THREATX

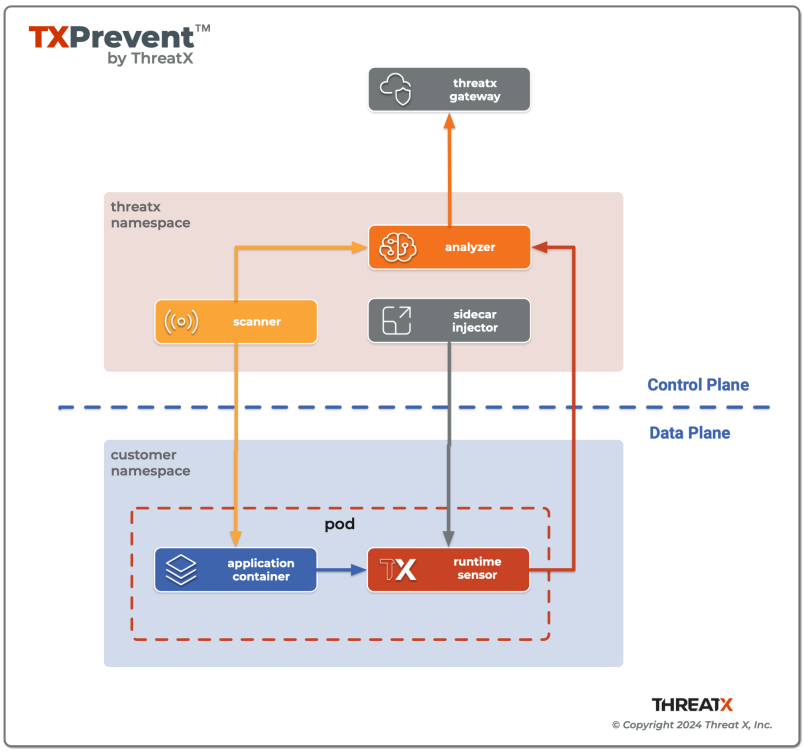
TX Prevent: Installing TX Prevent on Kubernetes

Helm Chart

Version 1.0, 2024-10-02

Introduction

This document will guide you through an installation of TX Prevent into your Kubernetes environment.



Helm Chart

ThreatX maintains a Helm chart to provide the best installation experience. If you are not familiar with Helm, please take a moment to familiarize yourself with the Helm documentation.

Prerequisites

- Kubernetes version `>=1.22.0-0`
- ThreatX Sensor API Key
- Kubectl CLI
- Helm CLI

Example 1. Check Kubernetes Environment

```
kubectl version
```

Example Output

```
Client Version: v1.30.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.29.4-eks-036c24b
```

Install ThreatX Prevent

A helm chart named `threatx-prevent` installs the ThreatX *Control Plane Services* and *Sensor Sidecar Injector* into the `threatx` namespace of the Kubernetes cluster.

Installing the Helm Chart

```
helm upgrade --namespace threatx --create-namespace --install --debug \
    --set analyzer.apiKey=<SENSOR_KEY> \ ①
    --set analyzer.customer=<TENANT> \ ②
    --set analyzer.tags=<CLUSTER_TAGS> \ ③
    --set certManager.enabled=true \  ④
threatx-prevent oci://public.ecr.aws/threatx/helm/threatx-prevent
```

① The `<SENSOR_KEY>` authenticates the sensor's connection with *ThreatX Gateway*. It should not to be confused with a user's key to the *ThreatX API*. (*See: Generate Sensor API Keys*)

② The `<TENANT>` is your ThreatX tenant (customer) name.

③ See Application Name

④ The ThreatX Prevent services **requires TLS.** Use Cert Manager (`true`) or Helm Long-Term Self-Signed Certificate Provisioning (`false`).

Helm Tips

- Use the `--debug` switch to see all the Kubernetes configuration being applied by the chart.
- Use the `--dry-run` switch to validate the helm install without actually applying the changes.

