THREATX

TX Prevent
Deployment

Version 1.3, 2025-01-30

Table of Contents

AWS EC2 Deployment Guide (cloudFormation)

This document will guide you through an installation of TX Prevent into your AWS environment by using CloudFormation.

Architecture

The TX Prevent deployment architecture leverages several Amazon Web Services (AWS) components to provide a highly available and secure product.

Runtime sensors will deploy onto EC2 instances alongside the applications or services you want to watch. These sensors communicate with the ThreatX Prevent control plane services.

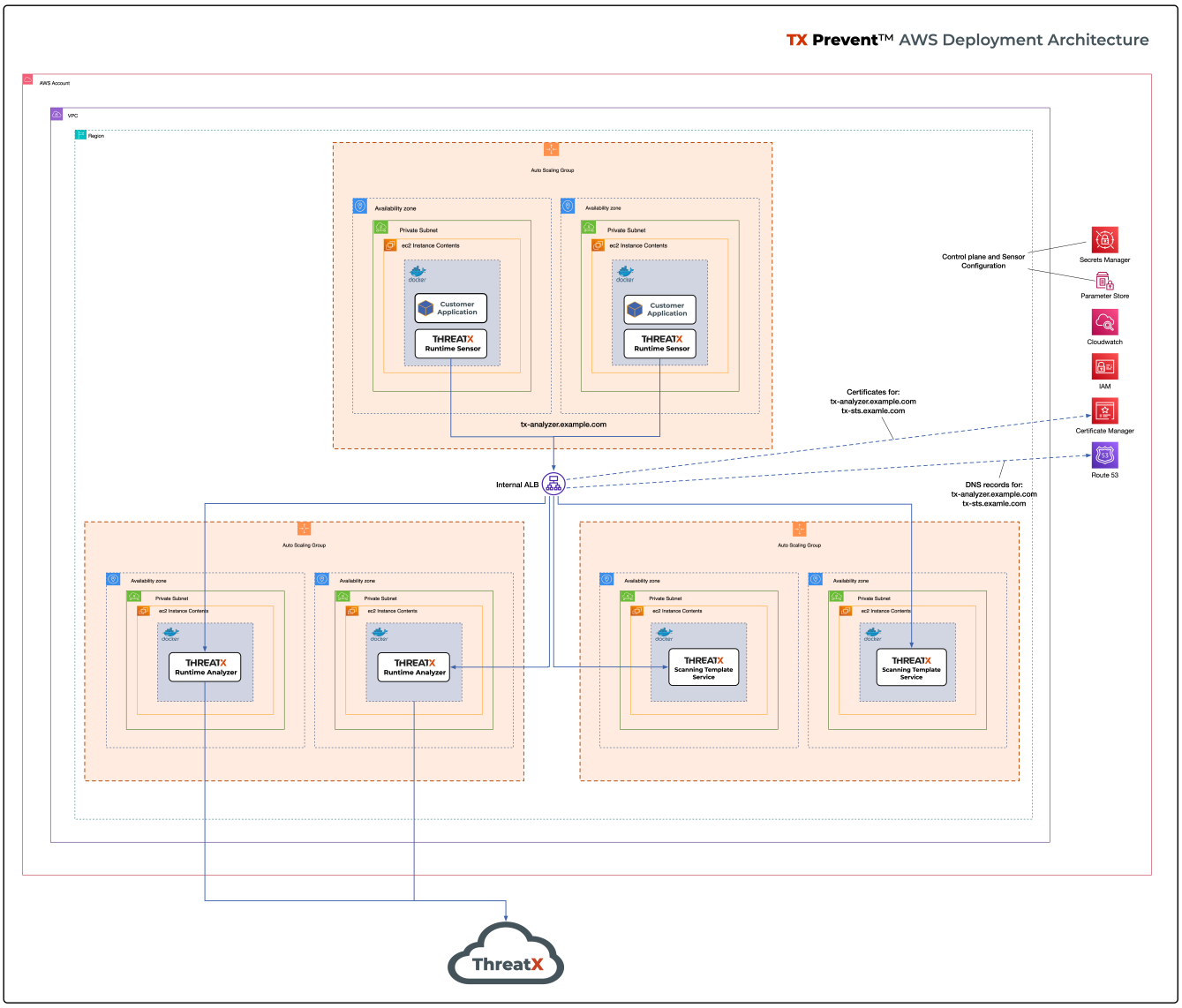


Figure 1. TX Prevent Context Diagram

High Availability

- For each control plane service, instances are created in multiple availability zones
- The instances are deployed in Auto Scaling Groups (ASG) where they are continuously monitored to ensure the desired number of healthy instances

Security

- All control plane services are deployed into private subnets and are never publicly exposed
- All traffic to Control plane services is encrypted using TLS with Amazon provisioned certificates

ThreatX Control Plane Services

Runtime Analyzer	A data aggregator, analysis engine, and event router. Connects to ThreatX and emits vulnerability metadata.
Scan Template Service	Ingests passively detected vulnerability data and generates highly targeted scan templates. Executes individual scans are executed and returns the results after determining efficacy.
OTEL Collector	This service collects logs and metrics from the sensors and other control plane services and send them back to ThreatX for enhanced product support.

AWS Components and Services

Application Load Balancer (ALB)	Fronts the TX Prevent control plane services. Each control plane service has multiple instances in at least *two* availability zones for high availability with the ALB distributing traffic between them.
Auto Scaling Group (ASG)	Maintains the desired number of healthy service EC2 instances. If an instance becomes unhealthy or is unexpectedly terminated the ASG will create another instance.
Parameter Store	Configuration properties for sensors and control plane services.
Secrets Manager	Sensitive configuration properties.
Route53	DNS records for the control plane services.
Certificate Manager (ACM)	Provisioning certificates for the control plane services.

