Menu

Navigation

Overview

Introduction

Architecture

Installation

Alauda Container Security Plugin Installation

Roxctl CLI Installation

Using Alauda Container Security in Offline Mode

Dashboards

Viewing Dashboard

Network

Introduction

Guides

HowTo

Violation

Introduction

Guides

Compliance

Introduction

Guides

How to

Vulnerablitiy

Introduction

Guides

How to

Risk

Introduction

Guides

Security Policy

Introduction

Guides

How To

Configuration

Managing Deployment Collections

Learn how to manage deployment collections in Alauda Container Security.

API Token Configuration

Integrating with a Generic Docker Registry

Integration with Email

Menu

Overview

Introduction

What is Alauda Container Security?

What Problems Does Alauda Container Security Solve?

Limitations

Architecture

System Architecture

Component Interactions

Introduction

TOC

What is Alauda Container Security?

What Problems Does Alauda Container Security Solve?

Limitations

What is Alauda Container Security?

Alauda Container Security is a comprehensive security solution designed for Kubernetes and containerized environments. It provides centralized management, automated vulnerability scanning, policy enforcement, and compliance checks to help organizations secure their container infrastructure across multiple clusters.

Alauda Container Security adopts a distributed, container-based architecture, consisting of Central Services (for management, API, and UI) and Secured Cluster Services (for monitoring, policy enforcement, and data collection). It integrates with CI/CD pipelines, SIEM, logging systems, and supports the built-in Scanner V4 vulnerability scanner.

What Problems Does Alauda Container Security Solve?

- **Vulnerability Management:** Alauda Container Security continuously scans container images and running workloads for known vulnerabilities, helping prevent exploitation that could lead to denial of service, remote code execution, or unauthorized data access.

- **Policy Enforcement:** By defining and enforcing security policies, Alauda Container Security helps prevent high-risk deployments and enables timely response to runtime security incidents.
- **Compliance Automation:** Alauda Container Security automates compliance checks against industry standards and regulatory frameworks CIS, providing clear visibility into compliance status and helping organizations address gaps efficiently.
- **Centralized Visibility:** It offers a unified portal for monitoring security events, compliance status, and policy violations across all protected clusters.
- **Integration:** Alauda Container Security integrates with CI/CD pipelines, image registries, and third-party tools to streamline security throughout the container lifecycle.

Limitations

- **Scanner Compatibility:** Scanner V4 is the default and only supported vulnerability scanner.
- **Deployment Constraints:** Central Services must be deployed on a single cluster, and certain scanner components may only be deployed in specific namespaces or configurations.
- **Kubernetes Focus:** Alauda Container Security is primarily designed for Kubernetes and Alauda Container Platform environments; support for other orchestration platforms may be limited.
- **External Dependencies:** Vulnerability data relies on external feeds (e.g., definitions.stackrox.io), and some integrations require additional configuration.
- **Resource Requirements:** Running multiple components (Central, Sensor, Collector, Scanner) may require adequate cluster resources and network configuration.

Alauda Container Security empowers organizations to proactively manage risks, enforce best practices, and maintain compliance in modern containerized environments.

Architecture

TOC

System Architecture
Abstract
Key Components
Scanner Overview
Vulnerability Sources
Deployment Notes
External Integrations
Component Interactions
Alauda Container Security with Scanner V4
Default Ports and Protocols

System Architecture

Abstract

This document provides a concise overview of the Alauda Container Security architecture for Kubernetes environments.

Alauda Container Security adopts a distributed, container-based architecture for scalable, low-impact security on Alauda Container Platform or Kubernetes clusters.

Key Components

- **Central Services**: Deployed on a single cluster, providing management, API, and UI (Alauda Container Security Portal). Includes Central, Central DB (PostgreSQL 13), and the Scanner V4 vulnerability scanner.
- **Secured Cluster Services**: Deployed on each protected cluster. Includes Sensor (cluster monitoring and policy enforcement), Admission Controller (policy admission), Collector (runtime and network data collection), and optional scanner components.

Scanner Overview

- **Scanner V4**: The default and only supported scanner since version 4.7. Supports language and OS-specific image scanning. Consists of Indexer, Matcher, and DB.

Vulnerability Sources

- Scanner V4: Red Hat VEX, Red Hat CVE Map, OSV, NVD, and additional OS sources.

Deployment Notes

- Operator installs a lightweight Scanner V4 on each cluster for integrated registry scanning.
- Helm installs require `scannerV4.disable=false` to enable the lightweight Scanner V4.
- If Central and secured cluster services share a namespace, only Central deploys Scanner V4 components.

External Integrations

- Third-party systems (CI/CD, SIEM, logging, email)
- roxctl CLI
- Image registries (auto/manual integration)
- definitions.stackrox.io (vulnerability feeds)
- collector-modules.stackrox.io (kernel modules)

Component Interactions

Alauda Container Security with Scanner V4

Component	Direction	Component	Description
Central	⇄	Scanner V4 Indexer	Image indexing and report generation
Central	⇄	Scanner V4 Matcher	Vulnerability matching and reporting
Sensor	⇄	Scanner V4 Indexer	Delegated image indexing
Scanner V4 Indexer	→	Image Registries	Pulls image metadata and layers
Scanner V4 Matcher	→	Scanner V4 Indexer	Fetches index reports
Scanner V4 Indexer	→	Scanner V4 DB	Stores indexing results
Scanner V4 Matcher	→	Scanner V4 DB	Stores and updates vulnerability data
Sensor	⇄	Central	Configuration and event sync
Collector	⇄	Sensor	Sends runtime/network data
Admission controller	⇄	Sensor	Policy enforcement and scan requests
Admission controller	→	Central	Direct communication if Sensor unavailable

Default Ports and Protocols

Connection	Type	Port	Notes
Central ↔ Scanner V4 Indexer	gRPC	8443	
Central ↔ Sensor	TCP/gRPC	443	Bidirectional, Sensor initiates
Central ↔ CLI	gRPC/HTTPS	443	See `roxctl` for options
Central ↔ Vulnerability feeds	HTTPS	443	definitions.stackrox.io
Collector → Sensor	gRPC	443	
Collector (Compliance) → Sensor	gRPC	8444	If node scanning enabled
Scanner V4 Indexer → Central	HTTPS	443	
Scanner V4 Indexer/Matcher → DB	TCP	5432	
Sensor ↔ Admission Controller	gRPC	443	Bidirectional

Menu

Installation

Alauda Container Security Plugin Installation

Installation Requirements

Central Service Plugin Installation

Cluster Service Plugin Installation

Plugin Uninstallation

Roxctl CLI Installation

Overview

Installation

Verification

Configuration

Using the roxctl CLI

Using Alauda Container Security in Offline Mode

Enabling Offline Mode

Updating Vulnerability Definitions

Alauda Container Security Plugin Installation

This guide provides step-by-step instructions for installing the Alauda Container Security plugin.

TOC

- Installation Requirements
- Central Service Plugin Installation
  - Pre-installation Steps
  - Install via UI
  - Install via YAML
  - Access Central Console
- Cluster Service Plugin Installation
  - Generate Cluster Access Certificate
  - Cluster Pre-installation Steps
  - Cluster Install via UI
  - Cluster Install via YAML
- Plugin Uninstallation

Installation Requirements

- Architecture: `amd64`
- Kernel version: `>=5.8`
- Resource requirements:

  - CPU: `>=4`
  - Memory: `>=8GB`
- PostgreSQL: `>=13`
- TLS Certificate

Central Service Plugin Installation

Pre-installation Steps

1. Create the stackrox namespace:

   ```
   kubectl create ns stackrox
   ```

2. Create the central-db-password secret:

   Store the password in the `password` data item.

   ```
   kubectl create secret generic central-db-password \
     --from-literal=password=<central db password> \
     -n stackrox
   ```

3. Enable Ingress and configure domain certificate:

   Create a TLS secret with your certificate and key.

   ```
   kubectl create secret tls central-ingress-tls \
     --cert=<path/to/tls.crt> \
     --key=<path/to/tls.key> \
     -n stackrox
   ```

4. Create additional CA secret :

```
kubectl create secret generic additional-ca \
  --from-file=00-ingress-ca.crt=<path-to-cert-file> \
  -n stackrox
```

Install via UI

1. In Platform Management, go to **Marketplace** → **Cluster Plugins**.
2. Click the **Install** button next to the Central Service for StackRox plugin.
3. Fill in the storage class and configuration parameters as prompted.

Parameters	Description
Central Database Connection String	The connection string for the central database.For example: `host=acid-business-1.proj01-postgres.svc port=5432 user=postgres sslmode=require`
Host	Specify a custom hostname for the central ingress. Specify a "central-ingress-tls" ts type secret in stackrox namespace that contains tls.crt, tls.key.

Install via YAML

Apply the following YAML to your target cluster:

```yaml
# YAML Deployment Method for StackRox Cluster Plugin
# Create a StackRox Central Services plugin instance
---
apiVersion: cluster.alauda.io/v1alpha1
kind: ClusterPluginInstance
metadata:
  annotations:
    cpaas.io/display-name: stackrox-central-services
  labels:
    create-by: cluster-transformer
    manage-delete-by: cluster-transformer
    manage-update-by: cluster-transformer
  name: stackrox-central-services
spec:
  pluginName: stackrox-central-services
  config:
    env:
      offlineMode: true     # Whether to run StackRox in offline mode.
    central:
      db:
        source:
          connectionString: "host=acid-business-1.proj01-postgres.svc port=54
      exposure:
        ingress:
          enabled: true
          host: "example.com"
        loadBalancer:
          enabled: false
```

Access Central Console

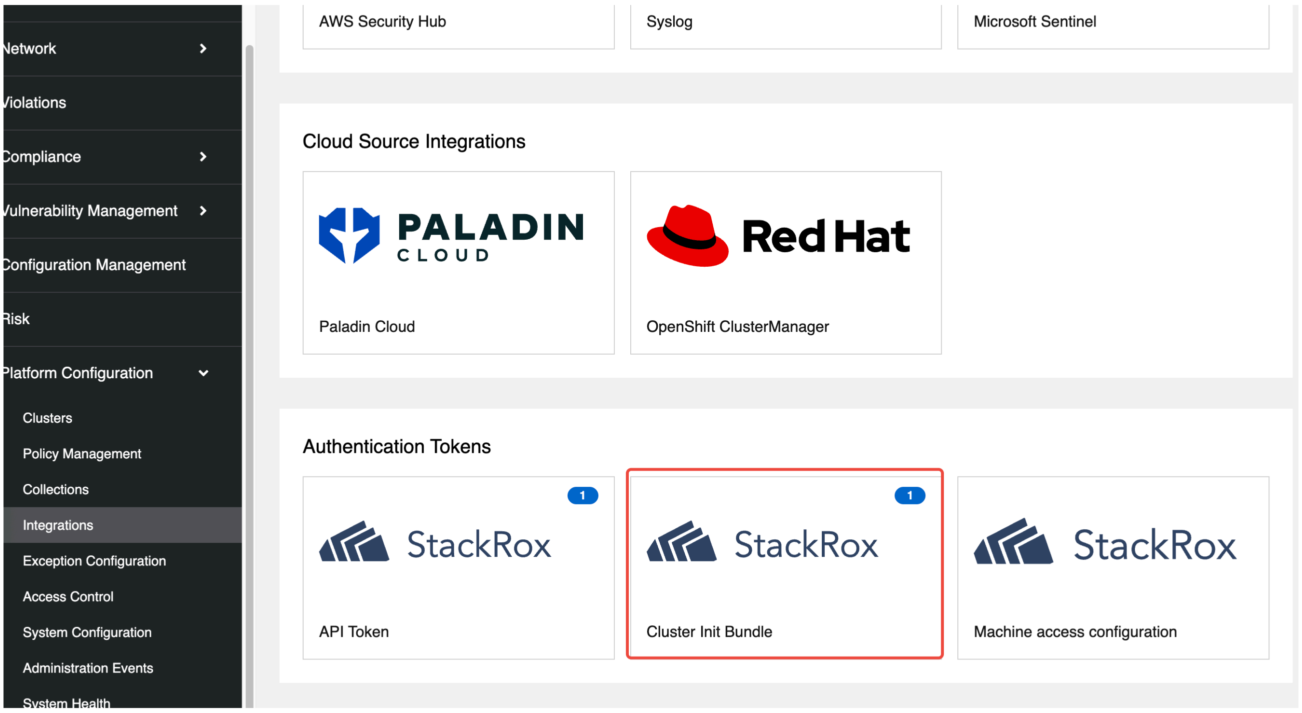
- **Address:** `https://example.com` (The address of the central ingress host)
- **Initial account:** `admin/07Apples@`

