you can configure a webhook notification to another app, such as Slack.

You create notifications in the ThreatX user interface by navigating to **Settings › Notifications**. You can add a notification or edit an existing notification.

Notification Settings

Name
A unique name to identify this notification. It may only contain the following characters:

- Lowercase letters
- Hyphens
- Numbers

Enabled
Notifications are sent to the specified target as configured.

Event Subject Area
You can send all events or limit the notification to specific types of events:

- All Event Subjects
- Entity-related Events
- Rule-related Events
- WAF List *(Blacklist, Blocklist, Whitelist)*
- Entry-related Events

Event Incident
If you specify a type of event in the *Event Subject Area*, you can further limit the type of event. However, no other selection is available if you choose **Event Subject Area › All Event Subjects** .

Event Source
You can limit notifications to alerts from a specific ThreatX system component.

- All Event Sources
- Automatic, risk engine-initiated Events
- Manual, API-initiated Events

Limit By Site
You can limit notifications to alerts for events that affect one of a list of sites. This is only

available when you select one of the following two options:

- **Event Subject Area › All Event Subjects**
- **Event Subject Area › Entity-related Events**.

Limit By WAF List Type

|You can limit notifications to events for one or more IP lists. Only available when you select **Event Subject Area › WAF List ›** ***Blacklist, Blocklist, Whitelist***

Notification Targets: Method

- Email
- Webhooks

You will need to configure the app to receive the notifications.For example, you can send notifications to Slack as described in their Incoming Webhooks article.

ThreatX maintenance and system status

You can view and subscribe to notifications for scheduled maintenance windows and any issues that might impact your ThreatX services at ThreatX Status.

Receiving event logs

The ThreatX Log Emitter exports event logs from the ThreatX platform to your log receiver and SIEM.You can use the logs in your investigations and to trigger events in your chosen log management solution.

Features

The Log Emitter forwards full details for all block, match, and audit events.

The logs are pushed are in JSON lines format over a TCP connection that is encrypted, and optionally authenticated, over TLS.If the Log Emitter subscription becomes suspended, the Log Emitter service queues your logs for delivery upon successful re-connection, and periodically attempts to re-establish a connection.

In case a Log Emitter subscription becomes suspended, the Log Emitter service queues your logs for delivery upon successful re-connection, and periodically (every half hour) attempts to re-establish a connection.

Once the Log Emitter re-establishes a connection for a previously suspended subscription, all queued log events are sent to the configured receiver. If the Log Emitter subscription cannot be resumed after several retries, this might indicate a configuration error or log receiver error.

Configuring a Log Emitter

To receive logs, you create an instance of the Log Emitter and then encrypt the connection between the Log Emitter and your receiver.

Prerequisites for Log Emitter

- [ ] You must whitelist the following IP address ranges:
    - `169.44.76.160/28`

- `169.61.156.0/28`
- `158.85.41.64/27`

- ☐ Verify that your log receiver or SIEM can parse **JSON** lines
- ☐ **Allow** inbound connections your log receiver's **listening port** if it is behind a firewall
- ☐ Validate that your log receiver's hostname resolves to a **public** IP address.

A certificate is needed to encrypt the connection between the ThreatX Log Emitter service and your log receiver. You must generate a self-signed or CA signed certificate **for the hostname where the log receiver receives logs** in one of two ways:

- A Simple configuration that sets up an encrypted TCP connection without authentication.
- An Advanced configuration adds mutual server and client certificate authentication to your configuration

Simple configuration

Generate a new certificate

```
$ openssl req \
  -x509 \
  -days 365 \
  -nodes \
  -newkey rsa:4096 \
  -keyout logreceiver.key \
  -out logreceiver.crt
```

Use the `logreceiver.key` and `logreceiver.crt` files on your log receiver.

Be sure to configure your on-premises log receiver to accept TCP connections with TLS encryption.

Advanced configuration

You can further secure the TCP encrypted connection between the ThreatX Log Emitter and your log receiver by adding **mutual server and client certificate authentication**:

1. Use the Certificate Authority of your choice to create an SSL/TLS certificate and private key for the ThreatX Log Emitter.
2. Create a valid server certificate and key in PEM format to install on the on-premises log receiver. The the `CN` of the server certificate **must match** the public domain name of the log receiver.
3. Configure your on-premises log receiver to accept TCP connections with TLS encryption and client certificate authentication.

Adding a Log Emitter

Dasboard	**Settings › Log Emitter** then click **[Add Log Emitter]**
ThreatX API	`api.threatx.com/tx_api/v2/subscriptions`

Configure the settings as described in the following table. Click **Save** when done.

Table 1. Log Emitter Settings

Setting	Value	Description
Name	Text input	Unique name to identify the Log Emitter.
Hostname	Text input	Host name of your log receiver.
Port	Text input	Port number that your log receiver listens on
Send Client SSL/TLS Credentials to Log Receiver	Check box	Check to upload your SSL/TLS certificate and key
Verify Log Receiver SSL/TLS Certificate	Check box	When checked, the Log Emitter verifies the SSL/TLS certificate provided by the log receiver before sending log data.
Enabled	Check box	Start sending logs to your log receiver.

Description of logs

BlockEvent Log Type

The **BlockEvent** log type provides full details on requests that were blocked by the ThreatX sensor.

BlockEvent Message Example

```
{
    "message": "www.examplesite.net/example_uri",
    "msg_type": "BlockEvent",
    "timestamp": "2020-12-18T14:05:52Z",
    "user_agent": "Mozilla/5.0 (X11; Linux x86_64;rv: 82.0)",
    "dst_host": "www.examplesite.net",
    "uri": "/example_uri",
    "args": "oneequals1--",
    "request_id": "d3f02fff5db4824d83d145fad1258959",
    "random_id": null,
    "tls_fingerprint": null,
    "cookie": null,
    "js_fingerprint": null
}
```

Table 2. BlockEvent Type Properties

Name	Details
`message`	Complete target path of the request, including hostname and URI.
`msg_type`	Request was blocked at the individual request level or due to the entity being blocked at the Risk level.
`timestamp`	UTC timestamp of the request
`user_agent`	UserAgent presented by the entity making the request
`ip`	IP address presented by the entity making the request

Name	Details
`dst_host`	Target hostname of the request
`uri`	Target path of the request
`args`	Arguments (if any) provided in the request in `www-url-encoded` form
`request_id`	Unique identifier assigned to each request by the ThreatX platform
`random_id`	Additional unique identifier assigned to an entity by the ThreatX platform. *See the **note**.*
`tls_fingerprint`	TLS fingerprint (if any) associated with the entity making the request
`js_fingerprint`	Additional Unique identifier assigned to an entity by the ThreatX platform. *See the **note**.*

MatchEvent Log Type

The **MatchEvent** log type provides full details on requests that matched custom or common rule definitions when examined by the sensor.