Prerequisites

Recommended Reading

Performing this deployment process requires familiarity with the following AWS services:

- Amazon VPC
- AWS CloudFormation
- AWS Route53

Additionally, for deployments requiring VPC connectivity between the TX Prevent VPC and another VPC containing monitored application/service:

- AWS VPC Peering
- AWS Transit Gateway

Preflight Checklist

The following items must be completed before the deployment can begin.

- ☐ **Valid ThreatX Tenant ID** (customer name)
- ☐ **Valid ThreatX API Key** (See ThreatX Sensor API Key)
- ☐ **AWS user or role** with either the **AdministratorAccess** policy or our custom deployment IAM policy
- ☐ **EC2 key pair** for SSH access to the EC2 instances. (See EC2 Key Pair)
- ☐ **Docker** installed on the EC2 instances where the sensors will be deployed
- ☐ **AWS Route53 Hosted Zone** for DNS records and certificates of control plane services
- ☐ *VPC* with at least:
  - **2** private subnets
  - **1** public subnet
  - **1** internet gateway
  - **1** NAT gateway

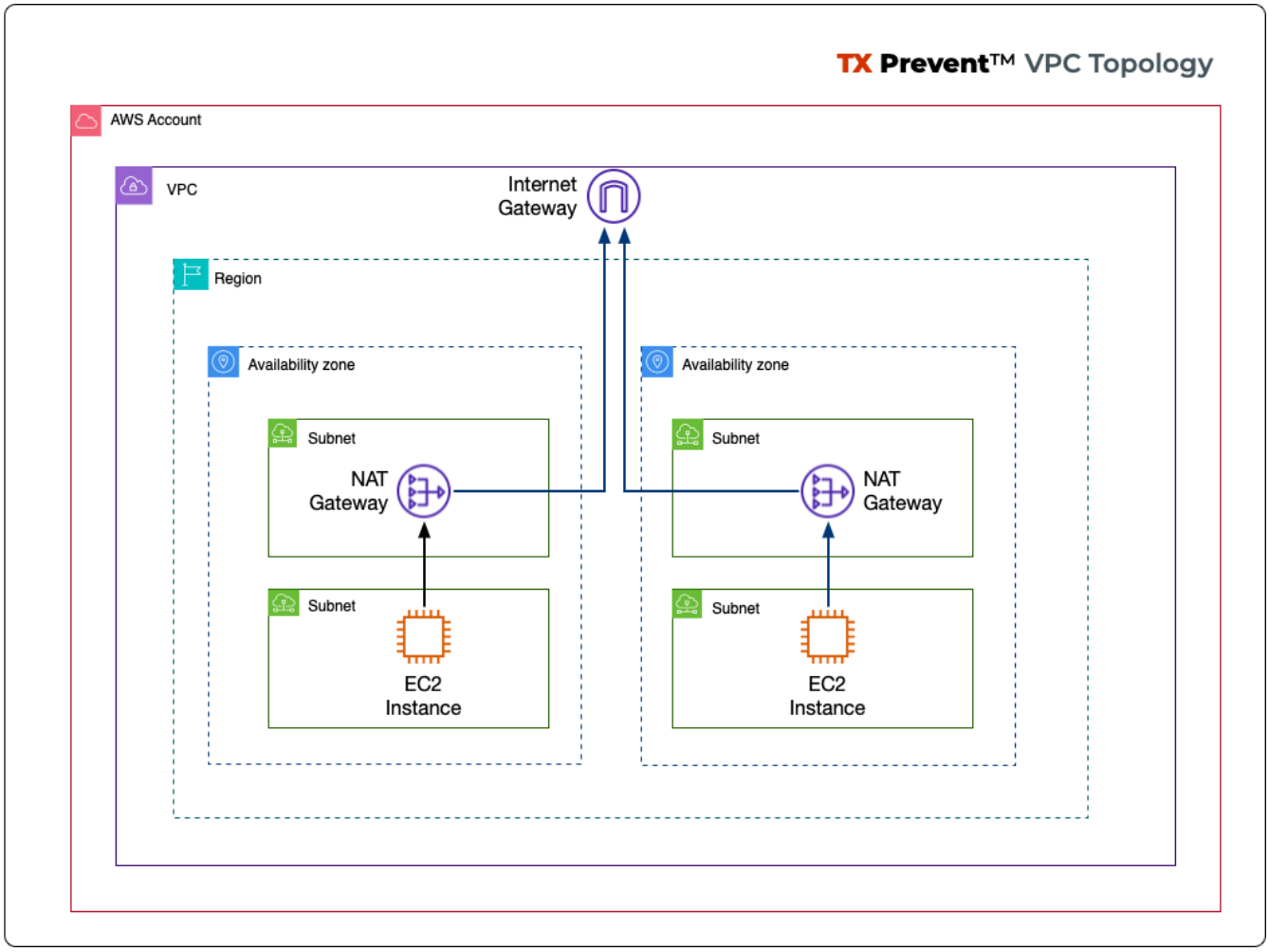


Figure 2. TX Prevent Standard VPC Topology

Creating an EC2 Key Pair

The EC2 Key pair will be used to SSH into the ThreatX Control Plane EC2 instances. To create one for the install follow the steps below:*

1. Open the **AWS EC2 Console**.
2. Select **Main Menu (left) › Network & Security › Key Pairs**
3. On the **Key pairs** page, click **[Create key pair]**
4. On the **Create Key Pair** page:
   a. Enter a name (e.g., *<threatx-prevent>*)
   b. Select **RSA**
   c. Select **.pem** format
   d. Add any **Tags** that you want
   e. Click on **[Create key pair]**.