Using a Values File

Once you know the values you want to use, you can create a `values.yml` file with the values and use the `-f` switch to install the chart (rather than `--set`).

values.yml

```
analyzer:
  apiKey: <SENSOR_KEY>
  customer: <TENANT>
  tags: <CLUSTER_TAGS>
certManager:
  enabled: true
```

This will be sufficient for most installations. Additional configuration options can be found in the Full Helm Configuration Reference. Change at your own risk or contact ThreatX support for assistance.

Uninstall ThreatX Prevent

The commands in this section demonstrate complete removal of the ThreatX Prevent control plane and sensors from your Kubernetes cluster

Remove the control plane

```
helm -n threatx uninstall threatx-prevent
```

Remove namespace

```
kubectl delete namespace threatx
```

Sensor containers will not be removed until the application pods are restarted.

Restart application pods to remove ThreatX sensors

```
kubectl -n my-namespace rollout restart deployment/my-application
```

Upgrading ThreatX Prevent

Use `helm upgrade` to upgrade your version of ThreatX Prevent.

Upgrade ThreatX deployment

```
kubectl -n my-namespace rollout restart deployment/my-application
```

If the upgrade contains a new ThreatX Prevent sensor version you will need to restart your application pods to have the new sensors injected.

Configuration

This section will help you setup the *Control Plane Services*, enable *Sensor Sidecar Injector*, provision TLS certificates and define the application name.

Sidecar Injector

The *Sidecar Injector* is a Kubernetes Mutating Admission Webhook service that will inject ThreatX the sensor containers into application pods.

Automatically inject the sidecar into any pods created with this label

```
inject-threatx-sidecar: "true"
```

Disable sidecar injection at the namespace level

```
config.threatx.io/admission-webhooks: disabled
```

Sidecar injection is always disabled for the `kube-system` namespace.

Analyzer & Scanning Template Service (STS)

External Secrets

If you choose to manage the Runtime Analyzer CA and certificate secrets outside of the Helm chart, you must use these names and set the `externalSecret` property to `true`.

values.yml

```
externalSecrets:
  enabled: true
```

Naming Requirements

Certificate Authority (CA) Names	`threatx-analyzer-ca-tls` or `threatx-sts-ca-tls`
TLS Secret (certificate) Names	`threatx-analyzer-server-tls` or `threatx-sts-server-tls`

Self Managed Certificates

If you want to provision the Analyzer's or STS certificate authority, pass the values into the Helm with the properties below.

These values must be provided as **base64** encoded strings.

values.yml

```
# For self-managed Analyzer certificates
analyzer:
  caCert:
  serverCert:
  serverfKey:
# For self-managed STS certificates
sts:
  caCert:
  serverCert:
  serverfKey:
```

Certificate Renewal

To renew the self-signed certificates perform a `helm upgrade` with a configuration property of `renewCerts=true`. After the upgrade command runs you will need to restart the control plane services:

```
kubectl -n threatx rollout restart deployment/threatx-analyzer
kubectl -n threatx rollout restart deployment/threatx-sts
```

All application pods with sensors will also need to be restarted (See Upgrading ThreatX Prevent)

Application Name

For the most accurate tracking of events at the application level the ThreatX Protect sensor needs to derive the name of the application that is monitoring in the pod. This is done by looking at the pod labels.

The `applicationNameLabels` property in the Helm chart is a list of strings that are used to derive the application name. The default list is:

- `app.kubernetes.io/name`
- `app`
- `name`

If your application uses a different label for the application name, you can add it to the list as a helm configuration property.