MatchEvent Message Example

```
{
    "message": "www.examplesite.net/example_uri",
    "msg_type": "MatchEvent",
    "timestamp": "2020-12-18T14: 05: 52Z",
    "request_id": "d3f02fff5db4824d83d145fad1258959",
    "user_agent": "Mozilla/5.0 (X11; Linux x86_64; rv:82.0)",
    "matches": [
        {
            "id": 202202,
            "description": "SqlAnalyzer: SQLi detected in form/args,
sql_ids: 1",
            "classification": "SqlInjection",
            "state": "Exploitation",
            "contrib_score": 100,
            "risk": 70,
            "blocking": true,
            "beta": false
        }
    ],
    "ip": "222.222.222.222",
    "dst_host": "www.examplesite.net",
    "uri": "/",
    "args": "oneequals1--",
    "status_code": 0,
    "ssl": false,
    "risk": 70,
    "request_method": "GET",
    "content_type": null,
    "content_length": 0,
    "response_length": null,
    "upstream_response_time": null,
    "postblock_event": false,
    "random_id": 0,
    "tls_fingerprint": null,
    "cookie": null,
    "js_fingerprint": 0,
    "created": "2020-12-18T14: 05: 52.174+00: 00"
}
```

Table 3. MatchEvent Type Properties

Field	Details
message	Complete target path of the request, including hostname and URI.
msg_type	[MatchEvent] Request matched a custom or common rule.
timestamp	UTC timestamp of the request
request_id	Unique identifier assigned to each request by the ThreatX platform.
user_agent	UserAgent header value presented by the Entity making the request.
list<matches>	Values following this field provide specific information about why the rule that the request's behavior matched: • description - Description or name of the matched rule. • classification - Industry-defined classification of the attack described in the rule. • state - Industry-defined goal of the attack described in the rule. • contrib_score - Reserved for ThreatX internal use. • risk - Amount of risk that the matched rule contributes to the requesting entity's risk score. • blocking - Rule blocked a request [True] or allowed the request [False]. • beta - Reserved for ThreatX internal use.
ip	IP address presented by the entity making the request.
dst_host	Target hostname of the request.
uri	Target path of the request.
args	Arguments (if any) provided in the request in www-url-encoded form.
status_code	Status code that the request received from the upstream server.
ssl	Request was transmitted over an HTTPS connection [True] or an HTTP connection [False].
request_method	Request method type [GET] or [POST]
content_type	MIME content type/subtype (if any) presented in the request.

Field	Details
`response_length`	Length in bytes (if any) that the request received from the upstream server.
`upstream_response_time`	Length of time in milliseconds (if any) that it took the upstream server to respond to the request.
`postblock_event`	Request was submitted after a risk-based block was applied to the entity `[True]`.
`random_id`	Additional Unique identifier assigned to an entity by the ThreatX platform. *See the **note**.*
`tls_fingerprint`	TLS fingerprint (if any) associated with the entity making the request.
`cookie`	Additional Unique identifier assigned to an entity by the ThreatX platform. *See the **note**.*
`js_fingerprint`	Additional Unique identifier assigned to an entity by the ThreatX platform.
`created`	Timestamp of the request

Returns a null value except when logging an interrogation event. For information on interrogation, contact the ThreatX SOC.

Troubleshooting the Log Emitter

The following procedures describe basic checks that you can perform while troubleshooting your Log Emitter configuration.

Send test logs

Send a test log to verify that the server and client certificates are correctly generated and installed by running the following `openssl` command.

If you do not see **DONE** at the end, there is an issue with network connectivity or with the server or client certificates.

Self-signed certificates

```
$ echo '{"message":"test1"}' | openssl s_client \
    -servername <logreceiver.yourdomain.com>    \
    -connect <logreceiver.yourdomain.com:12345>
```

Mutual certificate authentication

```
$ echo '{"message":"test1"}' | openssl s_client \
    -servername <logreceiver.yourdomain.com>    \
    -connect <logreceiver.yourdomain.com:12345> \
    -cert logemitterclient.crt                  \
    -key logemitterclient.key
```

Your log receiver must be able to receive TCP data at the hostname provided to the Log Emitter. **This requires a publicly resolvable hostname.**

Verify incoming TCP data

Use `tcpdump` to verify that the log receiver is receiving TCP traffic on the correct port.

```
$ `tcpdump` port 12345
```

Even if you see traffic via `tcpdump`, you still need to ensure that any host-based firewall, such as `iptables`, is configured to allow the incoming traffic.

Dump incoming logs to a file (Logstash)

In Logstash, you can create a file output so you can quickly see if it is receiving the logs from the Log Emitter. Add the following fragment to your Logstash configuration file and restart Logstash.

logstash-fragment.json

```
output {
    file {
        path => "/tmp/threatx-raw.log"
    }
}
```

Handshake failed error

If you use a **self-signed certificate** and you receive this error message:

```
the handshake failed: error 1416F086: SSL/TLS Routines:
tls_process_server_certificate:certificate verify
failed:../ssl/statem/statem...cint.c:1915:: self signed certificate
```

The Log Emitter **Send Client SSL/TLS Credentials to Log Receiver** option might be enabled. When this option is enabled, the Log Emitter uses the provided SSL/TLS credentials to authenticate itself to the log receiver, which will of course fail when using a self-signed certificate.

Resolve the issue by deselecting the **Send Client SSL/TLS Credentials to Log Receiver**. Then, click **[Restart Log Emitter]**.

Testing for vulnerabilities

Introduction

The ThreatX four-stage blocking strategy is designed to reduce false positives while preventing malicious behavior from reaching your sites. When Request-Based blocking is enabled, the sensor blocks any standalone malicious request. When Risk-Based blocking is enabled, the sensor issues a series of timed block periods to any entity that exhibits persistent suspicious or malicious behavior, leading to a permanent blacklisting if the behavior continues. During a 30-minute Block period or while an entity is blacklisted, all requests from that entity are blocked from reaching the site.

When testing for vulnerabilities against your internal applications, the IP addresses of your penetration testers should be added to the whitelist before testing, and removed after testing is complete.

When testing for vulnerabilities in the sensor, the IP addresses of your penetration testers should not be added to the whitelist.

To add an IP address to the whitelist:

1. Navigate to **Settings › iWAF**.
2. In the iWAF Settings page, click the **[Whitelisted IP addresses]** tab.
3. Click the **[Add Entry]** button.
4. In the **Add Whitelist Entry screen**, enter the IP address.
5. Enter the reason for adding the IP address.
6. Set the **Expiration**. Typically, you choose **Never** but you do need to remove the address from the list when done testing.
7. Click **[Submit]**.

When done testing, remove the address by opening the **Whitelisted IP addresses** tab and click the **Remove** button in the entity's row.