The private key will then be downloaded to your system.

Put this key in a safe place. It can be used to SSH into any of TX Prevent EC2's.

Runtime Sensor System Requirements

Resources	It is recommended to have **at least 2 cores** and **300MB of memory** available on the EC2 instance that they will be running on.
Network Connectivity	If Sensors are deployed into a *different VPC than that of the control plane*, VPC peering or Transit Gateway connectivity will need to be setup between the VPCs.
Scanning Requirements	You may need to adjust security groups to allow ingress traffic from the Scan Template Service to the target endpoints.

Control Plane Deployment

Cloudformation template URL

```
$ https://threatx-prevent-cf-template.s3.amazonaws.com/threatx-
prevent.yaml[*Download the TX Prevent CloudFormation template* -
__https://threatx-prevent-cf-template.s3.amazonaws.com/threatx-
prevent.yaml__]
```

CloudFormation Template Parameters

Table 1. TX Prevent Configuration Parameters

Key	Type	Default	Description
ApiKey	String		The API key for TX Prevent
TenantId	String		The Tenant ID for TX Prevent
LogLevel	String	info	The logging level to use for all services
VPC	AWS::EC2::VPC::Id		A virtual private cloud (VPC) to install into. See VPC Setup
Subnets	List<AWS::EC2::Subnet::Id>		At least two private subnets in different Availability Zones in the selected VPC
HostedZoneId	String		The ID of the Hosted Zone in Route53 to add DNS record to. Must align with the specified Hosted Zone Name.
KeyName	AWS::EC2::KeyPair::KeyName		Name of an existing EC2 key pair to allow SSH access to the control plane's EC2 instances
GatewayHostname	String	threatx-grpc2kafka-production-v1.xplat-production.threatx.io	The Gateway hostname for TX Prevent
HostedZoneName	String		The Hosed Zone Name in Route53 for the control plane service DNS records. Must align with the specified Hosted Zone Id.
LatestAmiId	String	/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64	The latest AMI ID for the TX Prevent services

Table 2. Analyzer Configuration Parameters

Key	Type	Default	Description
AnalyzerTags	String		The tag values for the Runtime Analyzer
EnvironmentName	String		The environment name for the Runtime Analyzer

Key	Type	Default	Description
AaeCachingEnabled	boolean	false	Enable caching for the Runtime Analyzer
SendCompressed	boolean	false	Enable compression for the Runtime Analyzer
AnalyzerImageTag	String	1.2.1	The tag for the Runtime Analyzer docker image
AcceptCompressed	boolean	false	Accept compressed data for the Runtime Analyzer
AnalyzerDesiredInstances	Number	2	Number of desired Runtime Analyzer instances
QueueSampleEnabled	boolean	false	Enable queue sampling for the Runtime Analyzer
EnableStdOutMetrics	boolean	false	Enable stdout metrics for the Runtime Analyzer
EnableCatalogMonitor	boolean	false	Enable catalog monitoring for the Runtime Analyzer
AnalyzerInstanceType	String	t3.small	The EC2 instance type for the Runtime Analyzer instances

Table 3. STS Configuration Parameters

Key	Type	Default	Description
StsImageTag	String	1.2.0	The tag for the Scan Template Service docker image
StsDesiredInstances	Number	2	Number of desired Scan Template Service instances
StsInstanceType	String	t3.small	The EC2 instance type for the Scan Template Service instances

Table 4. OTEL Configuration Parameters

Key	Type	Default	Description
OtelImageTag	String	1.1.0	The tag for the OTEL Collector docker image
OtelInstanceType	String	t3.small	The EC2 instance type for the OTEL Collector instances
OtelGatewayUrl	String	otlp-grpc-production.xplat-aml-prod.threatx.io	The Gateway URL for the OTEL Collector

Step-by-Step Console Deployment Instructions

Follow these steps to deploy the CloudFormation stack by using the AWS Console to create the TX Prevent services in your AWS environment.

1 Add the TX Prevent CloudFormation Template

1. Sign in to your AWS account via the AWS Console. Select the desired region for the deployment.
2. Open the CloudFormation console
3. Select **[Create stack]** and **[With new resources (standard)]**
4. Select **[Choose an existing template]** and then add the URL for the TX Prevent template to the **Amazon S3 URL** field: `threatx-prevent-cf-template.s3.amazonaws.com/threatx-prevent.yaml`