Cluster Service Plugin Installation

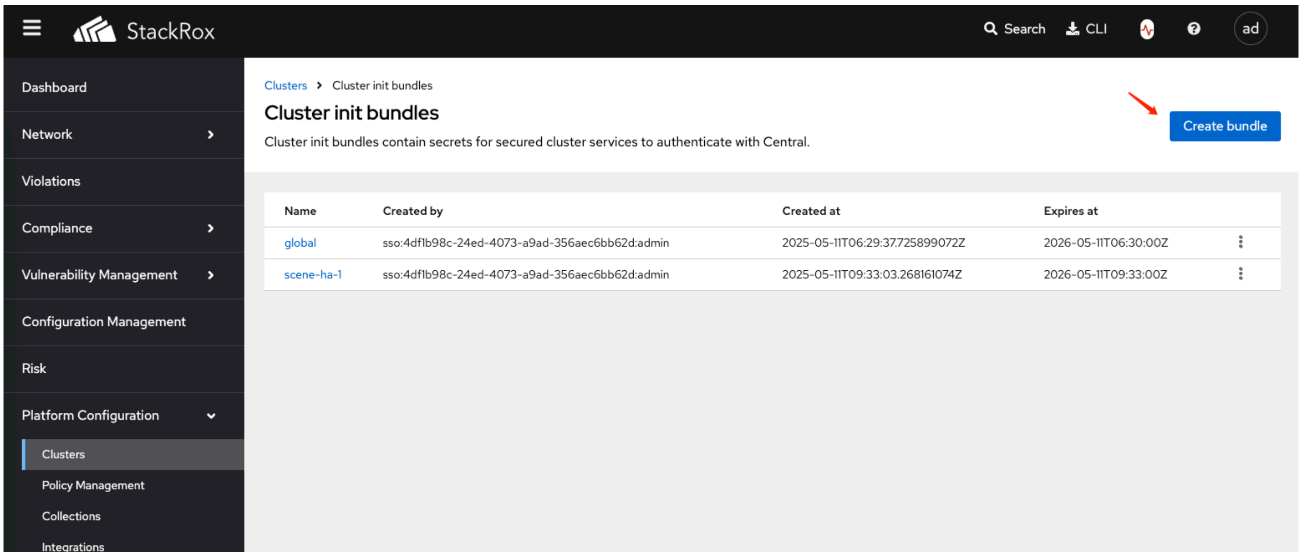
Generate Cluster Access Certificate

1. In Platform Configuration, go to **Platform Configuration** → **Integrations**.

2. Click the button of **Authentication Tokens --> Cluster Init Bundle**



3. Click **Create bundle**.

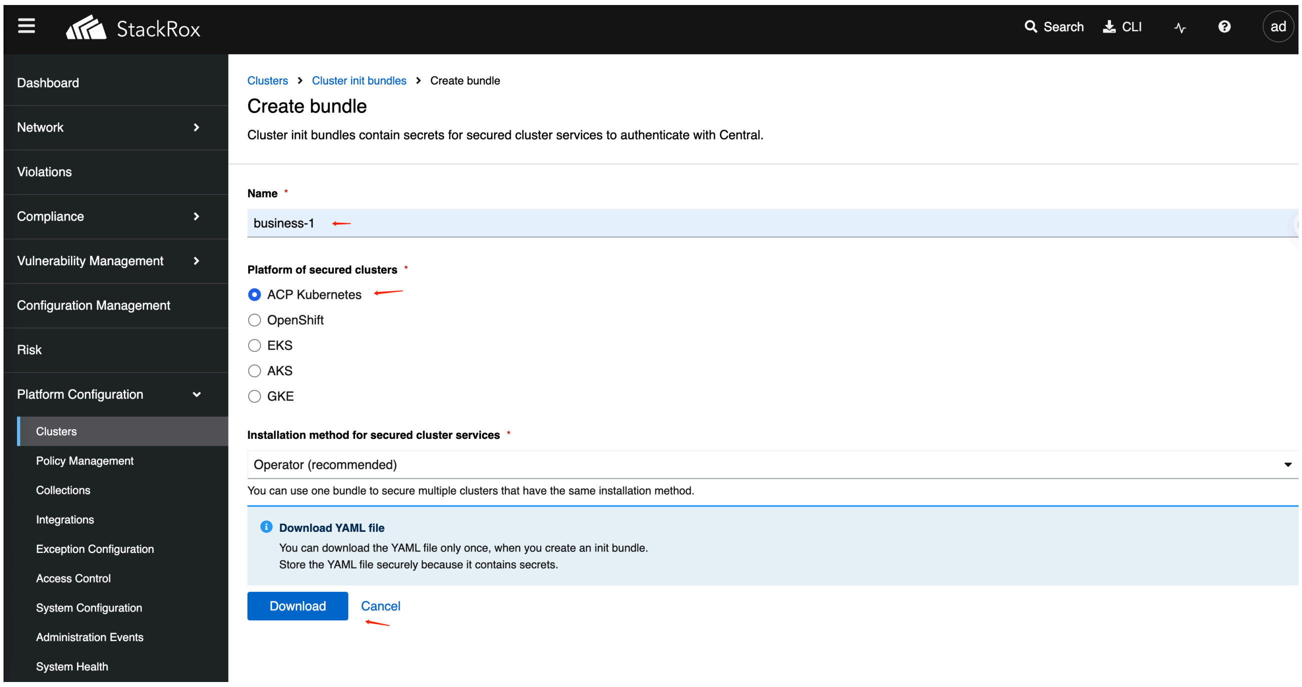


Cluster init bundles

Cluster init bundles contain secrets for secured cluster services to authenticate with Central.

Name	Created by	Created at	Expires at
global	sso:4df1b98c-24ed-4073-a9ad-356aec6bb62d:admin	2025-05-11T06:29:37.725899072Z	2026-05-11T06:30:00Z
scene-ha-1	sso:4df1b98c-24ed-4073-a9ad-356aec6bb62d:admin	2025-05-11T09:33:03.268161074Z	2026-05-11T09:33:00Z

4. Enter the name of the access cluster and download the generated file (e.g., `business-1-Operator-secrets-cluster-init-bundle.yaml`).



Create bundle

Cluster init bundles contain secrets for secured cluster services to authenticate with Central.

Name *

business-1

Platform of secured clusters *

[✓ ACP Kubernetes] OpenShift EKS AKS GKE

Installation method for secured cluster services *

Operator (recommended)

You can use one bundle to secure multiple clusters that have the same installation method.

Download YAML file

You can download the YAML file only once, when you create an init bundle.
Store the YAML file securely because it contains secrets.

Cluster Pre-installation Steps

1. Create the stackrox namespace:

```
kubectl create ns stackrox
```

2. Apply the cluster init bundle secret:

```
kubectl apply -f business-1-Operator-secrets-cluster-init-bundle.yaml -n st
```

3.If the central domain uses an untrusted certificate, you need to create the additional-ca-sensor secret. **Configure Ingress domain certificate :**

```
kubectl create secret generic additional-ca-sensor \
  --from-file=00-ingress-ca.crt=<path-to-cert-file> \
  -n stackrox
```

Cluster Install via UI

1. In Platform Management, go to **Marketplace** → **Cluster Plugins**.
2. Click the **Install** button next to the Cluster Service for StackRox plugin.
3. Fill in the storage class and configuration parameters as prompted.

> **Note:** The Central service address is the access address, e.g., `wss://example.com:443`.

Cluster Install via YAML

Apply the following YAML to your target cluster:

```
apiVersion: cluster.alauda.io/v1alpha1
kind: ClusterPluginInstance
metadata:
  annotations:
    cpaas.io/display-name: stackrox-secured-cluster-services
  labels:
    create-by: cluster-transformer
    manage-delete-by: cluster-transformer
    manage-update-by: cluster-transformer
  name: stackrox-secured-cluster-services
spec:
  pluginName: stackrox-secured-cluster-services
  config:
    centralEndpoint: "wss://example.com:443" # Specify the address of StackRo
```

Plugin Uninstallation

After uninstalling the plugin, manually clean up the following secret resources in the `stackrox` namespace if they remain:

Secret Name Pattern	Description
central-*	Central related secrets
scanner-*	Scanner related secrets
sensor-tls	Sensor TLS secret
service-ca	Service CA secret
admission-control-tls	Admission control TLS
collector-tls	Collector TLS secret
stackrox-generated-once	One-time generated secret

Roxctl CLI Installation

This guide provides instructions for installing, configuring, and using the `roxctl` command-line interface (CLI) for Alauda Container Security. The CLI is available for Linux, macOS, Windows, and as a container image.

TOC

- Overview
- Installation
  - Install on Linux
  - Install on macOS
  - Install on Windows
- Verification
- Configuration
  - Setting Environment Variables
  - Authentication Methods
    - API Token
- Using the roxctl CLI
  - Check Authentication and User Info

Overview

- **roxctl** is a CLI tool for managing and interacting with Alauda Container Security.
- Supported platforms: **Linux**, **macOS**, **Windows**.

- After installation, verify the CLI version to ensure correct setup.

Installation

Install on Linux

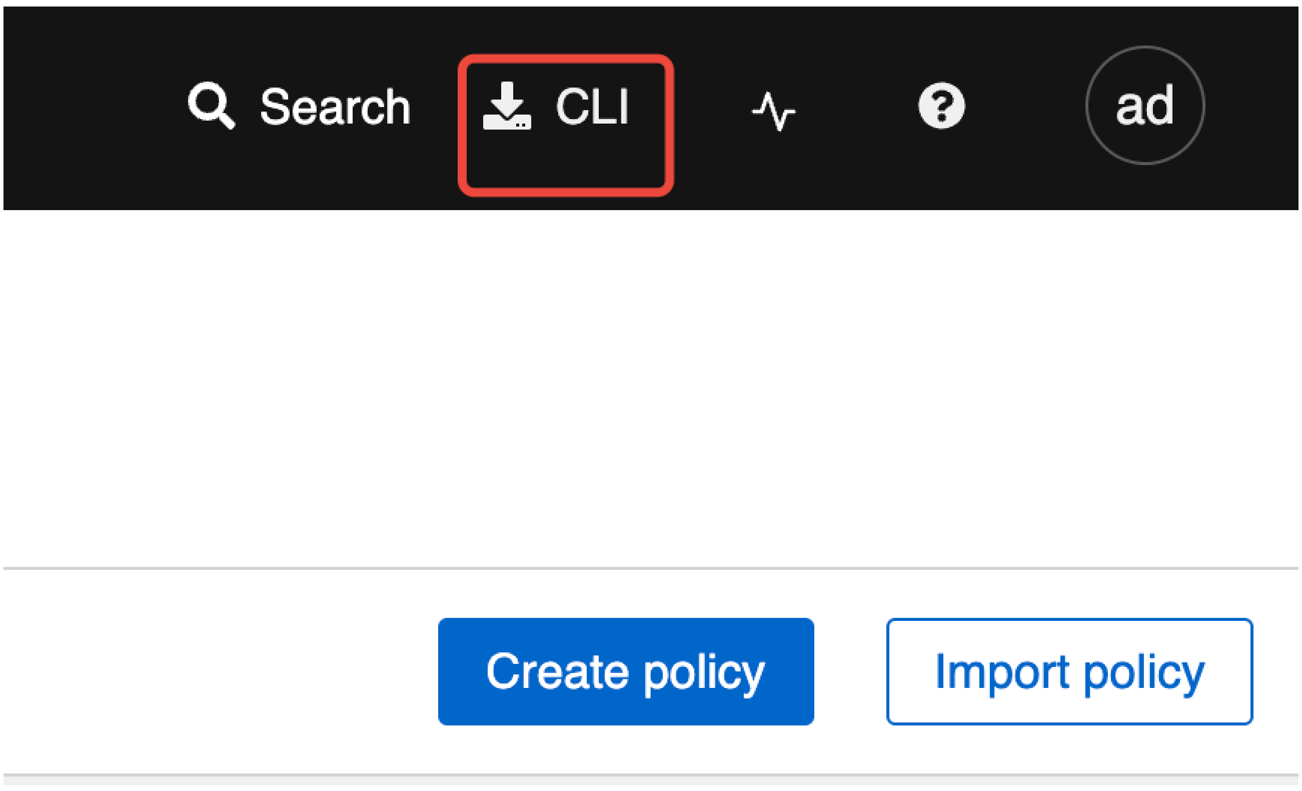
> Supported architectures: `amd64`, `arm64`, `ppc64le`, `s390x`