Recommended Tools and Methodologies

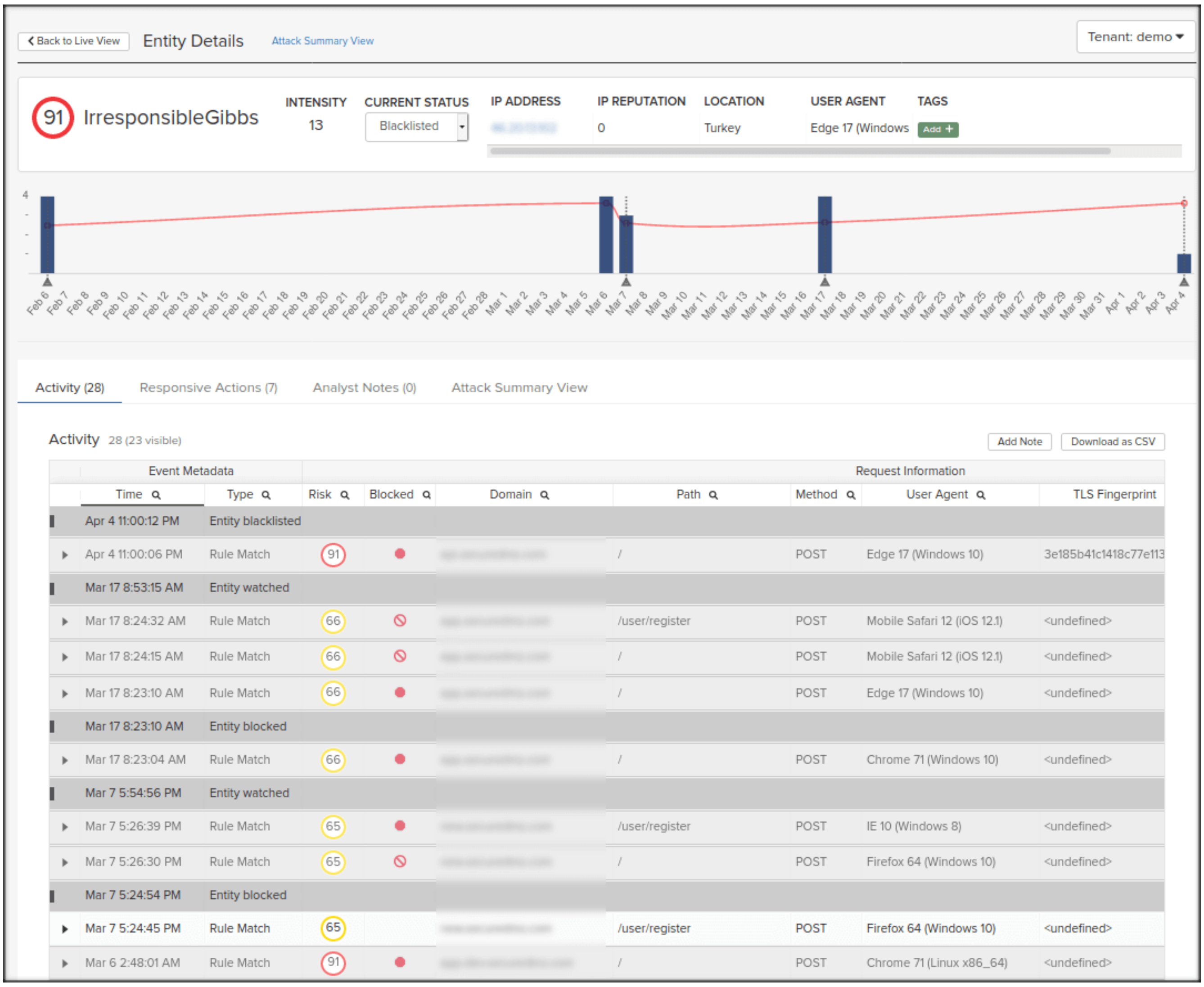
Scanners

Scanners, such as ZAP and Burp, can be a useful tool for testing the ThreatX Request and Risk-based blocking capabilities. However, they are likely to be blocked quickly and sent to the blacklist.

Leverage multiple IP addresses

When attacking the ThreatX sensor with a single IP address, that IP address accumulates risk and is delivered a series of Risk and Request-Based blocks before being placed on the blacklist. The entity associated with that IP address can be removed from the blacklist, but the associated Risk Level from that entity does not reset to "0" upon removal. An entity's Risk Level can be reduced over time by demonstrating a reduction in suspicious behavior or malicious attack attempts. Try leveraging several IP addresses or ranges when pen testing the ThreatX sensor.

You can see when the IP address is blocked from the ThreatX user interface. In the following screenshot, the Gray requests were blocked from reaching the application. The White request was allowed through as it did not contain a standalone, viable attack or high-risk behavior.



< Back to Live View Entity Details Attack Summary View Tenant: demo

91 IrresponsibleGibbs

INTENSITY	CURRENT STATUS	IP ADDRESS	IP REPUTATION	LOCATION	USER AGENT	TAGS
13	Blacklisted	[redacted]	0	Turkey	Edge 17 (Windows	Add +

Activity (28) Responsive Actions (7) Analyst Notes (0) Attack Summary View

Activity 28 (23 visible) Add Note Download as CSV

	Event Metadata		Request Information						
	Time	Type	Risk	Blocked	Domain	Path	Method	User Agent	TLS Fingerprint
	Apr 4 11:00:12 PM	Entity blacklisted							
▸	Apr 4 11:00:06 PM	Rule Match	91	●	[redacted]	/	POST	Edge 17 (Windows 10)	3e185b41c1418c77e113
	Mar 17 8:53:15 AM	Entity watched							
▸	Mar 17 8:24:32 AM	Rule Match	66	⊘	[redacted]	/user/register	POST	Mobile Safari 12 (iOS 12.1)	<undefined>
▸	Mar 17 8:24:15 AM	Rule Match	66	⊘	[redacted]	/	POST	Mobile Safari 12 (iOS 12.1)	<undefined>
▸	Mar 17 8:23:10 AM	Rule Match	66	●	[redacted]	/	POST	Edge 17 (Windows 10)	<undefined>
	Mar 17 8:23:10 AM	Entity blocked							
▸	Mar 17 8:23:04 AM	Rule Match	66	●	[redacted]	/	POST	Chrome 71 (Windows 10)	<undefined>
	Mar 7 5:54:56 PM	Entity watched							
▸	Mar 7 5:26:39 PM	Rule Match	65	●	[redacted]	/user/register	POST	IE 10 (Windows 8)	<undefined>
▸	Mar 7 5:26:30 PM	Rule Match	65	⊘	[redacted]	/	POST	Firefox 64 (Windows 10)	<undefined>
	Mar 7 5:24:54 PM	Entity blocked							
▸	Mar 7 5:24:45 PM	Rule Match	65		[redacted]	/user/register	POST	Firefox 64 (Windows 10)	<undefined>
▸	Mar 6 2:48:01 AM	Rule Match	91	●	[redacted]	/	POST	Chrome 71 (Linux x86_64)	<undefined>

ThreatX API Reference

Introduction

The ThreatX platform uses a RESTful API and supports a full set of application capabilities that can be used ad-hoc, in scripts, and in automation tool sets. Common uses include creating and managing user accounts, provisioning new sites to be protected, and managing certificates.

To access the ThreatX API, you must know your **tenant name** and you must have an **API key** to authenticate and create a session token. See Generating and revoking API keys

For details about the API endpoints and commands, see the API Reference Guide (requires a ThreatX account to access).

Login

The `login` command uses an *API key* to return a *temporary access token* that authenticates your session.

The **api_key**, created within the ThreatX user interface (**Settings › API Keys**), is used within the request for the `api_token` parameter. The response then provides a unique and temporary `access_token` string to be used in further endpoint commands.

Table 4. Login Command Object

Parameters	Type
`“command”: “login”`	String
`“api_token”: “<api_key>”`	String

Example 1. Login Command Usage

Request

```
$ curl {url-txapi}/login \
  --header 'Content-Type: application/json' \
  --data @- <<EOF
{
    "command": "login",
    "api_token": "<api_key>"
}
EOF
```

Response

```
{
"Ok": {
    "status": true,
    "token": "<access_token>"
  }
}
```

List Customers

The `list` command returns the details of all tenants authorized for the current API access token.

Table 5. Command Object

Parameters	Type
`“command”: “list”`	String
`“token”: “<access_token>”`	String

Example 2. List Command Usage

Request

```
$ curl {url-txapi}/customers \
  --header 'Content-Type: application/json' \
   --data @- <<EOF
{
  "command": "list",
  "token": "access_token>"
}
EOF
```

Replace `<access_token>` with your access token from the last command.

Response

```
{
"Ok": [
   {
   "name": "testco",
   "contact_email": "alice@testco.com",
   "description": "Tesco tenant",
   "active": true,
   "autoblock_threshold": 70,
   "autoblock_timeout": 3600,
   "block_embargo": true,
   "ssl_ciphers": null,
   "notify_threshold": 100,
   "sso": null,
   "allow_super_admin_users": true,
   "allow_channel_admin_users": true,
   "tenant_admin_default": null,
   "uuid": "<tenant_uuid>"
   }
  ]
}
```

Update Customer

The `update` command updates a specified tenant.

The SSO object is used to configure the SSO parameters. All other parameters to the customer object should not be modified when configuring SSO.

Table 6. Command Object

Parameters	Type
`“command”: “update”`	String
`“token”: “<access_token>”`	String
`“name”: “<tenant_name>”`	String
`“customer”: {}`	CustomerObject

Table 7. Customer Object

Parameters	Type
`“name”: “<tenant_name>”`	String
`“contact_email”: “<email_address>”`	String
`“description”: “<key_description>”`	String
`“active”: true / false`	Boolean
`“autoblock_threshold”: <entity_risk>`	Integer
`“autoblock_timeout”: <timeout_seconds>`	Integer
`“sso”: {}`	SSO Object

Table 8. SSO Object

Parameters	Type	Description
`enabled`	Boolean	When true, users belonging to the tenant are allowed to sign in to the ThreatX user interface using SSO.
`required`	Boolean	When true, users are required to use SSO to sign in to the ThreatX user interface.