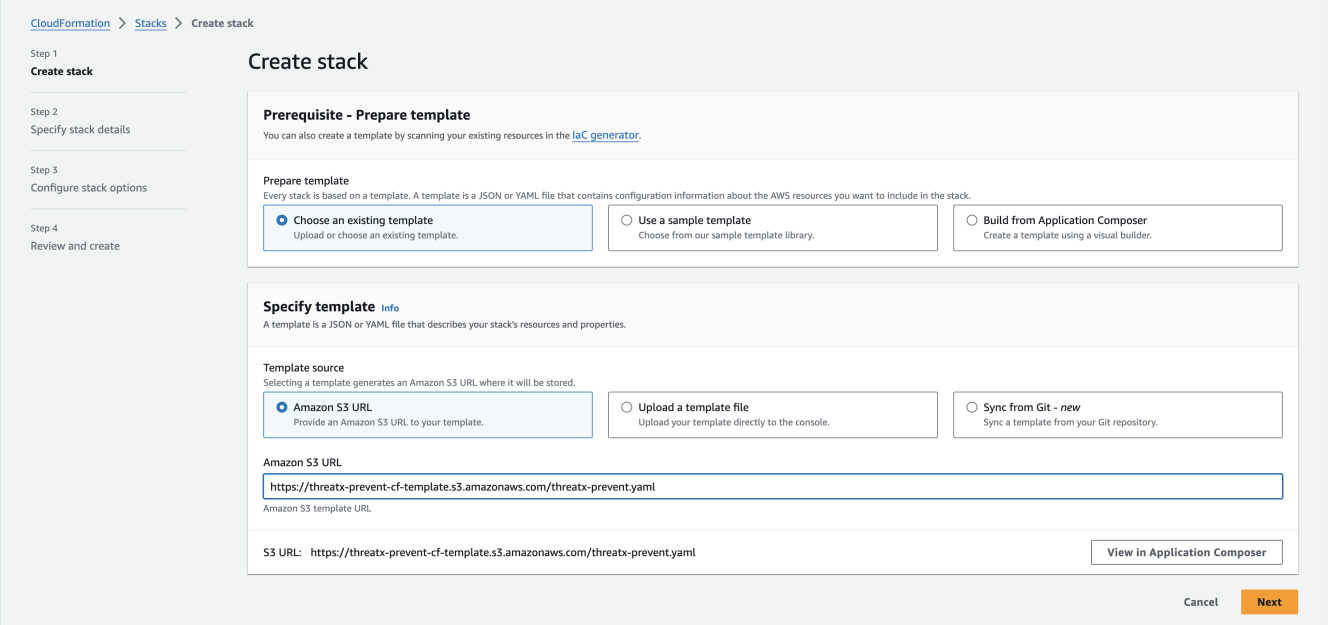


Figure 3. TX Prevent Standard VPC Setup

2 Configure the Stack Details

*On the **Specify stack details** Page*

1. In the **Stack Name** field, enter: *ThreatXPrevent*
   a. NOTE: If you choose to enter a different stack name then it must be 14 characters or less in length. This stack name is used as part of AWS resource tags and some of those have length limitations.
2. Provide values for the following parameters. Descriptions of all parameters can be found here: Template Parameters.
   a. `TenantId`
   b. `ApiKey`
   c. `VPC`
   d. `Subnets`
   e. `HostedZoneId`
   f. `HostedZoneName`
   g. `KeyName`
   h. `AnalyzerTags`

3. For all other parameters leave the default settings and adjust them only if instructed by ThreatX.
4. Select **[Next]**

3 Configure the Stack Options

*On the **Configure Stack Options** Page*

1. *(optional)* Specify tags for the resources in your stack and set any advanced options you want.
2. In the Capabilities section select `I acknowledge that AWS CloudFormation might create IAM resources with custom names`
3. Select **[Next]**

4 Review and Create the Stack

*On the **Review** page ...*

1. Review and confirm all of the template settings.
2. Under **Capabilities**, review and select the check boxes to acknowledge.
3. Select **[Create Stack]**

The TX Prevent deployment is ready when the stack status is **CREATE_COMPLETE**. Stack creation should take **5 to 10 minutes**.

You can watch creation events under the **Event** tab. To view all the created resources, choose the **Outputs** tab.

AWS CLI Deployment Instructions

Below is an example AWS CLI command to perform the installation. Replace any values shown in <> with your specific values

Installing with the CLI

```
$ aws cloudformation create-stack \
--template-url https://threatx-prevent-cf-
template.s3.amazonaws.com/threatx-prevent.yaml \
--stack-name ThreatXPrevent \
--region <your-region> \
--capabilities CAPABILITY_IAM \
--capabilities CAPABILITY_NAMED_IAM \
--parameters \
ParameterKey=TenantId,ParameterValue=<your-tenant-id> \
ParameterKey=ApiKey,ParameterValue=<your-api-key> \
ParameterKey=VPC,ParameterValue=<your-vpc-id> \
ParameterKey=Subnets,ParameterValue=<your-subnet-id-1>,<your-subnet-id-
2> \
ParameterKey=HostedZoneId,ParameterValue=<your-hosted-zone-id> \
```

```
ParameterKey=HostedZoneName,ParameterValue=<your-hosted-zone-name> \
ParameterKey=KeyName,ParameterValue=<your-ec2-key-pair-name> \
ParameterKey=AnalyzerTags,ParameterValue=<your-tag-value>
```

Runtime Sensor Deployment

The `run-tx-sensor.sh` script can be used to configure and launch ThreatX Runtime Sensor containers on an EC2 instance where an application to be monitored is running.

Download run-tx-sensor.sh

There are two types of sensor monitoring modes that can be setup:

- Container Monitoring Mode : The sensor monitors traffic on a specific application container
- Host Monitoring Mode : The sensor monitors traffic on the entire host

Features of the script

- In Container Monitoring Mode:
  - The sensor will be bound to a specified Docker application container's network stack.
  - The sensor container will be named based upon the specified application container it will be monitoring: `threatx-sensor-<app-container-name>`
  - Try to set the sensor to the correct network interface for the specified container. If the application container has multiple network interfaces then it will output those interface names to allow you to select the appropriate one.
- In Host networking Mode:
  - The sensor will be bound to the EC2 instance's host network.
  - The sensor container will be named based upon the specified application instance it will be monitoring: `threatx-sensor-<app-instance-name>`
  - Set the sensor to the instance's default route network interface.
- If a currently running sensor is detected it will gracefully shut it down before starting a new one.
- Check and enforce OS and other system requirements.
- Will output useful information that can be passed on to the ThreatX support team