Steps:

1. Determine your architecture:

   ```
   arch="$(uname -m | sed 's/x86_64//')"; arch="${arch:+-$arch}"
   ```

2. Download the binary form the Portal:



3. Make it executable:

```
chmod +x roxctl
```

4. (Optional) Move to a directory in your `PATH`:

```
echo $PATH
# mv roxctl /usr/local/bin/ # if you want to move it to a directory in your
```

Install on macOS

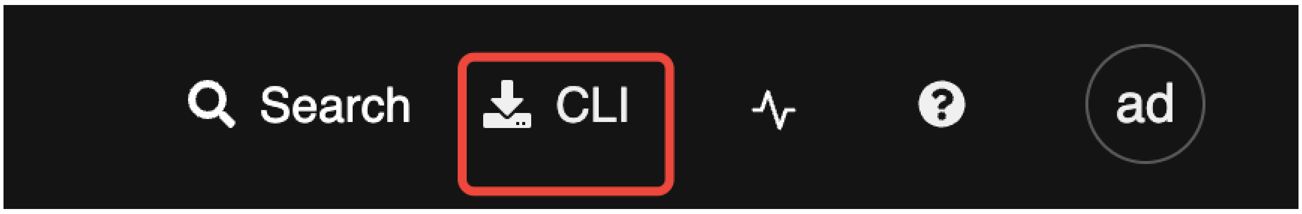
> Supported architectures: `amd64`, `arm64`

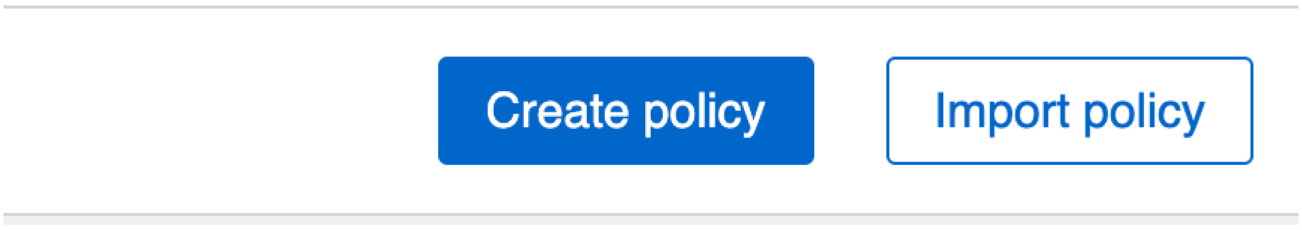
Steps:

1. Determine your architecture:

```
arch="$(uname -m | sed 's/x86_64//')"; arch="${arch:+-$arch}"
```

2. Download the binary form the Portal:





3. Remove extended attributes:

```
xattr -c roxctl
```

4. Make it executable:

```
chmod +x roxctl
```

5. (Optional) Move to a directory in your `PATH`:

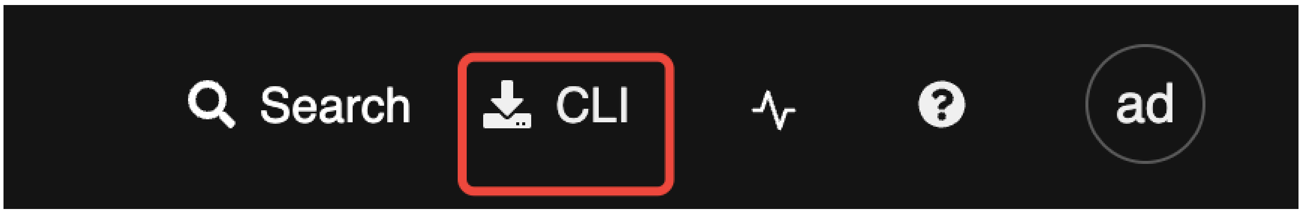
```
echo $PATH
# mv roxctl /usr/local/bin/
```

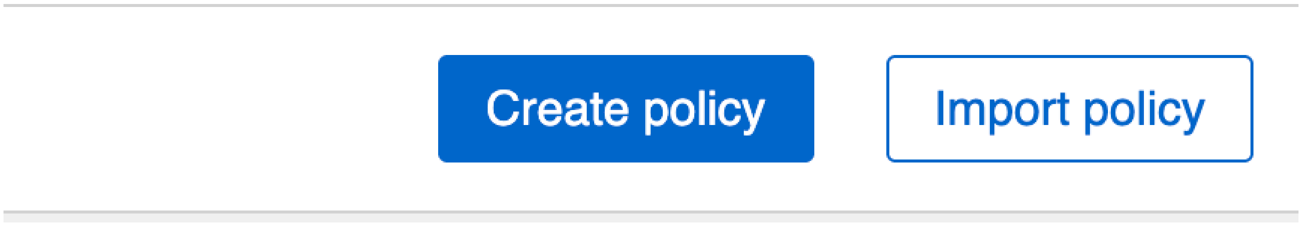
Install on Windows

> Supported architecture: `amd64`

Steps:

2. Download the binary form the Portal:





3. (Optional) Add the directory containing `roxctl.exe` to your system `PATH`.

4. Verify the installation:

```
roxctl.exe version
```

Verification

After installation, verify your `roxctl` version:

```
roxctl version
```

Configuration

Setting Environment Variables

Before using `roxctl`, set the required environment variables:

```
export ROX_ENDPOINT=<central_host:port>
```

If you use an API token for authentication:

```
export ROX_API_TOKEN=<api_token>
```

Alternatively, you can use the `--token-file` option:

```
roxctl central debug dump --token-file <token_file>
```

Note:

- Do not use both `--password` and `--token-file` options at the same time.
- If both `ROX_API_TOKEN` and `--token-file` are set, the CLI uses the token file.
- If both `ROX_API_TOKEN` and `--password` are set, the CLI uses the password.

Authentication Methods

You can authenticate using an API token, administrator password (for testing only), or via the `roxctl central login` command.

API Token

API tokens are recommended for production and automation scenarios. They provide specific access permissions and are valid for up to one year.

To generate an API token:

1. In the Alauda Container Security portal, go to **Platform Configuration > Integrations**.
2. Under **Authentication Tokens**, click **API Token**.

3. Click **Generate Token**.
4. Enter a name and select a role with the required access.
5. Click **Generate** and securely store the token.

To use the token:

```
export ROX_API_TOKEN=<api_token>
```

Using the roxctl CLI

Check Authentication and User Info

To view your current authentication status and user profile:

```
roxctl central whoami
```

Example output:

```
UserID:         <redacted>
User name:      <redacted>
Roles:          Admin, Analyst, Continuous Integration, etc.
Access:         rw Access, rw Administration, rw Alert, ...
```

Review the output to ensure your authentication and permissions are correct.

Using Alauda Container Security in Offline Mode

Alauda Container Security can be deployed in environments without internet access by enabling offline mode. In offline mode, all components operate without connecting to external addresses or hosts.

TOC

Enabling Offline Mode
Updating Vulnerability Definitions
Downloading the Definitions
Uploading the Definitions to Central
Using an API Token

Enabling Offline Mode

- When installing via YAML, set `env.offlineMode` to `true`.

Updating Vulnerability Definitions

Scanner maintains a local vulnerability database. In online mode, Central retrieves the latest vulnerability data from the internet, and Scanner syncs with Central. In offline mode, you must manually update the vulnerability data by uploading a definitions file to Central, which Scanner then retrieves.

- Scanner checks for new data from Central every 5 minutes by default.
- The offline data source is updated approximately every 3 hours.

Downloading the Definitions

- Use the following command to download the definitions:

  ```
  roxctl scanner download-db --scanner-db-file scanner-vuln-updates.zip
  ```

- Alternatively, download from: https://install.stackrox.io/scanner/scanner-vuln-updates.zip

Uploading the Definitions to Central

You can upload the vulnerability definitions database to Central using either an API token or your administrator password.

Using an API Token

- Prerequisites:
  - API token with administrator role
  - `roxctl` CLI installed
- Procedure:

  ```
  export ROX_API_TOKEN=<api_token>
  export ROX_CENTRAL_ADDRESS=<address>:<port_number>
  roxctl scanner upload-db \
    -e "$ROX_CENTRAL_ADDRESS" \
    --scanner-db-file=<compressed_scanner_definitions.zip>
  ```

Dashboards

Viewing Dashboard

Introduction

Status Bar

Dashboard Widgets

Viewing Dashboard

TOC

Introduction
Status Bar
Dashboard Widgets
Violations by Severity
Top Vulnerable Images
Top Risky Deployments
Image Age Distribution
Policy Violations by Category
Compliance by Standard

Introduction

The Alauda Container Security Dashboard provides a centralized view of your cluster's security and compliance posture. This document introduces the Dashboard's main components and explains how to use its features to monitor and manage your environment effectively.

Status Bar

The Status Bar offers a quick overview of key resources in your environment and provides direct navigation to detailed resource lists. The counters reflect your current access scope, as defined by your user roles.

Counter	Destination
Clusters	Platform Configuration Clusters
Nodes	Configuration Management Application & Infrastructure Nodes
Violations	Violations main menu
Deployments	Configuration Management Application & Infrastructure Deployments
Images	Vulnerability Management Dashboard Images
Secrets	Configuration Management Application & Infrastructure Secrets

Use the Status Bar to quickly access detailed lists of clusters, nodes, violations, deployments, images, and secrets.

A top-level filter applies to all widgets. You can select clusters and namespaces to narrow the data shown. If no selection is made, the view defaults to **All**. Changes to the filter are instantly reflected in all widgets, except the Status Bar.

Widgets are customizable, allowing you to sort, filter, and adjust their output. Customization options include:

- An **Options** menu for widget-specific settings.
- A **dynamic axis legend** to filter data by hiding or showing axis categories. For example, in the **Policy Violations by Category** widget, you can include or exclude violations by severity.

> **Note:**
> Widget customization settings are temporary and reset to defaults when you leave the Dashboard.

Dashboard Widgets

The Dashboard provides several actionable widgets to help you monitor and manage security risks and compliance. Each widget is described below with its main function and usage tips.

Violations by Severity

This widget helps you quickly identify the most critical policy violations in your environment. It displays the distribution of violations by severity for the filtered scope. Click a **severity level** to navigate to the **Violations** page, filtered accordingly. The widget also lists the three most recent **Critical** policy violations within the selected scope. Click a violation to open its detail page and take action.

Top Vulnerable Images

This widget highlights images with the highest risk, allowing you to prioritize remediation. It shows the top six vulnerable images in the filtered scope, sorted by risk priority, along with their critical and important CVEs. Click an image name to view its findings in **Vulnerability Management**. Use the **Options** menu to focus on fixable CVEs or active images.