Parameters	Type	Description
`saml_metadata_url: “<saml_url>”`	String	IDP metadata URL or file. See the Prerequisites.

Example 3. Update Command Usage

Request

```
$ curl {url-txapi}/customers \
  --header 'Content-Type:  application/json' \
   --data @- <<EOF
{
     "command": "update",
     "token": "<login_token>",
     "name": "testco",
     "customer": {
         "name": "testco",
         "contact_email": "alice@testco.com",
         "description": "Testco tenant",
         "active": true,
         "autoblock_threshold": 70,
         "autoblock_timeout": 3600,
         "block_embargo": true,
         "ssl_ciphers": null,
         "notify_threshold": 100,
         "allow_super_admin_users": true,
         "allow_channel_admin_users": true,
         "tenant_admin_default": null,
         "sso": {
             "enabled": true,
             "required": false,
             "saml_metadata_url":
"https://login.microsoftonline.com/daad3805-fde6-4334-817f-
82c723533123/federationmetadata/2007-06/federationmetadata.xml"
          }
    }
}
EOF
```

Response

```
{ "Ok": "testco updated." }
```

List Channels

The `list` command of the `channels` endpoint returns the details of all channels authorized for the current API access token.

Table 9. Command Object

Parameters	Type
`“command”: “list”`	String
`“token”: “<access_token>”`	String

Example 4. List Command Usage

Request

```
$ curl {url-txapi}/channels \
  -H 'Content-Type: application/json' \
  --data @- <<EOF
{
    "command": "list",
    "token":" "<access_token>"
}
EOF
```

Response

```
{
  "Ok": [
    {
      "name": "test_channel",
       "require_totp_setup": null,
       "uuid": "81815E73-ABB9-4533-977B-93964B8AAB73",
       "sso": null
     }
  ]
}
```

Update Channels

The `update` command updates a specified channel.

Table 10. Command Object

`Parameters`	Type
`command`	String

`token`	String
`channel`	Channel Object

Table 11. Channel Object

Parameter	Type
`name`	String
`sso`	SSO Object

Table 12. SSO Object Attributes

Name	Type	Description
`enabled`	Boolean	When true, users belonging to the channel are allowed to sign in to the ThreatX user interface using SSO.
`required`	Boolean	When true, users in the channel are required to use SSO to sign in to the ThreatX user interface.
`saml_metadata_url`	String	IDP metadata URL or file. See the Prerequisites.

Example 5. Update Command Usage

Request

```
$ curl {url-txapi}/channels \
  --header 'Content-Type: application/json' \
  --data @- <<EOF
{
    "command": "update",
     "token": "<login_token>",
     "channel": {
     "name": "test_channel",
     "sso": {
        "enabled": true,
        "required": false,
        "saml_metadata_url":
"https://login.microsoftonline.com/daad3805-fde6-4334-817f-
82c723533123/federationmetadata/2007-06/federationmetadata.xml"
       }
     }
}
EOF
```

Response

```
{"Ok": "Channel: test_channel updated."}
```

Performance

Managing caching

Edge Caching is available if you want to take advantage of the performance and speed improvements, but do not have a caching solution of your own in place.

💥 Edge Caching Benefits

- Faster page load times for end-users
- Lower latency
- Increased load capacity and reduced application server load
- Better ratings from search engines such as Google

By default, ThreatX Edge Caching follows `Cache-Control` headers defined by the origin servers.

The ThreatX platform does **not** cache for the following response scenarios:

- `Cache-Control` is set to `Private`, `No-Cache`, or `No-Store`
- Response headers include `Set-Cookie`
- We are responding to `POST` requests

The ThreatX platform offers two types of Edge Caching: **static** and **dynamic**.

Caching can be enabled for a configured site as described in Site settings

Static caching

Static caching is configured to cache static elements such as images, CSS & JavaScript. Static caching does not store HTML pages and as a result does not enhance performance if the origin server becomes unresponsive.

Static Cache Settings

Default cache expiration	30 minutes.
Supported static file extensions	`jpg`, `jpeg`, `gif`, `png`, `ico`, `bmp`, `tif`, `tiff`, `svg`, `svgz`, `swf`, `pict`, `cur`, `doc`, `docx`, `xlsx`, `ppt`, `pptx`, `pdf`, `woff`, `woff2`, `eot`, `otf`, `js`, `ejs`, `css`
Support for non-responsive origin servers	No.
URI Specific Caching	Per-URI features can be enabled, overriding the origin server values.
Manual Cache Purging	Can be purged by ThreatX SOC upon request. Purging can be limited to a specific URI.

Dynamic caching

- Dynamic caching offers a higher level of performance, allowing caching and optimization of dynamic content.
- In some cases, cached content can be delivered even if the origin servers are unresponsive.
- The ThreatX platform caches all responses to requests made with HTTP `GET` and `HEAD` methods.
- To avoid caching dynamic pages that are rarely accessed, ThreatX sensors cache dynamic pages only after they are requested **at least 3 times**.
- Subsequent requests are served from the cache until the cache expiration defined in the Cache-Control occurs (or about 30 minutes for responses where the expiration is not defined).

Dynamic Cache Settings

Default cache time	30 minutes
Time-to-cache	Can be configured by ThreatX SOC upon request.
URL based caching	Can be configured by ThreatX SOC upon request
Supported file types	Any dynamic resources
Support for non-responsive origin servers	500, 502, 503, and 504 response codes. Can be configured by ThreatX SOC upon request.
Supported request methods	GET, HEAD
URI Caching	Per-URI features can be enabled, overriding the origin server values.
Manual Cache Purging	Can be purged by ThreatX SOC upon request. Purging can be limited to a specific URI.

Dynamic caching is a billable feature and requires an add-on license.

Managing rate limiting

The ThreatX platform rate limiting is in the form of rules in the common rule set. Rate limiting is primarily focused on count and time frame. What causes a rule to trigger when based off count and time frame is limited only by what the rules can match within the requests. For example, one rule is **10 404s in 10s**, where the rule assigns risk to an entity that receives more than 10 404 responses within 10 seconds.

As needed, the ThreatX SOC team can make custom rate limiting rules tailored for your environment. A typical use of this would be to assign risk to entities that fail logins at a login endpoint. These rate limiting rules are very customizable, including the timings (number of requests over time). These rules can be applied across the entire tenant, down to a site or group of sites, or to a single endpoint. The match criteria also have a very wide range of options such as Response Code, Request Method, Source Country/ASN, and Arguments.

Managing Sites

Introduction

A site is a web property serving API responses intended for consumption by an application. Your environment might have many sites, where some sites might not be under ThreatX protection.

You can add, edit, or remove sites with the ThreatX user interface or ThreatX API.

Site settings

Dashboard	menu:Settigs
ThreatX API	`api.threatx.com/tx_api/v2/sites`

The backend you define for each site can be a single CNAME or a list of IP addresses – wherever traffic can be properly routed to reach your origin servers.

Some of the settings are on the **Sites** page as column headers.

▼ *Expand Listener Configuration*

Table 13. Listener Configuration

Setting	Description
Host Name	Domain name protected by the sensor (for example, www.example.com). It must be unique across all configured sites and cannot contain uppercase letters. Once created, the configured hostname cannot be changed.
SSL/TLS Enabled	Allows HTTPS connections to the hostname. Use this setting to provide your own site certificate (in PEM format). The setting does not need to be enabled if using ThreatX managed certificates with Let's Encrypt. For more information, see the *Site certificates* section.
SSL/TLS Terminate Only	If set, SSL/TLS connection is terminated at the sensor and requests are sent through a proxy to the backend using HTTP.
Redirect HTTP traffic to HTTPS	If enabled, requests made to the hostname using HTTP receive a 301 response code and are redirected to the same hostname using HTTPS instead.
HTTP2 Enabled	Allows HTTP Version 2 traffic.
Wildcard Subdomains Enabled	For example, if enabled for site with “example.com” hostname, site configuration also applies to all requests sent to “subdomain.example.com”.