Usage

Container Monitoring Mode

In this mode the sensor will monitor traffic on a specific application container. The application docker container name is specified via the app-name argument.

```
run-tx-sensor.sh \
  --tenant-id <tx-customer-id> \①
  --domain <dns-domain-of-tx-prevent-deployment> \②
  --app-name <app-container-name>③
```

① Your ThreatX customer ID

② The DNS domain name used for the installation of your ThreatX Prevent control plane

③ The docker container name of the application you want to monitor

Host Monitoring Mode

In this mode the sensor will monitor traffic on the EC2 instance's network. The `--host-mode` or `-H` command line switch is used to enable this mode and a name for the ec2 instance is specified via the app-name argument.

```
run-tx-sensor.sh \
  --tenant-id <tx-tenant-id> \①
  --domain <dns-domain-of-tx-prevent-deployment> \②
  --app-name <app-or-instance-name> \③
  --host-mode
```

① Your ThreatX customer ID

② The DNS domain name used for the installation of your ThreatX Prevent control plane

③ The docker container name of the instance you want to monitor

Command Line Arguments

```
Usage: run-tx-sensor.sh [OPTIONS]
Options:
  -n, --tenant-id string           The ThreatX tenant ID (required)
  -d, --domain string              The domain name used for the ThreatX
Prevent installation (required)
  -a, --app-name string            The application name (container name)
or an instance identifier (required)
  -s, --stack-name string          The stack name used for the ThreatX
Prevent installation if the default name of ThreatXPrevent was not used
(default: ThreatXPrevent)
  -i, --image-repo string          Docker image repository (default:
public.ecr.aws/threatx/raap/threatx-runtime-sensor)
  -t, --image-tag string           Docker image tag (default: 1.2.0)
  -N, --network-interface string   The network interface to use
  -l, --sensor-log-level string    Sensor log level : trace, debug,
info, warn, error (default: info)
  -p, --trace-path string          Trace path (default: /sys)
  -H, --host-mode                  Enable listening on the host network
(default: disabled)
  -I, --internal                   Enable for internal image access
(default: disabled)
  -h, --help                       Display this help message
```

CloudFormation IAM Permissions

There are two options for obtaining the permissions needed to create the TX Prevent stack:

1. Using an existing user or role with the **AdministratorAccess** policy
2. Creating a new custom IAM policy with the minimum required permissions according to least privilege which will be assigned to the existing user or role you want to use for installation (continue reading next section)

Configure AWS with the Minimum Permissions Required for Stack Creation

Now we will create a custom policy with the minimum permissions required to create the TX Prevent stack.

Create a Custom Policy

1. On the **AWS Services** page, Select **[IAM]**.
2. From **IAM Dashboard**, select **Main Menu (left)** › **Policies**
3. On the **Policies** page, Select **[Create policy]**
4. On the **Specify Permissions** page, under the **JSON** tab:
5. Copy the JSON below into the Policy editor.
6. Replace all placeholder instances with your actual values:
   a. `<account-id>` with your *AWS Account ID*
   b. `<hosted-zone-id>` with your *AWS Route53 Hosted Zone ID*

tx-prevent-cf-iam-policy.json

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LaunchTemplates",
            "Action": "ec2:CreateLaunchTemplate",
            "Effect": "Allow",
            "Resource": "arn:aws:ec2:*:<account-id>:launch-template/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeParameters"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ssm:*"
            ],
            "Resource": "arn:aws:ssm:*:<account-
id>:parameter/ThreatXPrevent*"
        },
        {
            "Sid": "EC2",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupEgress"
            ],
            "Resource": "arn:aws:ec2:*:<account-id>:security-group/*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/aws:cloudformation:stack-name":
"ThreatXPrevent*"
                }
            }
        },
        {
            "Sid": "EC2v3",
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances",
                "ec2:DeleteSecurityGroup",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RunInstances",
                "ec2:DescribeInstances",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeKeyPairs",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:DescribeSecurityGroups",
                "ec2:CreateLaunchTemplate",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:DeleteLaunchTemplateVersions",
                "ec2:ModifyLaunchTemplate",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
```

```
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeListenerCertificates",
                "elasticloadbalancing:DescribeSSLPolicies",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTags",
                "elasticloadbalancing:DescribeTrustStores"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ElasticLoadbalancing",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:RemoveTags",
                "elasticloadbalancing:DescribeTags",
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeListeners"
            ],
            "Resource": "*"
        },
        {
            "Sid": "TargetGroup",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:ModifyTargetGroup",
                "elasticloadbalancing:ModifyTargetGroupAttributes"
            ],
            "Resource": "arn:aws:elasticloadbalancing:*:<account-
id>:targetgroup/ThreatXPrevent*"
        },
        {
            "Sid": "ElasticLoadbalancingV2",
            "Effect": "Allow",
            "Action": [
```

```
"elasticloadbalancing:SetLoadBalancerPoliciesOfListener",

"elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:ModifyLoadBalancerAttributes",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:SetWebAcl",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyRule",
                "elasticloadbalancing:CreateListener"
            ],
            "Resource": "arn:aws:elasticloadbalancing:*:<account-
id>:loadbalancer/app/ThreatXPrevent*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:DeleteRule",
                "elasticloadbalancing:DeleteListener"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:*:<account-
id>:listener/app/ThreatXPrevent*",
                "arn:aws:elasticloadbalancing:*:<account-id>:listener-
rule/app/ThreatXPrevent*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets"
            ],
            "Resource":
"arn:aws:elasticloadbalancing:*:*:targetgroup/*/*"
        },
        {
            "Sid": "IAM",
            "Effect": "Allow",
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
```