> **Note:**
> When clusters or namespaces are selected in the Dashboard filter, only active images or those used by deployments in the filtered scope are shown.

Top Risky Deployments

This widget identifies the deployments most at risk in your environment, helping you focus on remediation. It shows the top deployments at risk, including their cluster, namespace, and risk score. Click a deployment to view its risk details, including policy violations and vulnerabilities, and take corrective action as needed.

Image Age Distribution

This widget helps you assess the risk posed by older images, which may contain known vulnerabilities. You can use default or custom age ranges, and view both active and inactive images. Click an age group to see those images in the **Vulnerability Management Images** page. Use this widget to prioritize updating or removing outdated images.

Policy Violations by Category

This widget provides insights into policy compliance challenges by showing the five most violated policy categories. Use the **Options** menu to filter by deploy or runtime violations, and change sorting modes (by highest severity or total violations). Some categories, like "Docker CIS," may have no critical policies, affecting the view depending on the sorting mode. Click a severity level at the bottom of the graph to include or exclude it. This may change the top five categories displayed. Data is filtered by the Dashboard filter.

Compliance by Standard

This widget helps you track compliance with key security benchmarks. It lists the top or bottom six compliance benchmarks, depending on sort order. Use **Options** to sort by coverage percentage. Click a benchmark label or graph to go to the **Compliance Controls** page, filtered by scope and benchmark. Use this widget to focus your compliance efforts where they are needed most.

By leveraging the Status Bar and Dashboard Widgets, you can efficiently monitor, investigate, and improve the security and compliance posture of your Kubernetes environment.

Network

Introduction

Introduction

Guides

Network Graph

Entities in the Network Graph

Network Components

Network Flows

Network Policies

Tips for Using the Network Graph

Viewing Deployment Details in a Namespace

Viewing Network Policies

Managing CIDR Blocks

Network Baseline Management in the Network Graph

How Network Baselining Works

Viewing and Managing Network Baselines

Downloading Network Baselines

Configuring Baseline Observation Period

Enabling Alerts for Anomalous Network Flows

HowTo

Generating Network Policies with Alauda Container Security

Overview

How to Generate Network Policies

Downloading and Applying Policies

Reverting and Deleting Policies

Additional Notes

Introduction

A Kubernetes network policy ↗ is a specification of how groups of pods are allowed to communicate with each other and other network endpoints. These network policies are configured as YAML files. By looking at these files alone, it is often hard to identify whether the applied network policies achieve the desired network topology.

Alauda Container Security gathers all defined network policies from your orchestrator and provides tools to make these policies easier to use.

To support network policy enforcement, Alauda Container Security provides the following tools:

- Network graph
- Network policy generator
- Network policy simulator
- Build-time network policy generator

Menu

Guides

Network Graph

Entities in the Network Graph

Network Components

Network Flows

Network Policies

Tips for Using the Network Graph

Viewing Deployment Details in a Namespace

Viewing Network Policies

Managing CIDR Blocks

Network Baseline Management in the Network Graph

How Network Baselining Works

Viewing and Managing Network Baselines

Downloading Network Baselines

Configuring Baseline Observation Period

Enabling Alerts for Anomalous Network Flows

Network Graph

The network graph in Alauda Container Security offers both high-level and detailed insights into deployments, network flows, and network policies within your environment. It helps you visualize how workloads communicate, monitor real-time and potential network traffic, and manage network security policies efficiently.

Alauda Container Security analyzes all network policies in each secured cluster, showing which deployments can communicate with each other and which can access external networks. It also tracks running deployments and their network traffic. The network graph displays the following core elements:

TOC

- Entities in the Network Graph
  - Internal Entities
  - External Entities
- Network Components
- Network Flows
- Network Policies
- Tips for Using the Network Graph
  - Scenarios for Internal Entities
- Viewing Deployment Details in a Namespace
- Viewing Network Policies
- Managing CIDR Blocks

Entities in the Network Graph

Internal Entities

Internal entities represent connections between a deployment and an IP address within the private address space as defined in RFC 1918↗. For more details, see "Connections involving internal entities".

External Entities

External entities represent connections between a deployment and an IP address outside the private address space as defined in RFC 1918↗. For more details, see "External entities and connections in the network graph".

Network Components

You can use the top menu to select namespaces (**NS** label) and deployments (**D** label) to display on the graph for a chosen cluster (**CL** label). Deployments can be further filtered by CVEs, labels, or images using the drop-down list.

Network Flows

The network graph supports two types of flow visualization:

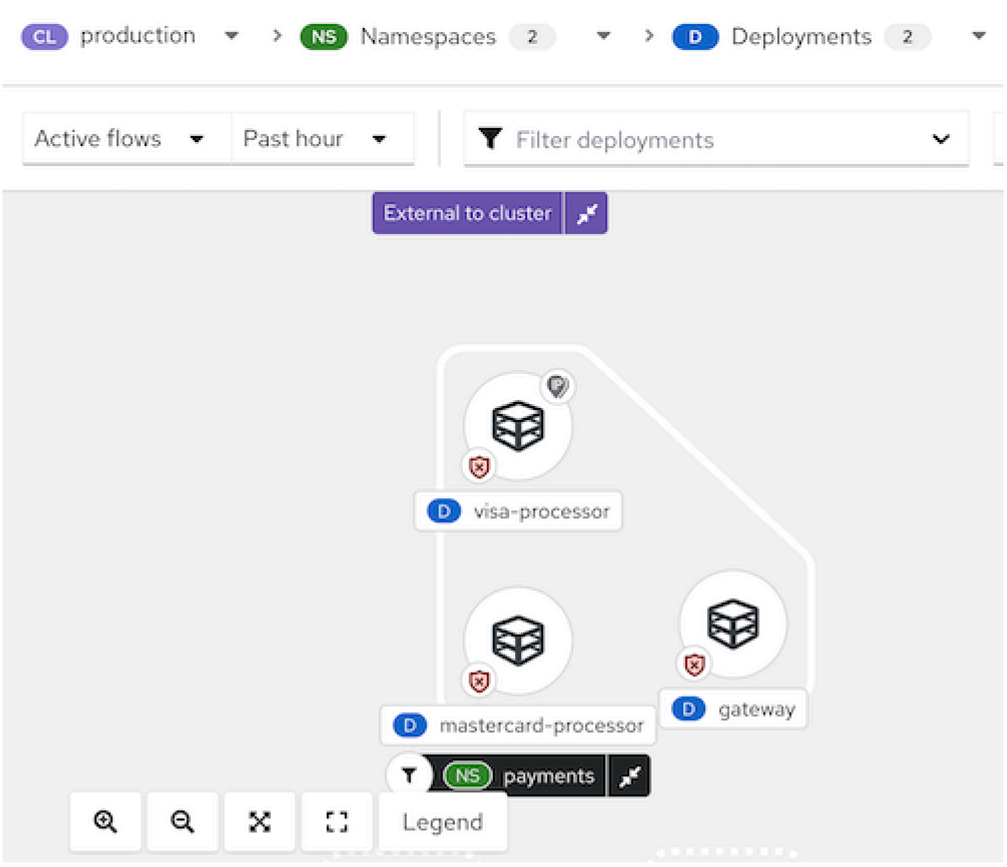
- **Active traffic**: Displays observed, real-time traffic for the selected namespace or deployment. You can adjust the time period for the data shown.
- **Inactive flows**: Shows potential flows allowed by your network policies, helping you identify where additional policies may be needed for tighter isolation.

Network Policies

You can view existing policies for a selected component or identify components without policies. The network graph also allows you to simulate network policies. For more information, see "Simulating network policies from the network graph".

You can interact with the network graph to view more details about items and perform actions such as adding a network flow to your baseline.

Figure 1 Network graph example



Tips for Using the Network Graph

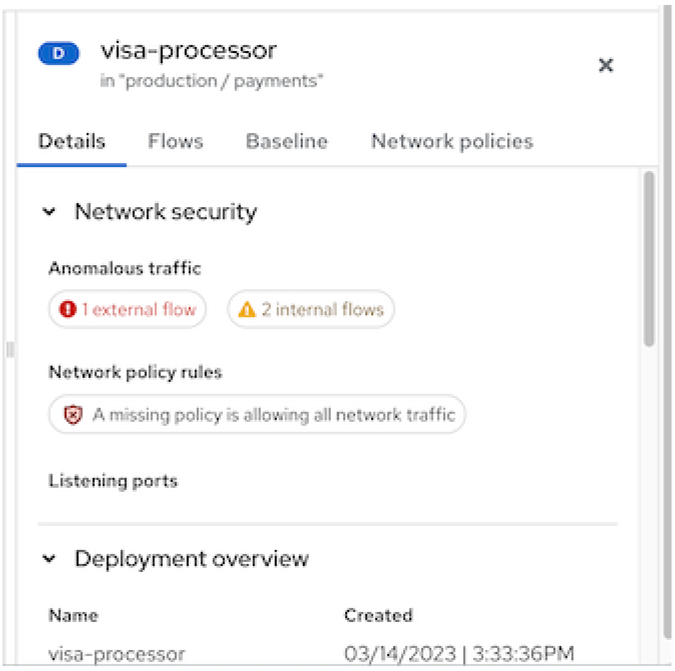
- Open the legend to learn about the symbols used for namespaces, deployments, and connections.
- Use the display options drop-down to show or hide icons such as the network policy status badge, active external traffic badge, and port/protocol labels for edge connections.
- Alauda Container Security detects changes in network traffic, such as nodes joining or leaving. When changes are detected, a notification appears showing the number of updates available. Click the notification to refresh the graph.

When you click an item in the graph, a side panel with collapsible sections presents detailed information about that item. You can select:

- Deployments
- Namespaces
- External entities
- CIDR blocks
- External groups

The side panel displays relevant information based on your selection. The **D** or **NS** label next to the item name (e.g., "visa-processor") indicates whether it is a deployment or a namespace. Below is an example of the side panel for a deployment:

Figure 2 Side panel for a deployment example



D visa-processor
in "production / payments"

Details | Flows | Baseline | Network policies

Network security

Anomalous traffic

1 external flow | 2 internal flows

Network policy rules

A missing policy is allowing all network traffic

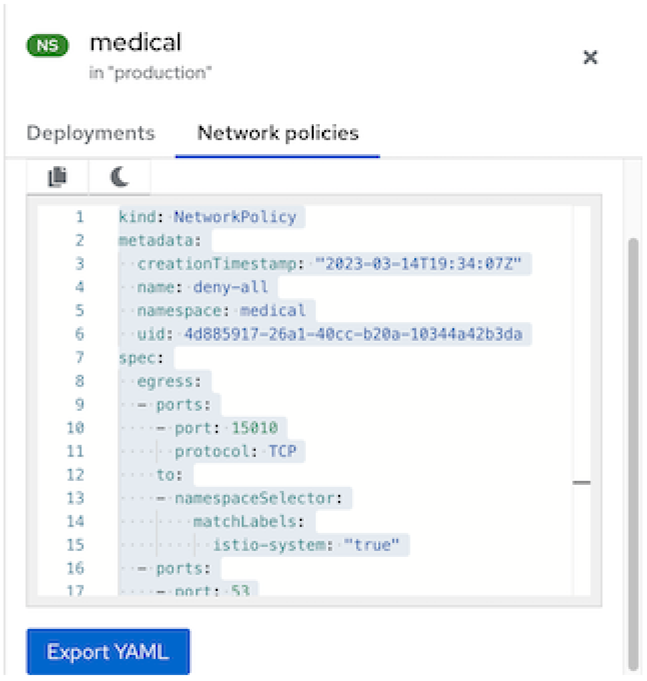
Listening ports

Deployment overview

Name	Created
visa-processor	03/14/2023 | 3:33:36PM

When viewing a namespace, the side panel includes a search bar and a list of deployments. You can click a deployment to view its information. The side panel also includes a **Network policies** tab, from which you can view, copy, or export any network policy defined in that namespace.