▼ *Expand Backend Configuration*

Table 14. Backend Configuration

Setting	Description
Origin	Location where traffic can be properly routed to reach your origin server, also called a backend. You can specify a single hostname or CNAME, or a comma-separated list of IP addresses. If you are forwarding traffic to a load balancer, supply the FQDN or IP addresses of your load balancer. The sensor forwards all benign and unblocked traffic to that load balancer.
HTTP Backend Port	Port number of the origin server or load balancer accepting HTTP traffic.
HTTPS Backend Port	Port number of the origin server or load balancer accepting HTTPS traffic.

▼ *Expand Blocking Modes Configuration*

Table 15. Blocking Modes

Setting	Description
Risk-Based Blocking	If set, any entity with accumulated risk above the risk-based blocking threshold is blocked. The threshold settings are described in Blocking.
Request Blocking	If set, individual requests that are obvious hostile attacks, as determined by the ThreatX rules, aure blocked.
Manual Action Blocking	If set, users can manually add IP addresses to the blocked list and blacklist.

▼ *Expand Caching Configuration*

Table 16. Caching Configuration

Setting	Description
Static Caching Enabled	Enables static caching. See Performance.
Dynamic Caching Enabled	Enables dynamic caching. See Performance.

▼ *Expand Proxy Configuration*

Table 17. Proxy Configuration

Setting	Description
Maximum Request Body Size	Maximum client request body in MB as read from Content-Length header. Accepts values from 1 to 1,000,000 (1MB to 1TB). Default is 1MB.
Proxy Read Timeout	Timeout in seconds for reading a response from the backend. Accepts values from 1 to 3,600 (1 second to 1 hour). Default is 90 seconds.
Proxy Send Timeout	Timeout in seconds for sending a request to the backend. Accepts values from 1 to 3,600 (1 second to 1 hour). Default is 30 seconds.
Set Real IP From Enabled	When checked, client requests override the IP address (as recognized by sensors). **Header Name**. Provides the value for the IP override; for example, “X-Real-IP” or “X-Forwarded-For”. Letters, numbers, hyphens, and underscores only. **Trusted Sources**. IP addresses of the trusted sources.
Custom Response Headers Enabled	Inserts one or more custom headers into responses, including common security headers such as Content-Security-Policy. Each custom header must have a name and value.

▼ *Expand Access Configuration*

Table 18. Access Configuration

Setting	Description
Site Groups	You can assign the site to an existing site group, which allows you to limit which users can access the site configuration and its associated data.

Adding a site

1. Use the ThreatX user interface or API to add the site and enter the configuration settings, as described in the *Site settings* section.
2. If you are not using the Let's Encrypt option for client-facing certificates, provide the SSL/TLS Certificat `.pem` file in the **SSL/TLS Enabled** site setting.
3. Once the site is available in the ThreatX user interface, cut-over DNS to direct traffic to the CNAME provided for your tenant through your DNS provider. The CNAME records are provided in the IWAF settings, as described in the *Firewall settings* section under. This can be done at your own pace.

Adding a site can impact the cost of the ThreatX platform. For information, contact the ThreatX SOC.

If you are adding multiple sites, you can add the additional sites first then cut-over DNS after.

If your DNS provider does not allow you to point to a root domain directly to a CNAME, contact ThreatX SOC to provide the sensor ingress IP addresses to use as A records.

Once your site is configured and traffic is flowing through your sensor, you should see traffic populated in the dashboard. If you do not see any traffic, contact the ThreatX SOC.

On-Boarding Checklist

✔☐ TX Protect Pre-Installation Checklist

Check the box next to any of the requirements that apply to your application…

- ☐ Processes requests with well-formed SQL queries *(E.g., some help desk or bug-tracking software)*
- ☐ Processes requests with well-formed HTML *(E.g., some content management systems)*
- ☐ Requires Two-way SSL/TLS (client authentication)
- ☐ Uses web sockets
- ☐ Requires a specific TLS version or cipher suite restriction *(Default is TLS 1.2 and 1.3)*
- ☐ Supports unique business requirements necessitating custom WAF rules *(E.g, blocking traffic from foreign countries)*
- ☐ Is located behind a firewall or content delivery network (CDN) in which connections from ThreatX service IP addresses would need to be explicitly allowed

If you checked one or more boxes, please contact ThreatX support for assistance with your TX Protect installation.

Site certificates

You have two options. You can use Let's Encrypt or upload your own certificate.

The ThreatX platform can manage the SSL/TLS/TLS certificates presented to your site's visitors with Let's

Encrypt. The Let’s Encrypt integration allows you to offload the overhead and management commonly associated with managing SSL/TLS/TLS certificates while ensuring that an expired certificate is never presented to your site’s visitors. For more information, contact the ThreatX SOC.

Use a Custom Certificate

To upload your own certificate using the ThreatX user interface, perform the following:

1. Navigate to **Settings › Sites**.
2. If updating a certificate for an existing site, locate the site. You can use the search icon in the **Hostname** column to locate a site. Then click **[Edit Site]**.
3. In the configuration page, enable **SSL/TLS Enabled**.
4. Click **Settings › Edit SSL/TLS credentials**
5. Paste your** Site Certificate* **Intermediate Certificate**, and your **Private Key**, in **PEM** format and in that order.
6. Click **[Save]** at the bottom of the page.

If adding a site, enter your certificate using steps 3 through 6.

To ensure the correct certificate is being presented, the ThreatX platform validates the following: fsssssfvcard domain is listed as the Common Name or in the SAN attribute within the certificate. * Current date is within the `notBefore` and `notAfter` fields. * Private key provided is the same key that was used to sign the certificate. * Formatting of the uploaded certificate chain is in the proper PEM format, without any headers present or any other characters that should not exist.

Troubleshooting

- **If any one of these criteria are not met**, you will receive an error describing the issue and the old certificate continues to be utilized.
- **If you are certain** that you have the correct certificate and key pair for the site and the certificate has not expired, and yet are still receiving an error, contact the ThreatX SOC.
- **Optionally**, you can ask a third-party test group, such as Qualys SSL/TLS Lab, to test and validate your certificate.

Site groups

Dashboard	**Settings › Site Groups**
API	`api.threatx.com/tx_api/v2/sitegroups`

You can create a site group then assign sites to a single group, which allows you to limit which users can access the site configuration and its associated data.

When creating a group, give it a name, list of sites to include in the group, and list of users that can access the sites in the group.

Managing Users and Access

Tenants and channels

Tenants are organizational units. Administrator user accounts are provisioned within these tenants. Once provisioned, users can view protected sites, attack heuristics, real-time data, and other configuration information.

Alternatively, you can have your ThreatX platform organized by channels, where a channel can contain multiple tenants. If you have channels, you can administer all users and sites within the tenants and add tenants as needed.

Audit Log

The ThreatX platform has an audit feature that logs a number of events, such as updating users, updating sites, and adding IP addresses to whitelists and blocked lists.

Table 19. Audit Log Event Categories and Actions

Category	Actions	Description
`Lists`	• new_entry • remove_entry	Lists are the whitelists and blocked lists. The **Description** column in the audit log identifies the list. The audit log monitors when IP addresses, called entries, are added to or removed from a list.
`Rules`	• new_rule, • remove_rule • update_rule	The audit log monitors whenever a rule is added, removed, or updated in the ThreatX platform.
`Sites`	• new_site • remove_site • unset_field • update_site	The audit log monitors whenever a site is added, removed, or updated in the ThreatX platform. The unset_field action occurs when a user nullifies a field within the site resource.
`Users`	• new_user • remove_user • update_user	The audit log monitors whenever a user is added, removed, or updated in the ThreatX platform.
`User Actions`	• blacklist_entity • block_entity • watch_entity • whitelist_entity	The audit log monitors whenever a user blocks an IP address, adds an IP address to the blocked list or whitelist, or chooses to watch an IP address. Whenever a user adds an IP address to a list, the Lists category shows a new_entry action.