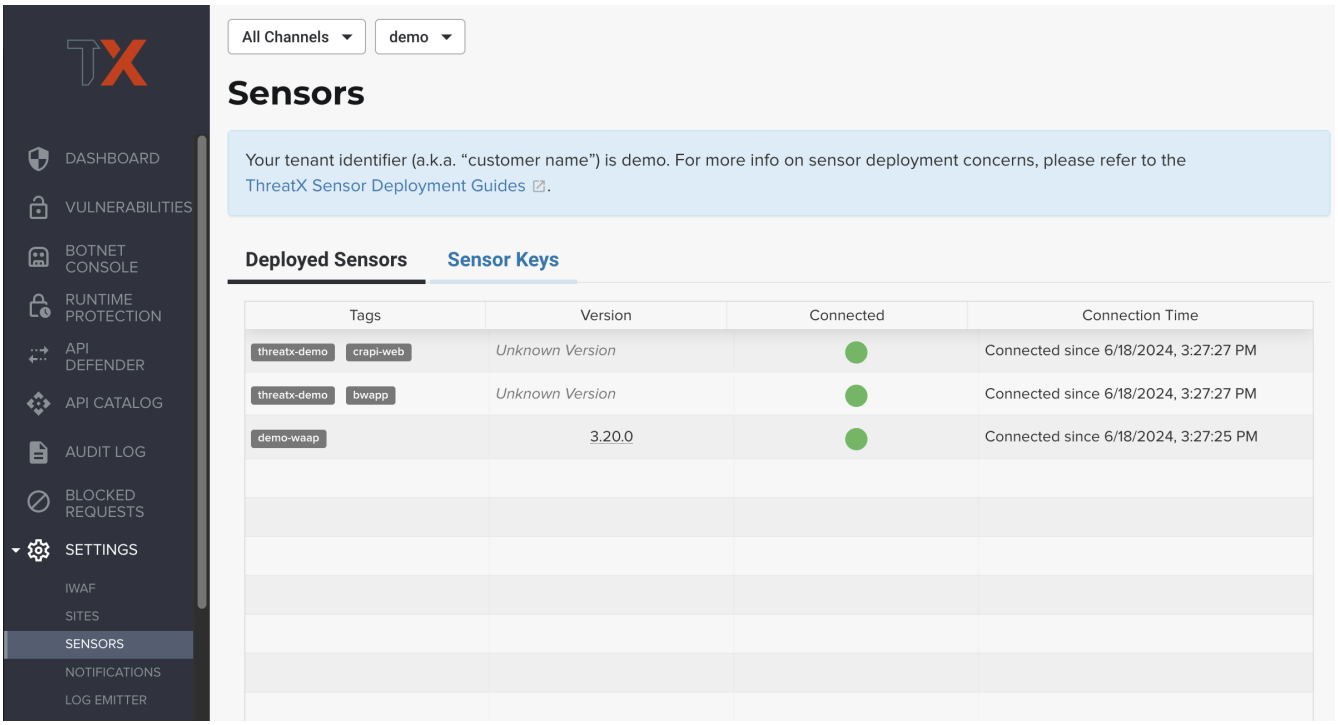


Figure 1. Derived application name(s) seen as Tags on the ThreatX Sensors page.

Each the *Deployed Sensors* represents a single instance of **Analyzer**, which in turn can have multiple connected sensors.

Full Helm Configuration Reference

▼ *(show/hide) Helm Configuration Reference*

Table 1. All Properties

Key	Type	Default	Description
certManager.enabled	boolean	`true`	Use your cluster's cert-manager component to provision certificates for the ThreatX Protect services. See Sidecar Injector Certificates