Figure 3 Side panel for a namespace example



```yaml
kind: NetworkPolicy
metadata:
  creationTimestamp: "2023-03-14T19:34:07Z"
  name: deny-all
  namespace: medical
  uid: 4d885917-26a1-40cc-b20a-10344a42b3da
spec:
  egress:
  - ports:
    - port: 15010
      protocol: TCP
    to:
    - namespaceSelector:
        matchLabels:
          istio-system: "true"
  - ports:
    - port: 53
```

Scenarios for Internal Entities

Common scenarios for internal entity connections include:

- A change of IP address or deletion of a deployment accepting connections (the server) while the client still attempts to reach it
- A deployment communicating with the orchestrator API
- A deployment communicating using a networking CNI plugin (e.g., Calico)
- A restart of the Sensor, resulting in a reset of the mapping of IP addresses to past deployments (e.g., when the Sensor does not recognize the IP addresses of past entities or past IP addresses of existing entities)
- A connection involving an entity not managed by the orchestrator (sometimes seen as *outside the cluster*) but using an IP address from the private address space as defined in RFC 1918

Internal entities are indicated with an icon. Clicking on **Internal entities** shows the flows for these entities.

Viewing Deployment Details in a Namespace

To view details for deployments in a namespace:

1. In the Alauda Container Security portal, go to **Network Graph** and select your cluster from the drop-down list.
2. Click the **Namespaces** list and use the search field to locate a namespace, or select individual namespaces.
3. Click the **Deployments** list and use the search field to locate a deployment, or select individual deployments to display in the network graph.
4. In the network graph, click on a deployment to view the information panel.
5. Click the **Details**, **Flows**, **Baseline**, or **Network policies** tab to view the corresponding information.

Kubernetes `NetworkPolicy` resources use labels to select pods and define rules specifying what traffic is allowed to or from the selected pods. Alauda Container Security discovers and displays network policy information for all your Kubernetes clusters, namespaces, deployments, and pods in the network graph.

Viewing Network Policies

To view network policies:

1. In the Alauda Container Security portal, go to **Network Graph** and select your cluster from the drop-down list.
2. Click the **Namespaces** list and select individual namespaces, or use the search field to locate a namespace.
3. Click the **Deployments** list and select individual deployments, or use the search field to locate a deployment.
4. In the network graph, click on a deployment to view the information panel.
5. In the **Details** tab, under **Network security**, you can view summary messages about network policy rules, including:

- Whether policies exist in the network that regulate ingress or egress traffic
- Whether your network is missing policies and is therefore allowing all ingress or egress traffic

6. To view the YAML file for the network policies, click on the policy rule or the **Network policies** tab.

Managing CIDR Blocks

You can specify custom CIDR blocks or configure the display of auto-discovered CIDR blocks in the network graph.

To manage CIDR blocks:

1. In the Alauda Container Security portal, go to **Network Graph**, then select **Manage CIDR Blocks**.
2. You can:

   - Toggle **Auto-discovered CIDR blocks** to hide auto-discovered CIDR blocks in the network graph.

     > **Note:** Hiding auto-discovered CIDR blocks applies to all clusters, not just the selected cluster in the network graph.

   - Add a custom CIDR block:

     6.1. Enter the CIDR name and CIDR address in the fields. To add more, click **Add CIDR block** and enter information for each block.

     6.2. Click **Update Configuration** to save the changes.

Network Baseline Management in the Network Graph

Alauda Container Security helps you minimize network risks by using network baselining. This proactive approach secures your infrastructure by learning normal network flows and identifying any deviations as anomalies.

TOC

- How Network Baselining Works
- Viewing and Managing Network Baselines
  - Steps to View Baselines
  - Marking Baseline Flows as Anomalous
  - Additional Options
- Downloading Network Baselines
- Configuring Baseline Observation Period
  - Setting Environment Variables
- Enabling Alerts for Anomalous Network Flows

How Network Baselining Works

When you first install Alauda Container Security , there is no default network baseline. As Alauda Container Security observes network activity, it automatically adds discovered network flows to the baseline:

- New network flows are added to the baseline during the observation phase.
- These flows are considered normal and do not trigger any alerts or violations.

After the observation phase:

- Alauda Container Security stops adding new flows to the baseline.
- Any new network flow not in the baseline is marked as anomalous, but does not trigger violations by default.

Viewing and Managing Network Baselines

You can view and manage network baselines in the network graph interface.

Steps to View Baselines

1. Click the **Namespaces** dropdown and search or select namespaces.
2. Click the **Deployments** dropdown and search or select deployments to display in the network graph.
3. In the network graph, click a deployment to open its information panel.
4. Go to the **Baseline** tab. Use the **filter by entity name** field to narrow down displayed flows.

Marking Baseline Flows as Anomalous

- To mark a single flow as anomalous, select the entity, click the overflow menu, and choose **Mark as anomalous**.
- To mark multiple flows, select them, click **Bulk actions**, and choose **Mark as anomalous**.

Additional Options

- **Exclude ports and protocols**: Check the box to ignore port and protocol information in the baseline.
- **Download as network policy**: Click **Download baseline as network policy** to export the baseline as a YAML file.

Downloading Network Baselines

You can export network baselines as YAML files for further use.

Steps:

1. In the Alauda Container Security portal, go to **Network Graph**.
2. Select the desired namespaces and deployments.
3. In the deployment's information panel, open the **Baseline** tab.
4. (Optional) Filter flows or exclude ports/protocols.
5. Click **Download baseline as network policy**.

Configuring Baseline Observation Period

You can adjust how long Alauda Container Security observes network flows before finalizing the baseline using environment variables.

Setting Environment Variables

Set the following variables in your deployment:

```
kubectl -n stackrox set env deploy/central ROX_NETWORK_BASELINE_OBSERVATION_P
kubectl -n stackrox set env deploy/central ROX_BASELINE_GENERATION_DURATION=<
```

- `<value>` must be a valid time unit, e.g., `300ms` , `2h45m` , `-1.5h` .
- Supported units: `ns` , `us` / `µs` , `ms` , `s` , `m` , `h` .

Enabling Alerts for Anomalous Network Flows

Alauda Container Security can be configured to trigger violations for anomalous network flows (flows not in the baseline).

Steps:

1. In the network graph, select the desired namespace and deployment.
2. Open the **Baseline** tab in the deployment's information panel.
3. Toggle the **Alert on baseline violations** option.

- When enabled, anomalous flows will trigger violations.
- Toggle off to stop receiving such alerts.

Menu

HowTo

Generating Network Policies with Alauda Container Security

Overview

How to Generate Network Policies

Downloading and Applying Policies

Reverting and Deleting Policies

Additional Notes

Generating Network Policies with Alauda Container Security

Alauda Container Security enables you to automatically generate Kubernetes network policies based on observed network flows, helping you secure pod communication and reduce your attack surface.

TOC

Overview

How to Generate Network Policies

Downloading and Applying Policies

Reverting and Deleting Policies

Additional Notes

Overview

Kubernetes network policies define which pods can receive or send network traffic. Manually creating these YAML files can be complex. Alauda Container Security simplifies this by generating policies according to the following principles:

- **One Policy per Deployment:** Alauda Container Security generates a network policy for each deployment in the selected namespace, using the deployment's pod selector.
  - If a deployment already has a network policy, Alauda Container Security will not overwrite or delete it.
  - New deployments are unrestricted until you generate or create new policies for them.

  - If a new deployment needs to access a protected deployment, you may need to update the policy.
- **Naming Convention:** Each policy is named `stackrox-generated-<deployment-name>` and includes an identifying label.
- **Allowing External Traffic:** Alauda Container Security generates a rule allowing traffic from any IP if:
  - The deployment receives connections from outside the cluster during the selected period, or
  - The deployment is exposed via a node port or load balancer service.
- **Ingress Rules:** For each observed incoming connection:
  - If from the same namespace, Alauda Container Security uses the source deployment's pod selector.
  - If from a different namespace, Alauda Container Security uses a namespace selector and automatically labels namespaces as needed.

> **Note:** If a standalone pod lacks labels, the generated policy may allow traffic from/to the entire namespace.

How to Generate Network Policies

You can generate policies for clusters, namespaces, or specific deployments using the Alauda Container Security Network Graph.

To generate network policies, follow these steps:

1. In the Alauda Container Security portal, go to **Network Graph**.
2. Select a cluster and one or more namespaces.
3. (Optional) Select specific deployments or use **Filter deployments** to narrow the scope.
4. Click **Network Policy Generator** in the header.
5. (Optional) In the info panel, select **Exclude ports & protocols** to remove port/protocol restrictions.

6. Click **Generate and simulate network policies**. The selected scope is shown at the top of the panel.
7. (Optional) Copy or download the generated YAML file.
8. (Optional) Click **Compare** to view existing and generated policies side by side.
9. (Optional) Use the **Actions** menu to:

- Share the YAML file with notifiers (e.g., Slack, ServiceNow, webhooks).
- Rebuild rules from active traffic.
- Revert to previously applied YAML.

> **Note:** Some namespaces (e.g., with existing ingress policies or protected namespaces) may not have generated policies.

Downloading and Applying Policies

After generating policies, you can download and apply them to your cluster using the CLI or automated tools.

To apply policies:

```
$ kubectl create -f "<generated_file>.yml"
```

To remove policies:

```
$ kubectl delete -f "<generated_file>.yml"
```

> **Warning:** Always test network policies in a development or test environment before applying to production, as they may disrupt running applications.

Reverting and Deleting Policies

- To revert to a previous policy, use the **Revert rules to previously applied YAML** option in the Alauda Container Security portal.
- To delete all automatically generated policies:

```
$ kubectl get ns -o jsonpath='{.items[*].metadata.name}' | \
xargs -n 1 kubectl delete networkpolicies -l \
'network-policy-generator.stackrox.io/generated=true' -n
```

Additional Notes

- The Network Graph does not visualize generated policies.
- Only ingress traffic is restricted by generated policies; egress policies are not generated.

Violation

Introduction

Introduction

What is a Policy Violation?

How Violations Are Detected

Guides

Responding to Violations

Namespace Conditions for Platform Components

Viewing Violations

Violation Details

Policy Tab

Introduction

TOC

What is a Policy Violation?
How Violations Are Detected

What is a Policy Violation?

Alauda Container Security for Kubernetes allows you to view, investigate, and address policy violations in your clusters. You can quickly identify the root cause of a violation and take corrective actions to improve your security posture.

How Violations Are Detected

Alauda Container Security's built-in policies detect a wide range of security issues, including:

- Vulnerabilities (CVEs)
- Violations of DevOps best practices
- High-risk build and deployment activities
- Suspicious runtime behaviors

You can use the default security policies or define your own custom policies. When an enabled policy is violated, Alauda Container Security reports it as a violation for your review and remediation.

Guides

Responding to Violations

Namespace Conditions for Platform Components

Viewing Violations

Violation Details

Policy Tab

Responding to Violations

Alauda Container Security helps you view, investigate, and address policy violations.

Its built-in policies detect vulnerabilities (CVEs), DevOps best practice violations, risky build/deploy actions, and suspicious runtime behaviors. Violations are reported when enabled policies are not met.

Understanding namespace conditions helps you manage which namespaces belong to the Alauda Container Platform, layered products, and third-party partners.