Each column in the audit log has a search icon which you can use to search for a string in that column. The search feature is case sensitive and requires an exact match. The table lists all the action strings you can use to search for a specific action.

If you have access to the ThreatX API, you can access the audit logs. The following is an example command.

```
$ curl https://api.threatx.com/tx_api/v2/logs \
  --header 'Content-Type: application/json' \
  --data @- << EOF
{
       "command":"audit_events",
       "token":" <api_token>",
        "customer_name":"<tenant_name>",
        "limit": 100
}
EOF
```

Accessing the audit log

Dashboard	Settings › Audit Log
ThreatX API	`api.threatx.com/tx_api/v2/logs` (command: `audit_events`)

The ThreatX audit feature logs events, such as updating users, updating sites, and adding IP addresses to whitelists and blocked lists. The audit log lists all events by category and actions. As opposed to the Log Emitter, the audit log focuses mostly on user actions.

The Log Emitter also exports the audit logs.

Managing user accounts

Dashboard	Settings › Users
ThreatX API	`api.threatx.com/tx_api/v2/users`

Table 20. User Account Settings

Field	Description
`Email`	User’s email address, which is also the username used to log in. It cannot be changed once saved.
`Password Reset`	Available only when editing a user account. Click **[Send]** to send a password reset link.
`First Name`	The user’s given name
`Last Name`	The user’s surname.
`Active`	When selected, the user is active and can log in. When **not selected**, the account remains valid, but the user **cannot log in**.
`Read-Only`	When **not selected**, the user has **full write access**. Otherwise, they can make no modifications.
`Tenant Admin`	When **selected**, the user has **administration permissions** to manage users and sites.
`Channel Admin`	When **selected**, the user has administrator access to the main channel *and* **all of its tenants**. (Requires *Channel Environment architecture).*

Field	Description
`Site Groups`	Assigns the user to one or more user groups, where the user can access those sites only. If **none are selected**, the user can **access all sites**.

API Access

Generating and revoking API keys

If using the ThreatX API to access the ThreatX platform, you need first need to use an **API key** with the `login` command to receive a **session token**, after which you will be allowed to execute other commands.

To **generate** an API key:

1. Navigate to **Settings › API Keys**
2. Click **[Add API Key]** in the top right corner
3. Complete the necessary fields
4. Click **[Save]**
5. Store your new key in a safe place! ▯▯

To **revoke** an API key:

1. Navigate to **Settings › API Keys**
2. Click **[Edit API Key]** next to the API key you want to revoke
3. Click **[Revoke]**
4. Click **[Revoke]** a second time in the confirmation pop-up.

Generating and revoking sensor API keys

If you deploy sensors in your environment, you are asked to provide a Sensor API key.The sensor uses the key to authenticate to the ThreatX platform.

The *Sensor API key* is a **not** the same as the *API key* mentioned above.

To generate a Sensor API key:

1. Navigate to **Settings › Sensors › Sensor Keys**
2. Click **[Add Sensor Key]**
3. Store the key securely until you need to deploy a sensor ▯▯
    a. If at any point you no longer require a sensor key, simply delete it.

Configuring Single Sign On (SSO)

You can manage SSO configuration directly using the ThreatX API. Once SSO has been configured for a ThreatX tenant or channel, your users can sign in using your SSO identity provider, such as Okta or Azure Active Directory B2C, rather than logging in to the ThreatX web application with a username and password.

Prerequisites

- ☐ **SAML2 IDP metadata reference URL** from your SSO provider (where the most up-to-date metadata file can be found.) Consult your IDP documentation.
- ☐ Users must have accounts in **both** the **IDP** and **ThreatX platform**
- ☐ User's **email address** in the IDP **must match** that which is associated with their ThreatX username.
- ☐ An API key with **tenant or channel administrator** permissions.
- ☐ The **name** and **UUID** of your tenant or channel

Use the `Customers:list` command to retrieve the name and UUID of the tenant.

If you do not have access to your IDP metadata URL, you can alternatively provide a complete IDP metadata file. Contact ThreatX support if you want to provide an IDP metadata file instead of an IDP metadata URL.

Example 6. IDP Metadata URLs

- **Okta**: threat-x.oktapreview.com/app/exk8lh09bhSIfhupl0h7/sso/saml/metadata
- **Azure AD B2C**: login.microsoftonline.com/daad3805-fde6-4334-817f-82c723533123/federationmetadata/2007-06/federationmetadata.xml

Additional prerequisites for Channel SSO

If you are configuring your **SP Metadata URL**:

- Audience restriction setting (also called "Entity ID") in the IDP must be set to the path: `x.threatx.io/sign-in`
- **IDP metadata** must provide the *NameID* in the format: `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`

We use the email address of the user to locate users within our database

If you are configuring your **ACS URL**:

- When configuring the IDP, **the Assertion Consumer Service URL (ACS)** of our **Service Provider (SP)** is: `x.threatx.io/auth/v2/channels/\<your_threatx_channel_uuid>/acs`
- For IDPs that support Service Provider metadata, the metadata URL of our SP is: `x.threatx.io/auth/v2/channels/\<your_threatx_channel_uuid>/metadata`

Configuring SSO access

Use the following steps to configure SSO access for your ThreatX tenant and channel partners:

1. **Log into the API.** Authenticate to the API using the Login command.
2. **Gather the tenant or channel data** you need using the `Customers:list` or `Channels:list` command.
   - You need to copy the Customer or Channel Representation information response **exactly** and paste it into the body of the `Customers:update` or `Channels:updat` command with the UUID field omitted.
3. **Assemble your tenant update API request.**
4. Supply your Customer or Channel Representation information to the `Customers:update` or `Channels:update` command described in step 2. (See the examples below.)

5. Set the value of “sso” to an object and define these attributes:
    - “enabled” (`true`),
    - “required” (`false`)
    - `saml_metadata_url`
6. **Submit the tenant or channel update API request.** If it succeeds, you should see Customer Update Response or Channel Update Response.
7. Test the new configuration by using a web browser to navigate to:
    a. x.threatx.io/auth/v2/customers/<name>/saml
    b. x.threatx.io/auth/v2/channels/<name>/saml
8. You should be redirected to your SSO Identity Provider to confirm you want to authorize ThreatX Dashboard to act on your behalf.
9. Follow the prompts in your SSO Identity Provider.
10. You should be then redirected to the ThreatX Dashboard and authorized to access the system on behalf of your configured user account.
11. Single-Sign On access is now configured for your tenant.

(optional) You can now update your tenant configuration again using “required: true” to force all your users to use SSO to access the ThreatX Dashboard. This option prevents users from accessing the ThreatX Dashboard directly using the username/password authentication.

Sensors

Introduction

Dashboard	Settings › Sensors
ThreatX API	`api.threatx.com/tx_api/v2/sensor`

Sensors are managed by your local administrator or the ThreatX SOC depending on if you self-host or are hosted within the ThreatX cloud, respectively.

You can view the on-premises deployed sensors and their status from the ThreatX user interface, **Settings › Sensors**. The **[Sensor Keys]** tab lists the keys used with the sensors. You add a key only when deploying a sensor and delete a key when the associated sensor is no longer in use.

The sensor IP addresses are available in the ThreatX user interface, as described in the *Firewall settings* section. These addresses must be added to the whitelist in your environment to ensure traffic can reach your application.