```
                "iam:GetInstanceProfile",
                "iam:GetRole",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:AddRoleToInstanceProfile",
                "iam:PassRole",
                "iam:DeleteRolePolicy",
                "iam:GetRolePolicy",
                "iam:GetPolicy",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:ListPolicyVersions",
                "iam:TagRole",
                "iam:DetachRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::<account-id>:role/ThreatXPrevent*",
                "arn:aws:iam::<account-id>:policy/ThreatXPrevent*",
                "arn:aws:iam::<account-id>:instance-
profile/ThreatXPrevent*"
            ]
        },
        {
            "Sid": "IAMv2",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": [
                "arn:aws:iam::<account-id>:role/ThreatXPrevent*",
                "arn:aws:iam::<account-id>:policy/ThreatXPrevent*"
            ]
        },
        {
            "Sid": "ACM",
            "Effect": "Allow",
            "Action": "acm:*",
            "Resource": "arn:aws:acm:*:<account-id>:certificate/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
```

```
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds",
                "secretsmanager:CreateSecret",
                "secretsmanager:PutSecretValue",
                "secretsmanager:TagResource",
                "secretsmanager:DeleteSecret"
            ],
            "Resource": "arn:aws:secretsmanager:*:<account-
id>:secret:/ThreatXPrevent*"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:ListSecrets",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "route53:ChangeResourceRecordSets",
                "route53:GetHostedZone"
            ],
            "Resource": "arn:aws:route53:::hostedzone/<hosted-zone-id>_"
        },
        {
            "Effect": "Allow",
            "Action": "route53:GetChange",
            "Resource": "arn:aws:route53:::change/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:PutScalingPolicy",
                "autoscaling:DescribePolicies",
                "autoscaling:DeletePolicy",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DescribeScalingActivities"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

1. When you are complete, click **[Next]**
2. Give the policy a name (e.g., *threatx-prevent-install*)
3. Add a 🏷️ **Tag**:
   - **Key**: *product*
   - **Value**: *threatx-prevent*
4. Click **[Create Policy]**.

🎭 Creating A New Role For The Installation

From the **IAM Console**…

1. In the **main menu** to the left, select **Access Management** › **Roles**
2. Click the **[Create Role]** button.

From the **Create Role** page…

1. Verify that the AWS service button is selected.
2. From the list, select *CloudFormation* and click **[Next]**.
3. In the **Filter Policies** field, locate and select the checkbox of the policy you created. Click **[Next]**.
4. For **Role Name**, enter *threatx-prevent-install*.
5. Add a 🏷️ **Tag**:
   - **Key**: *product*
   - **Value**: *threatx-prevent*
6. Click **[Create Role]**

🚀 Use The New Role To Create The Stack

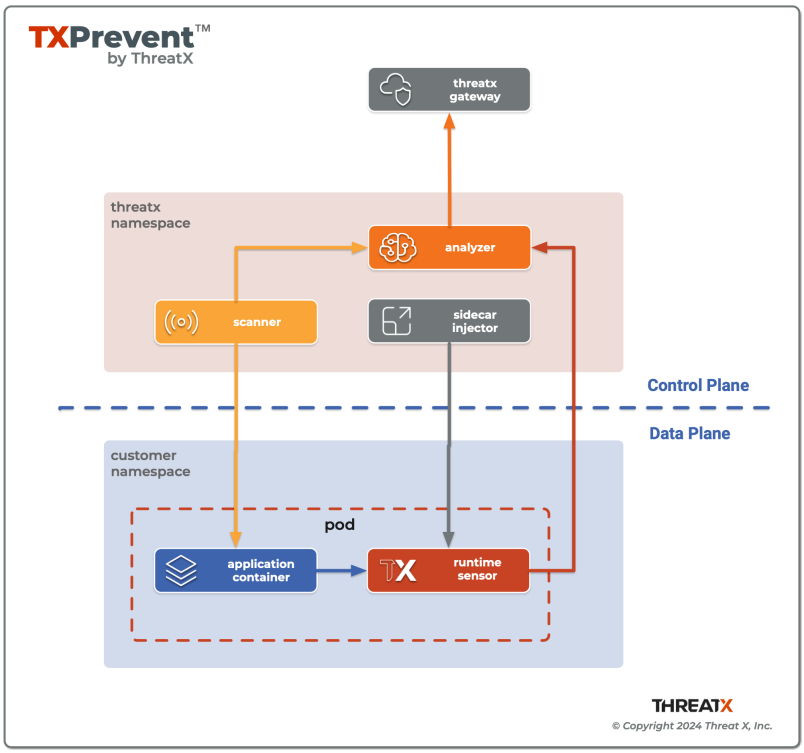
From the **Configure Stack Options Page** …

1. Locate the **Permissions** section
2. In the **IAM Role Name** field, select the newly created role: *threatx-prevent-install*

Kubernetes Installation Guide (Helm)

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, take a moment to familiarize yourself with the Helm documentation.

Prerequisites

- Kubernetes version `>=1.27.0-0`
- Sensor API Keys
- 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 TX 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 apiKey=<SENSOR_KEY> \ ①
    --set 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 product's component connections with ThreatX Gateway. It should not to be confused with a user's key to the ThreatX API. See Generating & revoking sensor API keys.