TOC

- Namespace Conditions for Platform Components
- Viewing Violations
- Violation Details
  - Violation Tab
  - Deployment Tab
    - Container Configuration
    - Port Configuration
    - Security Context
    - Network Policy
- Policy Tab
  - Policy Overview
  - Policy Behavior
  - Policy Criteria

Namespace Conditions for Platform Components

Platform Component	Namespace Condition
Alauda Container Platform	Namespace = `cpaas-system`, Namespace starts with `kube-`
Layered Products	Namespace = `stackrox`, Namespace starts with `acs-operator`, Namespace starts with `open-cluster-management`, Namespace = `multicluster-engine`, Namespace = `aap`, Namespace = `hive`
Third Party Partners	Namespace = `nvidia-gpu-operator`

Alauda Container Security uses the following regex to identify platform workloads:

```
^kube-.*|^alauda-.*|^stackrox$|^acs-operator$|^open-cluster-management$|^mult
```

This definition is not customizable. To see its effect:

1. Click **Search** in the portal.
2. Select **Show Orchestrator Components**.
3. Filter by `Platform Component: true`.

Viewing Violations

1. In the portal, click **Violations**.
2. Tabs let you view violations by category:

- **User Workloads**: User-managed workloads
- **Platform**: Platform and layered services

- **All Violations**: All, including audit log violations

3. Tabs let you view by type:

- **Active**: Unresolved or in build/deploy
- **Resolved**: Addressed or manually resolved
- **Attempted**: Blocked by enforced policies

4. Sort, filter, and view details as needed.

5. To exclude deployments from a policy:

- For one: use the overflow menu, select **Exclude deployment from policy**
- For multiple: use **Row actions** > **Exclude deployments from policy**

Violation Details

The **Violations** page shows:

- **Policy**: Violated policy name
- **Entity**: Where the violation occurred
- **Type**: Entity type (e.g., Deployment, Pod, DaemonSet, Secrets, ConfigMaps, ClusterRoles)
- **Enforced**: Whether enforcement was active
- **Severity**: `Low` , `Medium` , `High` , `Critical`
- **Categories**: Policy category
- **Lifecycle**: `Build` , `Deploy` , `Runtime`
- **Time**: When the violation occurred

Selecting a violation opens a details panel:

Violation Tab

Shows how the policy was violated, including specific values or runtime process details.

Deployment Tab

Shows deployment details:

- **Deployment ID/Name/Type**
- **Cluster/Namespace/Replicas**
- **Created/Updated** times
- **Labels/Annotations/Service Account**

Container Configuration

- **Image Name**
- **Resources**: CPU/Memory requests and limits
- **Volumes**
- **Secrets**: Name and container path
- **Volume Details**: Name, source, destination, type

Port Configuration

- **containerPort**
- **protocol**
- **exposure**
- **exposureInfo**: Internal/external, service name/ID, cluster IP, service port, node port, external IPs

Security Context

- **Privileged**: `true` or `false`

Network Policy

- Lists namespace and network policies; click a policy name to view YAML

Policy Tab

Shows details of the policy that caused the violation.

Policy Overview

- **Severity**
- **Categories**
- **Type**: User or system policy
- **Description**
- **Rationale**
- **Guidance**
- **MITRE ATT&CK**: Related tactics/techniques

Policy Behavior

- **Lifecycle Stage**: `Build` , `Deploy` , `Runtime`
- **Event Source** (for `Runtime`):
  - **Deployment**: Triggered by process/network activity, pod execution, or port forwarding
  - **Audit logs**: Triggered by matching audit log records
- **Response**:
  - **Inform**: Generates a violation
  - **Inform and enforce**: Enforced
- **Enforcement**:
  - **Build**: Fails CI builds for noncompliant images
  - **Deploy**: Blocks creation/update of noncompliant deployments if admission controller is enabled
  - **Runtime**: Deletes pods when events match policy criteria

Policy Criteria

Alauda Container Security supports two deploy-time enforcement types:

- **Hard Enforcement**: Admission controller blocks creation or update of violating deployments

- **Soft Enforcement**: Sensor scales replicas to 0 for violating deployments

Note: By default, certain admin namespaces (e.g., `stackrox`, `kube-system`, `cpaas-system`, `istio-system`) are excluded from enforcement. Requests from service accounts in system namespaces are also bypassed.

For existing deployments, policy changes are enforced at the next relevant Kubernetes event. To reassess, go to **Policy Management** and click **Reassess All**.

Compliance

Introduction

Introduction

Guides

Workload and Cluster Compliance Monitoring

Overview

Key Concepts

Running a Compliance Scan

Viewing Compliance Results

Generating Compliance Reports

How to

How to

Introduction

Alauda Container Security provides automated compliance checks to help your Kubernetes clusters meet industry standards and regulatory requirements. By continuously scanning your environment against benchmarks CIS, it enables you to identify and address compliance gaps efficiently.

With support for multiple compliance frameworks—including CIS, allows you to:

- Evaluate and demonstrate regulatory compliance across your infrastructure.
- Strengthen the security posture of your Kubernetes environment.
- Gain clear visibility into the compliance status of clusters, namespaces, and nodes.

By leveraging these capabilities, you can proactively manage compliance risks, streamline audits, and ensure your containerized workloads operate in accordance with best practices and regulatory requirements.

Guides

Workload and Cluster Compliance Monitoring

Overview

Key Concepts

Running a Compliance Scan

Viewing Compliance Results

Generating Compliance Reports

Workload and Cluster Compliance Monitoring

Alauda Container Security enables you to perform compliance scans to assess the compliance status of your entire infrastructure. The compliance dashboard provides a centralized view, allowing you to filter data and monitor compliance across clusters, namespaces, and nodes.

TOC

Overview
Key Concepts
Running a Compliance Scan
Viewing Compliance Results
  Compliance Dashboard
  By Standard
  By Control
  Filtering Compliance Data
Generating Compliance Reports
  Evidence Report Fields

Overview

By generating detailed compliance reports and focusing on specific standards, controls, and industry benchmarks, you can track and share the compliance status of your environment, ensuring your infrastructure meets required standards.

A compliance scan creates a snapshot of your environment, including alerts, images, network policies, deployments, and host-based data. Data is collected from Sensors and compliance containers running in each Collector pod.

The compliance container gathers:

- Configurations for the container daemon, runtime, and images
- Container network information
- Command-line arguments and processes for the container runtime, Kubernetes, and Alauda Container Platform
- Permissions for specific file paths
- Configuration files for Kubernetes and Alauda Container Platform core services

After data collection, Alauda Container Security analyzes the results, which are available in the compliance dashboard and can be exported as reports.

Key Concepts

- **Control**: A single requirement in an industry or regulatory standard. Alauda Container Security verifies compliance with a control by performing one or more checks.
- **Check**: A specific test performed during a control assessment. If any check fails, the control is marked as Fail.

Running a Compliance Scan

1. In the Alauda Container Security portal, go to **Compliance Dashboard**.
2. (Optional) To filter by specific standards:

  - Click **Manage standards**.
  - Deselect any standards you do not want to display.
  - Click **Save**.

3. Click **Scan environment**.

INFO

Scanning the entire environment typically takes about 2 minutes, depending on the number of clusters and nodes.

Viewing Compliance Results

Compliance Dashboard

The dashboard provides an overview of compliance standards across all clusters, namespaces, and nodes, including charts and options to investigate issues.

- To view compliance status for all clusters: Go to **Compliance Dashboard** and select the **Clusters** tab.
- To view a specific cluster: In the **Passing standards by cluster** widget, click a cluster name.
- To view all namespaces: Go to **Compliance Dashboard** and select the **Namespaces** tab.
- To view a specific namespace: In the **Namespaces** table, click a namespace to open its details.

By Standard

Alauda Container Security supports CIS compliance standards. To view controls for a specific standard:

1. Go to **Compliance Dashboard**.
2. In the **Passing standards across clusters** widget, click a standard to see all associated controls.

By Control

To view the compliance status for a specific control:

1. Go to **Compliance Dashboard**.
2. In the **Passing standards by cluster** widget, click a standard.
3. In the **Controls** table, click a control to view its details.

Filtering Compliance Data

You can filter compliance data by clusters, standards, or control status:

1. Go to **Compliance Dashboard**.
2. Select the **Clusters**, **Namespaces**, or **Nodes** tab.
3. Enter filtering criteria in the search bar and press **Enter**.

Generating Compliance Reports

Alauda Container Security allows you to generate:

- **Executive reports**: Business-focused, with charts and summaries (PDF format)
- **Evidence reports**: Technical, with detailed information (CSV format)

To export reports:

1. Go to **Compliance Dashboard**.
2. Click the **Export** tab:

  - Select **Download Page as PDF** for executive reports
  - Select **Download Evidence as CSV** for evidence reports

INFO

The **Export** option is available on all compliance pages and filtered views.

Evidence Report Fields

CSV Field	Description
Standard	The compliance standard, e.g., CIS Kubernetes
Cluster	The name of the assessed cluster
Namespace	The namespace or project where the deployment exists
Object Type	The Kubernetes entity type (e.g., node, cluster, DaemonSet, Deployment)
Object Name	The unique name of the object
Control	The control number as per the compliance standard
Control Description	Description of the compliance check
State	Whether the compliance check passed or failed
Evidence	Explanation for the compliance check result
Assessment Time	The time and date when the compliance scan was run

Menu

How to

Menu

Vulnerablitiy

Introduction

Introduction

Guides

Vulnerability Management Process

Overview

Key Steps in Vulnerability Management

Asset Assessment

Key Assets to Monitor

Vulnerability Scanning and Assessment

Prioritizing Vulnerabilities

Exposure Assessment

Taking Action

Viewing and Addressing Vulnerabilities

Overview of Vulnerability Management

Navigating Vulnerability Views

Exception Management

Identifying and Remediating Vulnerabilities

Exporting Vulnerability Data

Best Practices

Vulnerability Reporting

Planning Vulnerability Reports

Creating a Vulnerability Report

Configuring Delivery Destinations and Schedule

Reviewing and Creating the Report Configuration

Access Control and Permissions

Editing and Managing Report Configurations

Generating and Downloading Reports

Sending Reports Immediately

Report Retention and Expiry Settings

How to

Examining Images for Vulnerabilities

Scanner V4 Overview

Scanner Workflow

Supported Platforms and Formats

Image Scanning and Watch List

Vulnerability Data Updates

Generating SBOMs from Scanned Images

What is an SBOM?

How to Generate SBOMs

Image Scanning Using the roxctl CLI

Scanning an Image in a Remote Cluster

roxctl image scan Command Options

Introduction

Security vulnerabilities can be exploited by attackers to perform actions such as denial of service, remote code execution, or unauthorized access to sensitive data. Effective vulnerability management is essential for building a secure Kubernetes environment.

Menu

Guides

Vulnerability Management Process

Overview

Key Steps in Vulnerability Management

Asset Assessment

Key Assets to Monitor

Vulnerability Scanning and Assessment

Prioritizing Vulnerabilities

Exposure Assessment

Taking Action

Viewing and Addressing Vulnerabilities

Overview of Vulnerability Management

Navigating Vulnerability Views

Exception Management

Identifying and Remediating Vulnerabilities

Exporting Vulnerability Data

Best Practices

Vulnerability Reporting

Planning Vulnerability Reports

Creating a Vulnerability Report

Configuring Delivery Destinations and Schedule

Reviewing and Creating the Report Configuration

Access Control and Permissions

Editing and Managing Report Configurations

Generating and Downloading Reports

Sending Reports Immediately

Report Retention and Expiry Settings

Vulnerability Management Process

TOC

Overview
Key Steps in Vulnerability Management
Asset Assessment
Key Assets to Monitor
Vulnerability Scanning and Assessment
Prioritizing Vulnerabilities
Exposure Assessment
Taking Action
    Remediation Methods

Overview

Vulnerability management is a continuous process to identify and remediate vulnerabilities. Alauda Container Security helps you facilitate an effective vulnerability management process.

Key Steps in Vulnerability Management

A successful vulnerability management program typically includes the following key tasks:

- Asset assessment
- Vulnerability prioritization

- Exposure assessment
- Taking action
- Continuous reassessment

Alauda Container Security enables organizations to continuously assess their Alauda Container Platform and Kubernetes clusters, providing the contextual information needed to prioritize and address vulnerabilities more effectively.

Asset Assessment

To assess your organization's assets, follow these steps:

- Identify assets in your environment
- Scan these assets to detect known vulnerabilities
- Report vulnerabilities to relevant stakeholders

When you install Alauda Container Security on your Kubernetes or Alauda Container Platform cluster, it aggregates the assets running inside your cluster to help you identify them. Alauda Container Security allows organizations to perform ongoing assessments and provides the context required to prioritize and remediate vulnerabilities efficiently.