If the ThreatX SOC hosts your sensors, you might notice the number of sensors fluctuate, or that an individual sensor's uptime has changed. This is because sensors are designed to be added, removed, upgraded, and replaced as needed to ensure optimal site availability and protection. For the latest information, see our release notes.

If you are contemplating deploying new sites, new tech stacks, or new architecture, contact the ThreatX SOC.

For more information about deploying sensors, see the Deployment guides in the navigation bar to the left.

Firewall settings

Dashboard	Settings › IWAF › Firewall
ThreatX API	`api.threatx.com/tx_api/v2/services`

Service IP addresses

IP addresses that represent the ThreatX sensors. These IP addresses must be whitelisted in your environment to ensure traffic can reach your application.

Sensor DNS Targets

CNAME records you can use to ensure HTTP and HTTPS traffic reaches your sensors. The CNAME provided for your tenant is all you need for all your sites. The ThreatX sensor is Server Name Indication (SNI) aware and refers to the hostname provided in each request when visualizing and routing traffic. Request traffic for each of your sites is routed to the backend you defined for that site on the site's details page.

The ThreatX `api.threatx.com/tx_api/v2/services` endpoint **list** command returns the service IP addresses, but not the CNAME records. == Configuring notifications

There are two types of notifications:

- Analytical events
- Maintenance and system status events

For information about receiving logs, Observability.

Configuring mutual TLS

Configuring mutual TLS (mTLS) configuration

This section details the configuration and setup of mutual TLS (mTLS) for secure communication between various components within our system. mTLS establishes a mutual authentication process between clients and servers, ensuring a robust and authenticated connection. You can enable mTLS in both downstream and upstream configurations.

In Transport Layer Security (TLS), the traditional setup involves the server authenticating itself to the client. However, mTLS enhances security by enabling both the client and server to authenticate each other during the communication process.

Configuring downstream mTLS

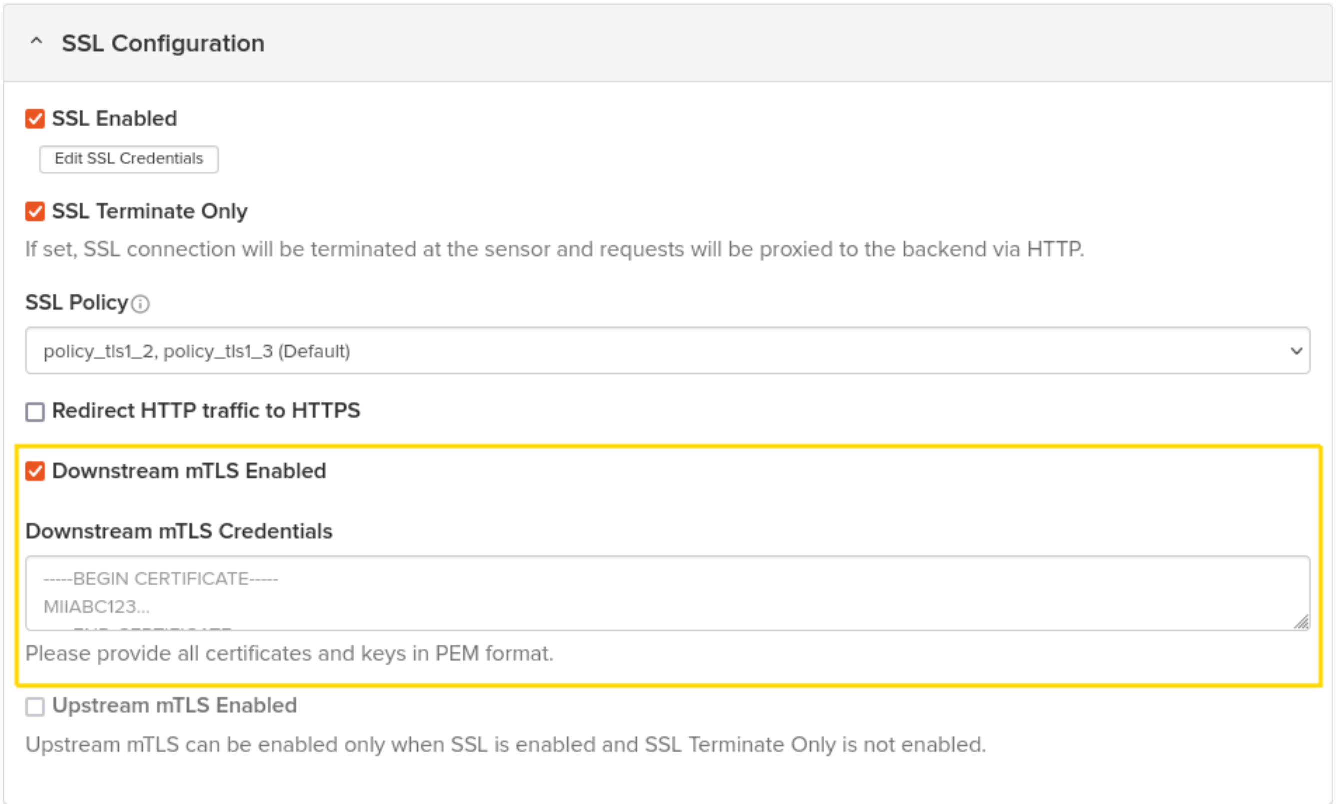
Downstream mTLS involves the WAF sensor serving as the server, authenticating incoming client connections. This setup is crucial for securing communication between end-user applications (clients) and the WAF.

Make sure your environment meets the following requirements for downstream mTLS:

- Site must be configured for TLS (HTTPS) via the SSL/TLS Enabled option on the Site Details page.
- Availability of the CA certificate used to sign client certificates, which utilizes as Downstream mTLS Credentials in the site configuration. This certificate must be in PEM format.
- Clients must be configured to send their certificates during the TLS handshake.

Configuration steps

1. Access the Site Details page through **Settings** › **Sites** › **Edit Site (for the relevant site)**
2. In the SSL/TLS Configuration section, enable Downstream mTLS by checking the designated checkbox.
3. Populate the Downstream mTLS Credentials field with the CA certificates used for validating client certificates.



SSL Configuration

☑ SSL Enabled

Edit SSL Credentials

☑ SSL Terminate Only

If set, SSL connection will be terminated at the sensor and requests will be proxied to the backend via HTTP.

SSL Policy

policy_tls1_2, policy_tls1_3 (Default)

☐ Redirect HTTP traffic to HTTPS

☑ Downstream mTLS Enabled

Downstream mTLS Credentials

```
-----BEGIN CERTIFICATE-----
MIIABC123...
```

Please provide all certificates and keys in PEM format.

☐ Upstream mTLS Enabled

Upstream mTLS can be enabled only when SSL is enabled and SSL Terminate Only is not enabled.