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

③ Tag values for the analyzer instances. See Analyzer Tags

④ The TX 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

```
apiKey: <SENSOR_KEY>
customer: <TENANT>
analyzer:
  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 TX Prevent

The commands in this section demonstrate complete removal of the TX 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 TX Prevent

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

Sensor Upgrades

Since sensors run as containers in your application pods you will need to restart those pods to get a new sensor version injected into them.

Manually restart a application deployment

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

The ThreatX Prevent Helm chart can also be configured to do rolling restarts of your application deployments during the upgrade. A set of namespaces and deployments within those namespace can be specified in the `podRestart` properties and the Helm upgrade command will perform rolling restarts of those using Helm post-upgrade hooks.

values.yml

```
podRestart:
  enabled: true
  items:
    <app-namespace>:
      deployments:
        - <app-deployment-name>
    <another-app-namespace>:
      deployments:
        - <app-deployment-name>
        - <another-app-deployment-name>
```

Configuration

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

Certificates

Communication between the ThreatX Prevent components use TLS for security. This requires the use of certificates. We provide several different options around certificate management.

Using the cert-manager Component

If cert-manager is installed in your cluster you can enable the install to use it for certificate provisioning and management.

```
certManager.enabled: true
```

Self-Signed Certificates

You can choose to have the Helm chart create self-signed certificates on installation.

```
certManager.enabled: false
```

Certificate Renewal

The self-signed certificates created on install are good for 2 years. 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 TX Prevent)

External Secrets

You can also choose to manage the product certificate secrets outside of the Helm chart, you must use these Kubernetes secret names and set the `externalSecret` property to `true`.

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`

values.yml

```
externalSecrets:
  enabled: true
```

Self Managed Certificates

If you want to self provision the product certificates and then pass them into the installation you can use the following properties.

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:
```

Analyzer & Scanning Template Service (STS)

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 it 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 pod label names that are used to derive the application name. The default list is:

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

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

Analyzer Tags

The tagging of analyzer instances is done with the `analyzer.tags` property. The value of this should be a comma-separated list of strings that can identify the set of analyzers in a particular deployment.

Single tag

```
analyzer.tags=production
```

Multiple tags

```
analyzer.tags=production,pci,east
```

These tags are visible in the CtrlX Sensor Dashboard

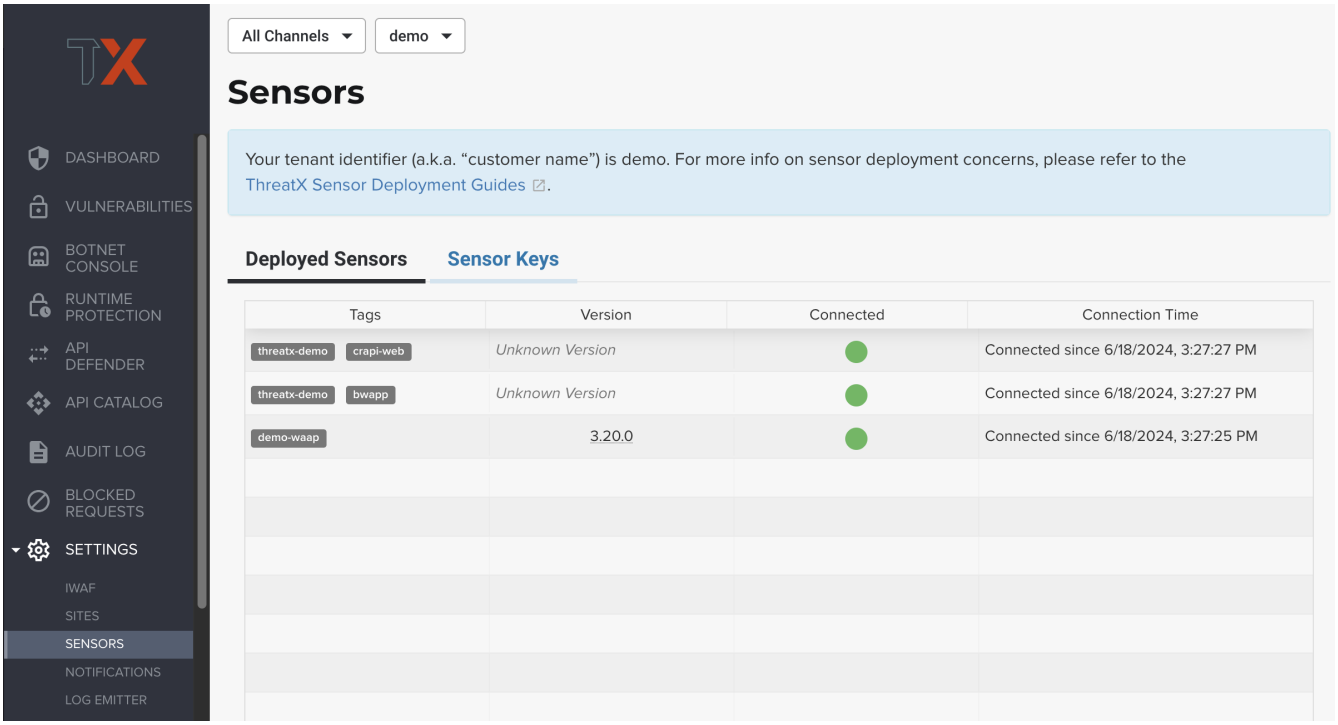


Figure 4. Analyzer tags seen as Tags on the ThreatX Sensors page.

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

Analyzer event sampling

The `analyzer.enableSampling` property controls the sampling of API Analyzer events.