Key Assets to Monitor

Key assets to monitor in your vulnerability management process using Alauda Container Security include:

- **Components**: Software packages used as part of an image or running on a node. Components are the lowest level where vulnerabilities exist. Organizations must upgrade, modify, or remove software components to remediate vulnerabilities.
- **Images**: Collections of software components and code that create an environment to run executable code. Images are where you upgrade components to fix vulnerabilities.

- **Nodes**: Servers used to manage and run applications using Alauda Container Platform or Kubernetes, including the components that make up the platform or service.

Alauda Container Security organizes these assets into the following structures:

- **Deployment**: A definition of an application in Kubernetes that may run pods with containers based on one or more images.
- **Namespace**: A grouping of resources, such as Deployments, that support and isolate an application.
- **Cluster**: A group of nodes used to run applications using Alauda Container Platform or Kubernetes.

Vulnerability Scanning and Assessment

Alauda Container Security scans assets for known vulnerabilities and uses Common Vulnerabilities and Exposures (CVE) data to assess their impact.

Prioritizing Vulnerabilities

To prioritize vulnerabilities for action and investigation, consider the following questions:

- How important is the affected asset to your organization?
- How severe must a vulnerability be to warrant investigation?
- Can the vulnerability be fixed by patching the affected software component?
- Does the vulnerability violate any of your organization's security policies?

The answers to these questions help security and development teams determine the exposure and necessary response to a vulnerability.

Alauda Container Security provides tools to facilitate the prioritization of vulnerabilities in your applications and components. You can use data reported by Alauda Container Security to decide which vulnerabilities are critical to address. For example, when reviewing vulnerability findings by CVE, consider the following data provided by Alauda Container Security to sort and prioritize vulnerabilities:

- **CVE severity**: Number of images affected by the CVE and its severity rating (e.g., low, moderate, important, or critical).
- **Top CVSS**: The highest Common Vulnerability Scoring System (CVSS) score, from vendor sources, for this CVE across images.
- **Top NVD CVSS**: The highest CVSS score from the National Vulnerability Database for this CVE across images. Scanner V4 must be enabled to view this data.
- **EPSS probability**: The likelihood that the vulnerability will be exploited, according to the Exploit Prediction Scoring System (EPSS). This provides a percentage estimate of the probability that exploitation will be observed in the next 30 days. EPSS data should be used alongside other information, such as the age of the CVE, to help prioritize vulnerabilities.

Exposure Assessment

To assess your exposure to a vulnerability, ask:

- Is your application impacted by the vulnerability?
- Is the vulnerability mitigated by other factors?
- Are there known threats that could lead to exploitation?
- Are you using the vulnerable software package?
- Is it worthwhile to spend time addressing this specific vulnerability and package?

Taking Action

Based on your assessment, you may take the following actions:

- Mark the vulnerability as a false positive if there is no exposure or it does not apply in your environment.

- Decide whether to remediate, mitigate, or accept the risk if you are exposed.
- Remove or change the software package to reduce your attack surface.

Once you decide to act on a vulnerability, you can:

- Remediate the vulnerability
- Mitigate and accept the risk
- Accept the risk
- Mark the vulnerability as a false positive

Remediation Methods

To remediate vulnerabilities, you can:

- Remove a software package
- Update a software package to a non-vulnerable version

Viewing and Addressing Vulnerabilities

Alauda Container Security provides comprehensive tools for discovering, viewing, prioritizing, and addressing vulnerabilities in your container and cluster environments. This document describes how to use the platform to manage vulnerabilities efficiently and securely.

TOC

- Overview of Vulnerability Management
- Navigating Vulnerability Views
  - User Workload Vulnerabilities
    - How to View User Workload Vulnerabilities
    - User Workload Filter Options
  - Platform Vulnerabilities
    - How to View Platform Vulnerabilities
    - Platform Filter Options
  - Node Vulnerabilities
    - How to View Node Vulnerabilities
    - Node Filter Options
  - More Views
    - How to Use More Views
- Exception Management
  - Snoozing CVEs
    - Steps to Snooze/Unsnooze CVEs
  - Marking CVEs as False Positives
    - Steps to Mark as False Positive
  - Deferring CVEs
    - Steps to Defer CVEs

- Managing Exception Requests
- Viewing Deferred and False Positive CVEs

Identifying and Remediating Vulnerabilities

- Identifying Vulnerable Dockerfile Lines
  - Steps
- Upgrading Components
  - Steps

Exporting Vulnerability Data

- How to Export via API
  - Example

Best Practices

Overview of Vulnerability Management

Alauda Container Security enables you to:

- Identify vulnerabilities in workloads, platform components, and nodes
- Filter and prioritize vulnerabilities based on risk
- Take action through remediation, deferral, or exception management
- Export vulnerability data for further analysis

Navigating Vulnerability Views

Vulnerability data is organized into several main views, accessible from **Vulnerability Management > Results**:

- **User workloads**: Vulnerabilities in workloads and images you have deployed
- **Platform**: Vulnerabilities in platform components (e.g., Alauda Container Platform and layered services)
- **Nodes**: Vulnerabilities across all nodes

- **More views**: Additional perspectives, such as all vulnerable images, inactive images, images without CVEs, and Kubernetes components

User Workload Vulnerabilities

View and filter vulnerabilities in your deployed workloads and images.

How to View User Workload Vulnerabilities

1. Go to **Vulnerability Management > Results**.
2. Select the **User Workloads** tab.
3. Use the **Observed**, **Deferred**, or **False positives** tabs to filter by vulnerability status.
4. Refine results by namespace, severity, or other filters as needed.
5. Use the filter bar to search by entity (e.g., CVE, image, deployment).

> **Note** The **Filtered view** icon indicates that results are filtered. Click **Clear filters** to remove all filters, or remove individual filters by clicking them.

User Workload Filter Options

Entity	Attributes
Image	**Name**; **Operating system**; **Tag**; **Label**; **Registry**
CVE	**Name**; **Discovered time**; **CVSS**; **EPSS probability**
Image Component	**Name**; **Source** (OS, Python, Java, Ruby, Node.js, Go, Dotnet Core Runtime, Infrastructure); **Version**
Deployment	**Name**; **Label**; **Annotation**; **Status**
Namespace	**ID**; **Name**; **Label**; **Annotation**
Cluster	**ID**; **Name**; **Label**; **Type**; **Platform type**
	CVE severity; **CVE status**

Platform Vulnerabilities

View vulnerabilities in platform components and layered services.

How to View Platform Vulnerabilities

1. Go to **Vulnerability Management > Results**.
2. Select the **Platform** tab.
3. Use the **Observed**, **Deferred**, or **False positives** tabs as needed.
4. Refine results by namespace, severity, or other filters.
5. Use the filter bar to search by entity.

Platform Filter Options

Entity	Attributes
Image	**Name**; **Operating system**; **Tag**; **Label**; **Registry**
CVE	**Name**; **Discovered time**; **CVSS**; **EPSS probability**
Image Component	**Name**; **Source**; **Version**
Deployment	**Name**; **Label**; **Annotation**; **Status**
Namespace	**ID**; **Name**; **Label**; **Annotation**
Cluster	**ID**; **Name**; **Label**; **Type**; **Platform type**
	CVE severity; **CVE status**

Node Vulnerabilities

View vulnerabilities across all nodes in your environment.

How to View Node Vulnerabilities

1. Go to **Vulnerability Management > Results**.

2. Select the **Nodes** tab.
3. Optionally, click **Show snoozed CVEs**.
4. Use filters to narrow down by node, CVE, component, or cluster.

Node Filter Options

Entity	Attributes
Node	**Name**; **Operating system**; **Label**; **Annotation**; **Scan time**
CVE	**Name**; **Discovered time**; **CVSS**
Node Component	**Name**; **Version**
Cluster	**ID**; **Name**; **Label**; **Type**; **Platform type**

More Views

Access additional perspectives on vulnerabilities:

- **All vulnerable images**: See all images with vulnerabilities
- **Inactive images**: View vulnerabilities in watched or inactive images
- **Images without CVEs**: Identify images with no detected vulnerabilities
- **Kubernetes components**: View vulnerabilities in the underlying Kubernetes structure

How to Use More Views

1. Go to **Vulnerability Management > Results**.
2. Click **More Views** and select the desired view.
3. Use available filters and columns to organize and analyze the data.

Exception Management

Exception management allows you to snooze, defer, or mark CVEs as false positives, tailoring vulnerability management to your organization's needs.

Snoozing CVEs

Temporarily ignore a CVE for a specified period. Snoozed CVEs do not appear in reports or trigger policy violations.

Steps to Snooze/Unsnooze CVEs

1. Go to **Vulnerability Management > Platform CVEs** or **Node CVEs**.
2. Select CVEs and use the overflow menu or bulk actions to snooze or unsnooze.
3. Choose the duration and confirm.

Marking CVEs as False Positives

Mark a CVE as a false positive globally or for specific images. Requires approval.

Steps to Mark as False Positive

1. Go to **Vulnerability Management > Results** > **User Workloads**.
2. Select CVEs and use the overflow menu or bulk actions.
3. Enter a rationale and submit the request.

Deferring CVEs

Defer a CVE, accepting the risk for a specified period. Requires approval.

Steps to Defer CVEs

1. Go to **Vulnerability Management > Results** > **User Workloads**.
2. Select CVEs and use the overflow menu or bulk actions.
3. Choose the deferral period, enter a rationale, and submit.

Managing Exception Requests

Review, approve, deny, update, or cancel exception requests in **Vulnerability Management > Exception Management**.

Viewing Deferred and False Positive CVEs

In **User Workloads**, use the **Deferred** or **False positives** tabs to view relevant CVEs.

Identifying and Remediating Vulnerabilities

Identifying Vulnerable Dockerfile Lines

Alauda Container Security can show which Dockerfile line introduced a vulnerable component.

Steps

1. Go to **Vulnerability Management > Results** > **User Workloads**.
2. Click a CVE to view details and expand to see the affected Dockerfile line.

Upgrading Components

Find and upgrade to a fixed version of a vulnerable component.

Steps

1. Go to **Vulnerability Management > Results** > **User Workloads** > **Images**.
2. Select an image and expand the CVE to see the fixed version.
3. Update your image accordingly.

Exporting Vulnerability Data

Export vulnerability data for further analysis or reporting using the API.

How to Export via API

- Use the `/v1/export/vuln-mgmt/workloads` streaming API.
- Output is JSON, each line contains a deployment and its images.

Example

```
curl -H "Authorization: Bearer $ROX_API_TOKEN" $ROX_ENDPOINT/v1/export/vuln-m
```

Best Practices

- Use filters and exception management to focus on relevant vulnerabilities.
- Regularly review deferred and false positive CVEs.
- Integrate exported data with external tools for compliance and reporting.
- Keep Alauda Container Security and scanners up to date.

Summary

Alauda Container Security provides a robust platform for vulnerability discovery, prioritization, remediation, exception management, and data export. By following the structured procedures and best practices in this document, you can effectively manage container and cluster security risks in your environment.

Vulnerability Reporting

Alauda Container Security allows you to create, schedule, and download on-demand image vulnerability reports from the **Vulnerability Management > Vulnerability Reporting** menu. These reports provide a comprehensive list of vulnerabilities (CVEs) in images and deployments (user workloads).

You can share these reports with auditors or internal stakeholders by scheduling email delivery or downloading and distributing the report manually. Scheduled communications help keep key stakeholders informed about the vulnerability status of your environment.