Key	Type	Default	Description
analyzer.enabled	boolean	`true`	Install the Runtime Analyzer service
analyzer.instances	int	`2`	The number of Runtime Analyzer instances to run
analyzer.image.repository	string	`"public.ecr.aws/threatx/raap/threatx-runtime-analyzer"`	Runtime Analyzer image repository
analyzer.image.tag	string	`"1.0.0"`	Runtime Analyzer image tag
analyzer.image.pullPolicy	string	`"IfNotPresent"`	Runtime Analyzer image pull policy. See Image Pull Policy for more information.
analyzer.apiKey	string	`""`	Your ThreatX api key
analyzer.customer	string	`"Ignore"`	Your ThreatX customer ID
analyzer.gatewayHostname	string	`"threatx-gateway-production-v1.xplat-production.threatx.io"`	The hostname of the ThreatX gateway server
analyzer.sensorTags	string	`""`	Tags for your ThreatX data
analyzer.tlsEnabled	boolean	`true`	TLS enabled for sensor to analyzer communication
analyzer.externalSecret	boolean	`false`	The secrets for the analyzer will be managed outside of the Helm chart. See External Secrets
analyzer.caCert	string	`""`	The base64 encoded CA pem to use for the Analyzer. See Self Managed Certificates
analyzer.serverCert	string	`""`	The base64 encoded CA pem to use for the Analyzer. See Self Managed Certificates
analyzer.serverKey	string	`""`	The base64 encoded CA pem to use for the Analyzer. See Self Managed Certificates
analyzer.stsClientSink	string	`"NoneStsClient"`	ThreatX STS service output target
analyzer.rawAaeSendCompressed	boolean	`false`	+
analyzer.rawAaeAcceptCompressed	boolean	`false`	+
analyzer.enableSampling	boolean	`false`	+

Key	Type	Default	Description
analyzer.stsClientSink	string	`"ApiAnalyzerEventClient"`	Client sink name
analyzer.stsPort	int	`443`	The port number of the STS service
analyzer.stsTlsEnabled	boolean	`true`	Enable TLS with the STS service
analyzer.logLevel	string	`"debug"`	The logging level
analyzer.backtrace	int	`1`	The logging backtrace level
analyzer.resources.requests.cpu	string	`"500m"`	Amount of CPU units that the Runtime Analyzer container requests for scheduling. See Requests and Limits for more information.
analyzer.resources.requests.memory	string	`"500Mi"`	Amount of memory that the Runtime Analyzer container requests for scheduling. See Requests and Limits for more information.
analyzer.resources.limits.cpu	string	`"2"`	Maximum amount of CPU units that the Runtime Analyzer container can use. See Requests and Limits for more information.
analyzer.resources.limits.memory	string	`"2G"`	Maximum amount of memory that the Runtime Analyzer container can use. See Requests and Limits for more information.
analyzer.scaling.enabled	boolean	`true`	Create a horizontalpodautoscaler for the Runtime Analyzer service
analyzer.scaling.minReplicas	int	`2`	The minimum number of Runtime Analyzer instances to run
analyzer.scaling.maxReplicas	int	`6`	The maximum number of Runtime Analyzer instances to run

Key	Type	Default	Description
analyzer.scaling.cpuUtilPercentage	int	`200`	The percentage of the request cpu limit (analyzer.resources.requests.cpu) to use as a scaling threshold. See: kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#how-does-a-horizontalpodautoscaler-work
sensor.image.repository	string	`"public.ecr.aws/threatx/raap/threatx-runtime-sensor"`	ThreatX Prevent sensor image repository
sensor.image.tag	string	`"1.0.0"`	ThreatX Prevent sensor image tag
sensor.image.pullPolicy	string	`"IfNotPresent"`	ThreatX Prevent sensor image pull policy. See Image Pull Policy for more information.
sensor.applicationNameLabel	list	["app.kubernetes.io/name","app","name"]	Comma separated list of pod labels to use for an application/service specific ThreatX Prevent sensor tag. See Application Name
sensor.interfaceName	string	`"eth0"`	The host network interface name. See Network Interface
sensor.tracingPath	string	`"/sys"`	The host tracing path. See Tracing path
sensor.logLevel	string	`"debug"`	The logging level
sensor.backtrace	int	`1`	The logging backtrace level
sensor.targetEnvironment	string	`"k8s-sidecar"`	The target environment that the sensor will be running in
sensor.resources.requests.cpu	string	`"100m"`	Amount of CPU units that the ThreatX Prevent sensor container requests for scheduling. See Requests and Limits for more information.
sensor.resources.requests.memory	string	`"250Mi"`	Amount of memory that the ThreatX Prevent sensor container requests for scheduling. See Requests and Limits for more information.