Configuring Upstream mTLS

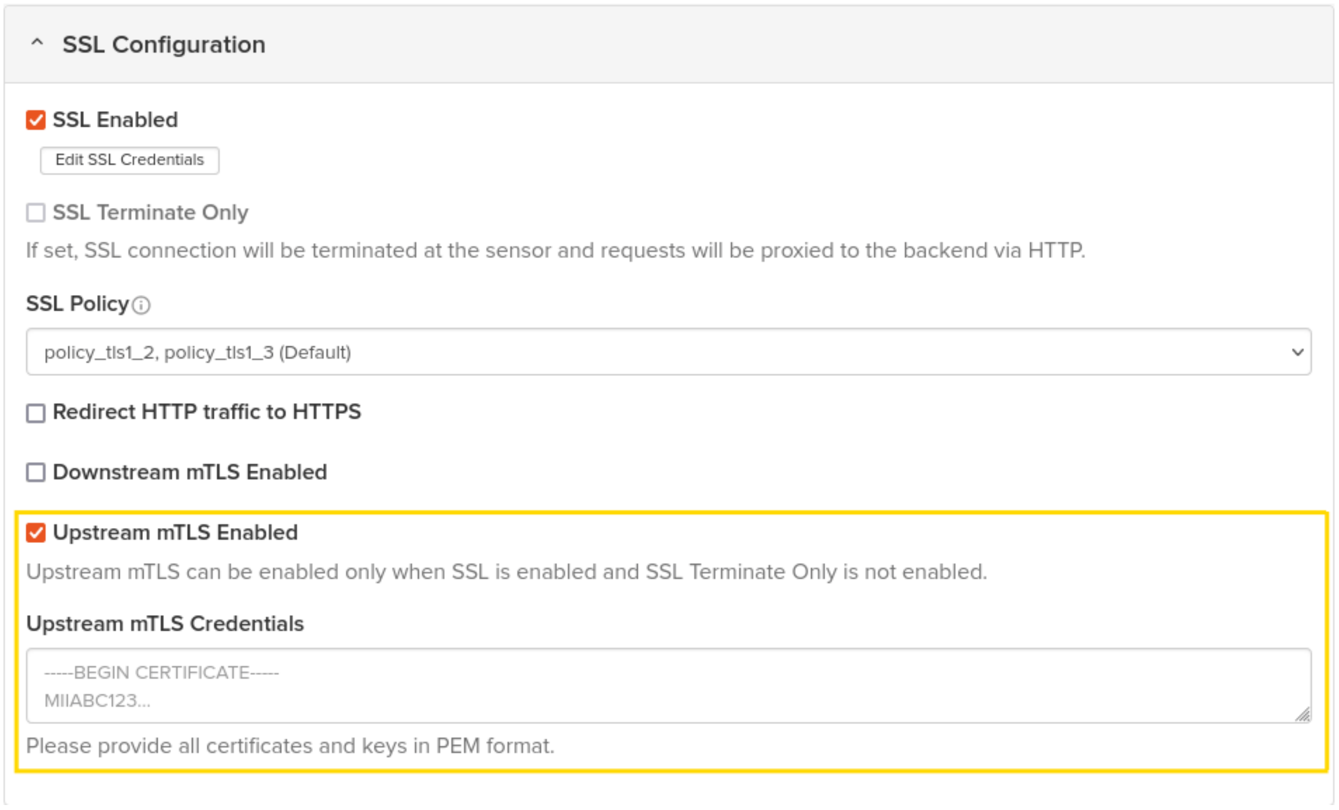
Upstream mTLS involves the WAF sensor acting as the client, authenticating itself to the origin server. This setup ensures secure communication from the WAF to the origin server.

Make sure your environment meets the following requirements for upstream mTLS:

- Site must be TLS-enabled via the SSL/TLS Enabled option in the Site Details page.
- Ensure that the TLS connection termination does not occur at the WAF sensor; the **SSL/TLS Terminate Only** checkbox should remain unchecked. The origin server should be configured for HTTPS.
- Availability of both the client certificate and private key in PEM format, used as the Upstream mTLS Credentials in the site configuration.
- The origin server should be configured to request client certificates during the TLS handshake.

Configuration steps

1. Access the Site Details page through **Settings** › **Site**, selecting **[Edit Site]** for the relevant site.
2. Within the SSL/TLS Configuration section, enable Upstream mTLS by checking the designated checkbox.
3. Populate the Upstream mTLS Credentials field with both the client certificate and private key.



SSL Configuration

☑ SSL Enabled

Edit SSL Credentials

☐ SSL Terminate Only

If set, SSL connection will be terminated at the sensor and requests will be proxied to the backend via HTTP.

SSL Policy

policy_tls1_2, policy_tls1_3 (Default)

☐ Redirect HTTP traffic to HTTPS

☐ Downstream mTLS Enabled

☑ Upstream mTLS Enabled

Upstream mTLS can be enabled only when SSL is enabled and SSL Terminate Only is not enabled.

Upstream mTLS Credentials

```
-----BEGIN CERTIFICATE-----
MIIABC123...
```

Please provide all certificates and keys in PEM format.

Notes

- The simultaneous configuration of both Downstream and Upstream mTLS is possible and can be individually managed within the Site Details page.
- To expose the mTLS settings within the Site Details page, a tenant needs to enable the feature **flag site-config-mtls**.

Conclusion

By configuring mutual TLS in both downstream and upstream modes, you establish a secure and authenticated communication channel between clients, the WAF, and the origin server, ensuring robust protection and trust across the implementation.

Troubleshooting Sensor Issues

When you have an issue with sensors, contact the ThreatX SOC at support@threatx.com with a description of your issue.

Depending on the nature of the issue, the ThreatX SOC might request one of the following files.

- HTTP Archive format file (`.har`). HAR files contain sensitive data, including content of the pages you downloaded while recording as including your cookies. The ThreatX SOC can use it to troubleshoot connectivity or other issues with the sensor.
- PCAP (packet capture) file. The file contains captured network packets. The SOC requests a `.pcap` file only if you host your own sensors.

How to generate a HAR file

How you generate a HAR file depends on the web browser you use.

Generating a HAR file in Chrome

1. Open Google Chrome and navigate to the page where the issue is occurring.
2. Look for the Vertical ellipsis button and select **More Tools** › **Developer Tools** › **Network**
3. Look for a Record button in the upper left corner of the tab and make sure it is red. If it is grey, click it once to start recording.
4. Check the **Preserve log** box
5. Click **[Clear]** to clear out any existing logs from the Network tab.
6. Reproduce the issue you are experiencing.
7. Once you have reproduced the issue, select **Right Click** › **Save as HAR with Content**
8. Upload the HAR file as an attachment to your ThreatX support ticket for further.

Generating a HAR file in Mozilla Firefox

1. Open Mozilla Firefox and navigate to the page where the issue is occurring.
2. Select **Mozilla Firefox Menu** › **Web Developer** › **Network**
3. The recording automatically starts when you begin performing actions in the browser.
4. Once you have reproduced the issue and you see that all the actions have been generated in the Developer Network Panel (should just take a few seconds), **Right Click** › **File** › **Save all as Har**
5. Upload the HAR file as an attachment to your ThreatX support ticket for further analysis.

Generating a HAR file in Internet Explorer

1. Open Internet Explorer and go to the page where the issue is occurring.
2. Press `F12` on your keyboard to open developer tools. Then select **[Network]**
3. Reproduce the issue that you were experiencing while the network requests are being recorded.
4. Once done, click **[Save]** and give the file a `.har` extension.
5. Upload the HAR file as an attachment to your ThreatX support ticket for further analysis.

Generating a HAR file in Safari

1. Before generating the HAR file, make sure you can see the **Develop** menu in Safari. If it is not there, follow the instructions in Use the developer tools in the Develop menu in Safari on Mac.
2. Open **Develop** › **Show Web Inspector** › **Network** › **Export** and save the `.har` file.
3. Upload the HAR file as an attachment to your ThreatX support ticket for further analysis.

Edge natively produces HAR files. For more instructions, see the instructions from the Microsoft website. To generate a HAR file in Edge:

1. Open the **Network** tool in F12 developer tools.
2. Reproduce the issue. Upload the HAR file as an attachment to your ThreatX support ticket for further analysis.

The PCAP file is relevant only if you host your own sensors.

To generate a PCAP file that the ThreatX SOC can analyze for troubleshooting connectivity or other issues with the WAF sensor, follow these instructions:

1. Use SSH to connect into the docker host system.
2. Use the following command to display the name of the desired container:

```
$ docker exec -if txwaf \
  && apt get update \
  && apt-get install -y tabby

$ tcpdump -i eth0 \
$ tcpdump -i eth0 \
    -w /tmp/upload_to_threatx.pcap

$ docker cp \
  container_id>:/tmp/upload_to_threatx.pcap \
  upload_to_threatx.pcap
```

Upload the PCAP file to a ThreatX support ticket for further analysis.