When enabled, it caches duplicate API Analyzer Events to reduce the number reported to the ThreatX backend. It is enabled by default to reduce egress traffic

We recommend setting the sampling to `false` when initially testing out a deployment, but then flipping it back to `true` after the deployment has been verified.

Runtime Sensor Deployment

Sidecar Injector

The *Sidecar Injector* is a Kubernetes Mutating Admission Webhook service that will inject ThreatX runtime sensor containers into application pods based upon the presence of a pod label.

Adding a sensor to an application

The Sidecar Injector will inject the runtime sensor container into any pods created with this label

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

Sample Pod resource spec with inject label added

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: sample-app
    inject-threatx-sidecar: "true"
  name: sample-app
  namespace: sample
spec:
  containers:
  - name: sample-app
 # ...
```

This label should typically be added to the application's Kubernetes Deployment, Statefulset, or Daemonset resource. This will ensure that all created pods by that resource get a sensor injected.

Sample Deployment resource spec with inject label added

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: sample-app
  namespace: sample
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
        inject-threatx-sidecar: "true"
    spec:
      containers:
      - name: dvwa
    # ...
```

Simply adding the label to a resource with pre-existing pods will not automatically inject those pods; **you will need to restart them** (e.g. with `kubectl rollout restart` and so on). This is because Kubernetes does not call the webhook until it needs to start the underlying resources.

Sensor injection can be disabled at the namespace level with the following label

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

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

Verifying a sensor is running

The sensor can be verified after injection by describing the application pod. The `public.ecr.aws/threatx/threatx-runtime-sensor` image should be seen running inside the pod.

Helm Chart Configuration Reference

Table 5. All Helm Chart Values

Key	Type	Default	Description
`apiKey`	string	""	Your ThreatX api key
`customer`	string	""	Your ThreatX customer ID
`certManager.enabled`	boolean	false	Use your cluster's cert-manager component to provision certificates for the sidecar injector. See Certificates
`analyzer.enabled`	boolean	true	Install the Runtime Analyzer service

Key	Type	Default	Description
analyzer.instances	int	2	The number of Analyzer instances to run
analyzer.image.repository	string	public.ecr.aws/threatx/threatx-runtime-analyzer"	Runtime Analyzer image repository
analyzer.image.tag	string	1.2.1	Runtime Analyzer image tag
analyzer.image.pullPolicy	string	"IfNotPresent"	Runtime Analyzer image pull policy. See Image Pull Policy for more information.
analyzer.gatewayHostname	string	threatx-grpc2kafka-production-v1.xplat-production.threatx.io	The hostname of the ThreatX gateway server
analyzer.tags	string	""	Tags for your ThreatX analyzers which are visible in the CtrlX dashboard
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	compress the API Analyzer Events sent from the Analyzer to STS
analyzer.rawAaeAcceptCompressed	boolean	false	allow compressed events from STS
analyzer.enableSampling	boolean	true	cache duplicate API Analyzer Events to reduce the number sent to the ThreatX backend
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	"info"	The logging level
analyzer.backtrace	int	1	The logging backtrace level

Key	Type	Default	Description
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. 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
analyzer.scaling.cpuUtilPercentage	int	200	The percentage of the request cpu limit (analyzer.resources.requests.cpu) to use as a scaling threshold. See: How does a horizontalpodautoscaler work
otel.enabled	boolean	true	Install the Threatx OTEL service
otel.hostname	string	""	The hostname of the ThreatX OTEL server that will receive log and event data
otel.*			Options for the Grafana Alloy Helm Chart, See: grafana.com/docs/alloy/latest/configure/kubernetes
sensor.image.repository	string	public.ecr.aws/threatx/threatx-runtime-sensor"	ThreatX Prevent sensor image repository
sensor.image.tag	string	1.2.1	ThreatX Prevent sensor image tag
sensor.image.pullPolicy	string	"IfNotPresent"	ThreatX Prevent sensor image pull policy. See Image Pull Policy for more information.
sensor.applicationNameLabels	list	["app.kubernetes.io/name","app","name"]	List of pod labels to use for deriving the pod's application name. See Sensor Tags
sensor.interfaceName	string	"eth0"	The host network interface name.

Key	Type	Default	Description
sensor.tracingPath	string	"/sys"	The host tracing path. S
sensor.logLevel	string	"info"	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.
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 ThreatX Prevent Scan Template Service (STS)
sts.instances	int	2	The number of Scan Template Service instances to run
sts.image.repository	string	public.ecr.aws/threatx/threatx-sts"	Scan Template Service image repository
sts.image.tag	string	1.2.0	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	boolean	true	TLS enabled
sts.grpcListenPort	string	"50051"	The gRPC listener port
sts.externalSecret	boolean	false	The secrets for the STS will be managed outside of the Helm chart. See External Secrets
sts.caCert	string	""	The base64 encoded CA pem to use for the STS. See Self Managed Certificates
sts.serverCert	string	""	The base64 encoded CA pem to use for the STS. See Self Managed Certificates
sts.serverKey	string	""	The base64 encoded CA pem to use for the STS. See Self Managed Certificates
sts.logLevel	string	"info"	The logging level

Key	Type	Default	Description
sts.resources.requests.cpu	string	"500m"	Amount of CPU units that the STS container requests for scheduling. See Requests and Limits for more information.
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	boolean	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/threatx-sidecar-injector"	ThreatX Prevent sidecar injector image repository
sidecarInjector.image.tag	string	1.2.0	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.
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.

Key	Type	Default	Description
`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	Regenerate certificates for the control plane services.