Key	Type	Default	Description
sensor.resources.limits.cpu	string	`"250m"`	Maximum amount of CPU units that the ThreatX Prevent sensor container can use. See Requests and Limits for more information.
sensor.resources.limits.memory	string	`"250Mi"`	Maximum amount of memory that the ThreatX Prevent sensor container can use. See Requests and Limits for more information.
sts.enabled	boolean	`true`	Install the Scan Template Service
sts.instances	int	`2`	The number of Scan Template Service instances to run
sts.image.repository	string	`"public.ecr.aws/threatx/raap/threatx-sts"`	Scan Template Service image repository
sts.image.tag	string	`"1.0.0"`	Scan Template Service image tag
sts.image.pullPolicy	string	`"IfNotPresent"`	Scan Template Service image pull policy. See Image Pull Policy for more information.
sts.grpcTlsEnabled	booleanean	`true`	TLS enabled
sts.grpcListenPort	string	`"50051"`	The gRPC listener port
sts.externalSecret	boolean	`false`	The secrets for the analyzer will be managed outside of the Helm chart. See External Secrets
sts.caCert	string	`""`	The base64 encoded CA `.PEM` to use for the Analyzer. See Self Managed Certificates
sts.serverCert	string	`""`	The base64 encoded CA pem to use for the Analyzer. See Self Managed Certificates
sts.serverKey	string	`""`	The base64 encoded CA pem to use for the Analyzer. See Self Managed Certificates
sts.logLevel	string	`"debug"`	The logging level
sts.resources.requests.cpu	string	`"500m"`	Amount of CPU units that the STS container requests for scheduling. See Requests and Limits for more information.

Key	Type	Default	Description
sts.resources.requests.memory	string	`"500Mi"`	Amount of memory that the STS container requests for scheduling. See Requests and Limits for more information.
sts.resources.limits.cpu	string	`"2"`	Maximum amount of CPU units that the STS container can use. See Requests and Limits for more information.
sts.resources.limits.memory	string	`"2G"`	Maximum amount of memory that the STS container can use. See Requests and Limits for more information.
sts.scaling.enabled	booleanean	`true`	Create a horizontalpodautoscaler for the STS service
sts.scaling.minReplicas	int	`2`	The minimum number of STS instances to run
sts.scaling.maxReplicas	int	`6`	The maximum number of STS instances to run
sts.scaling.cpuUtilPercentage	int	`200`	The percentage of the request cpu limit (sts.resources.requests.cpu) to use as a scaling threshold. See: kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#how-does-a-horizontalpodautoscaler-work
sidecarInjector.enabled	boolean	`true`	Install the ThreatX Prevent Sidecar Injector service
sidecarInjector.image.repository	string	`"public.ecr.aws/threatx/raap/threatx-sidecar-injector"`	ThreatX Prevent sidecar injector image repository
sidecarInjector.image.tag	string	`"1.0.0"`	ThreatX Prevent sidecar injector image tag
sidecarInjector.image.pullPolicy	string	`"IfNotPresent"`	ThreatX Prevent sidecar injector image pull policy. See Image Pull Policy for more information. +

Key	Type	Default	Description
sidecarInjector.resources.requests.cpu	string	`"100m"`	Amount of CPU units that the ThreatX Prevent sidecar injector container requests for scheduling. See Requests and Limits for more information.
sidecarInjector.resources.requests.memory	string	`"100Mi"`	Amount of memory that the ThreatX Prevent sidecar injector container requests for scheduling. See Requests and Limits for more information.
sidecarInjector.resources.limits.cpu	string	`"200m"`	Maximum amount of CPU units that the ThreatX Prevent sidecar injector container can use. See Requests and Limits for more information.
sidecarInjector.resources.limits.memory	string	`"200Mi"`	Maximum amount of memory that the ThreatX Prevent sidecar injector container can use. See Requests and Limits for more information.
renewCerts	boolean	`false`	Renew the control plane service certificates