TOC

Planning Vulnerability Reports

Creating a Vulnerability Report

  Steps

Configuring Delivery Destinations and Schedule

Reviewing and Creating the Report Configuration

Access Control and Permissions

Editing and Managing Report Configurations

  Editing a Report Configuration

  Cloning a Report Configuration

  Deleting a Report Configuration

Generating and Downloading Reports

  Steps

Sending Reports Immediately

Report Retention and Expiry Settings

Planning Vulnerability Reports

When planning scheduled vulnerability reports, consider:

- What schedule is most effective for your stakeholders?
- Who is the audience?
- Should the report include only specific severity levels?
- Should the report include only fixable vulnerabilities?

Alauda Container Security guides you through creating a vulnerability report configuration, which determines the content and schedule of each report.

Creating a Vulnerability Report

Steps

1. In the Alauda Container Security portal, go to **Vulnerability Management > Vulnerability Reporting**.
2. Click **Create report**.
3. On the **Configure report parameters** page, provide:

  - **Report name**: Name for your report configuration.
  - **Report description**: (Optional) Description of the report.
  - **CVE severity**: Select the severity levels to include.
  - **CVE status**: Select one or more statuses (**Fixable**, **Unfixable**).
  - **Image type**: Select one or more types (**Deployed images**, **Watched images**).
  - **CVEs discovered since**: Select the time period for included CVEs.
  - (Optional) **Include NVD CVSS**: Add the NVD CVSS column to the report.
  - **Configure collection included**: Select or create at least one collection to include. You can view, edit, or preview collections.

> **Note** For more about collections, see "Creating and using deployment collections".

1. Click **Next** to configure delivery destinations and schedule (optional unless you selected to include CVEs discovered since the last scheduled report).

Configuring Delivery Destinations and Schedule

1. In **Configure delivery destinations**, add a destination and set up a schedule.
2. To email reports, configure at least one email notifier. Select an existing notifier or create a new one. Default recipients appear in the **Distribution list**; you can add more addresses separated by commas.
3. Edit the default email template if needed:

    1.1. Click the edit icon and customize the subject and body in the **Edit** tab.

    1.2. Preview your template in the **Preview** tab.

    1.3. Click **Apply** to save changes.

    > **Note** When reviewing report jobs, you can see whether the default or a custom template was used.

4. In **Configure schedule**, select the frequency and day of the week.
5. Click **Next** to review and finish creating the report configuration.

Reviewing and Creating the Report Configuration

1. In the **Review and create** section, review all configuration parameters, delivery destination, email template, schedule, and format. Click **Back** to edit any field.
2. Click **Create** to save the configuration.

Access Control and Permissions

- You can only view, create, and download reports for data your user account has permission to access.
- You can only download reports you have generated; you cannot download reports generated by others.
- If your access permissions change, old reports do not reflect the new permissions. To view new data, create a new report.

Editing and Managing Report Configurations

You can edit, clone, or delete report configurations as needed.

Editing a Report Configuration

1. In **Vulnerability Management > Vulnerability Reporting**, locate the report configuration.
2. Click the overflow menu (three dots) and select **Edit report**, or click the report name, then **Actions > Edit report**.
3. Make changes and save.

Cloning a Report Configuration

1. In the list, click **Clone report** for the desired configuration.
2. Modify parameters and destinations as needed.
3. Click **Create**.

Deleting a Report Configuration

1. In the list, click the overflow menu for the configuration and select **Delete report**.

> **Note** Deleting a configuration also deletes all reports previously run using it.

Generating and Downloading Reports

You can generate and download on-demand vulnerability reports.

Steps

1. In **Vulnerability Management > Vulnerability Reporting**, locate the desired configuration.
2. Generate the report:

- From the list: Click the overflow menu and select **Generate download**. The status appears in **My active job status**. When processing is complete, the report is ready for download.
- From the report window: Click the report name, then **Actions > Generate download**.

3. To download, open the report configuration, click **All report jobs**, and click the **Ready for download** link in the **Status** column. The report is a `.csv` file compressed as `.zip`.

> **Note** You can only download reports you have generated.

Sending Reports Immediately

You can send a report immediately instead of waiting for the scheduled time.

1. In **Vulnerability Management > Vulnerability Reporting**, locate the configuration.
2. Click the overflow menu and select **Send report now**.

Report Retention and Expiry Settings

You can configure how long report jobs and downloadable files are retained.

1. In **Platform Configuration > System Configuration**, set:

- **Vulnerability report run history retention**: Number of days to keep report job records.
- **Prepared downloadable vulnerability reports retention days**: Number of days downloadable reports are available.
- **Prepared downloadable vulnerability reports limit**: Maximum space (MB) for downloadable reports; oldest jobs are removed when the limit is reached.

2. Click **Edit** to change values, then **Save**.

> **Note** These settings do not affect jobs in `WAITING` or `PREPARING` state, the last successful scheduled/on-demand/emailed/downloaded job, or jobs not yet deleted manually or by pruning.

Menu

How to

Examining Images for Vulnerabilities

Scanner V4 Overview

Scanner Workflow

Supported Platforms and Formats

Image Scanning and Watch List

Vulnerability Data Updates

Generating SBOMs from Scanned Images

What is an SBOM?

How to Generate SBOMs

Image Scanning Using the roxctl CLI

Scanning an Image in a Remote Cluster

roxctl image scan Command Options

Examining Images for Vulnerabilities

Alauda Container Security for Kubernetes enables you to analyze container images for vulnerabilities using the built-in Scanner V4. The scanner inspects image layers, identifies packages, and matches them against vulnerability databases from sources like NVD, OSV, and OS-specific feeds.

When vulnerabilities are detected, Alauda Container Security:

- Displays them in the **Vulnerability Management** view
- Ranks and highlights them for risk assessment
- Checks them against enabled security policies

The scanner identifies installed components by inspecting specific files. If these files are missing, some vulnerabilities may not be detected. Required files include:

Component Type	Required Files
Package managers	`/etc/alpine-release`; `/etc/lsb-release`; `/etc/os-release` or `/usr/lib/os-release`; `/etc/oracle-release`; `/etc/centos-release`; `/etc/redhat-release`; `/etc/system-release`; other similar files
Language-level dependencies	`package.json` (JavaScript); `dist-info` / `egg-info` (Python); `MANIFEST.MF` (Java JAR)
Application-level dependencies	`dotnet/shared/Microsoft.AspNetCore.App/`; `dotnet/shared/Microsoft.NETCore.App/`

TOC

Scanner V4 Overview
Scanner Workflow
  Workflow Steps
  Common Scanner Warning Messages
Supported Platforms and Formats
  Supported Linux Distributions
  Supported Package Formats
  Supported Programming Languages
  Supported Container Image Layer Formats
Image Scanning and Watch List
Vulnerability Data Updates

Scanner V4 Overview

Scanner V4 enhances scanning for language and OS-specific components. Scanner V4 is enabled by default and is required for all vulnerability scanning scenarios.

Scanner Workflow

Workflow Steps

1. Central requests Scanner V4 Indexer to analyze images.
2. Indexer pulls metadata and downloads layers.
3. Indexer produces an index report.
4. Matcher matches images to vulnerabilities and generates reports.

Common Scanner Warning Messages

Message	Description
Unable to retrieve the OS CVE data, only Language CVE data is available	Base OS not supported; no OS-level CVEs.
Stale OS CVE data	OS is end-of-life; data may be outdated.
Failed to get the base OS information	Scanner could not determine the base OS.
Failed to retrieve metadata from the registry	Registry unreachable or authentication failed.
Image out of scope for Red Hat Vulnerability Scanner Certification	Image is too old for certification.

Supported Platforms and Formats

Supported Linux Distributions

Distribution	Version
Alpine Linux	`alpine:3.2` – `alpine:3.21` , `alpine:edge`
Amazon Linux	`amzn:2018.03` , `amzn:2` , `amzn:2023`
CentOS	`centos:6` , `centos:7` , `centos:8`
Debian	`debian:11` , `debian:12` , `debian:unstable` , Distroless ↗
Oracle Linux	`ol:5` – `ol:9`
Photon OS	`photon:1.0` – `photon:3.0`
RHEL	`rhel:6` – `rhel:9`
SUSE	`sles:11` – `sles:15` , `opensuse-leap:15.5` , `opensuse-leap:15.6`

Distribution	Version
Ubuntu	`ubuntu:14.04` – `ubuntu:24.10`

INFO

Some older Debian/Ubuntu versions are not updated by the vendor. Fedora is not supported for OS CVEs.

Supported Package Formats

Package Format	Package Managers
apk	apk
dpkg	apt; dpkg
rpm	dnf; microdnf; rpm; yum

Supported Programming Languages

Language	Package Format
Go	Binaries (analyzes stdlib and, if present, `go.mod` dependencies)
Java	JAR; WAR; EAR; JPI; HPI
JavaScript	package.json
Python	egg; wheel
Ruby	gem

Supported Container Image Layer Formats

Format	Scanner V4
No compression	Yes

Format	Scanner V4
bzip2	Yes
gzip	Yes
xz	No
zstd	Yes

Image Scanning and Watch List

Alauda Container Security scans all active images every 4 hours. You can also enable automatic scanning of inactive images (from version 3.0.57) via the **Watch** setting.

Steps:

1. In the portal, go to **Vulnerability Management > Results**.
2. Click **More Views > Inactive images**.
3. Click **Manage watched images** and add or remove images as needed.

INFO

Data for removed images is retained for the configured period in **System Configuration**.

Vulnerability Data Updates

Central fetches vulnerability definitions every 5 minutes from `https://definitions.stackrox.io`

Menu ON THIS PAGE >

Generating SBOMs from Scanned Images

Alauda Container Security enables you to generate a Software Bill of Materials (SBOM) from scanned container images. This feature provides a detailed overview of software components, dependencies, and libraries within your application, helping organizations locate vulnerable packages and comply with security requirements.

TOC

What is an SBOM?

How to Generate SBOMs

  Using the Portal

  Using the `roxctl` CLI

    CLI Options

What is an SBOM?

A Software Bill of Materials (SBOM) is a digital record listing the components of a piece of software and their origins. SBOMs help organizations:

- Identify the presence of vulnerable packages and components
- Respond quickly to mitigate risks
- Comply with regulations such as Executive Order 14028

SBOMs can be generated in different ways. The SBOMs generated by Alauda Container Security are "Analyzed" SBOMs, created by analyzing artifacts such as executables, packages, containers, and VM images. According to CISA, analyzed SBOMs:

- Provide information without requiring an active development environment
- Can be generated without access to the build process
- Help discover hidden dependencies

The SBOM generated by Alauda Container Security is in System Package Data Exchange (SPDX) 2.3 ↗ format.

How to Generate SBOMs

You can generate SBOMs using the Alauda Container Security portal, the `roxctl` CLI, or the API.

Using the Portal

1. Go to **Vulnerability Management > Results** and locate the image you want.
2. Do one of the following:

   - In the image row, click the overflow menu and select **Generate SBOM**.
   - Select the image to view details, then click **Generate SBOM**.

3. A window will display information about the image and the SBOM format. Click **Generate SBOM** to create the file in JSON format. The file will be downloaded automatically depending on your browser settings.

Using the `roxctl` CLI

Run the following command:

```
roxctl image sbom --image=image-name
```

Replace `image-name` with the name and reference of the image (e.g., `nginx:latest` or `nginx@sha256:...`).

CLI Options

Option	Description
`-f, --force`	Bypass Central's cache for the image and force a new pull from the scanner. Default: `false`.
`-d, --retry-delay integer`	Time to wait between retries in seconds. Default: 3.
`-i, --image string`	Image name and reference (e.g., `nginx:latest` or `nginx@sha256:...`).
`-r, --retries integer`	Number of times Scanner V4 should retry before exiting with an error. Default: 3.

Image Scanning Using the roxctl CLI

You can scan images stored in image registries, including cluster local registries such as the Alauda Container Platform integrated image registry, by using the `roxctl` CLI.

INFO

Image scanning requires appropriate permissions and network access to the registry and Central.

TOC

Scanning an Image in a Remote Cluster
Example Output
roxctl image scan Command Options
Option Descriptions

Scanning an Image in a Remote Cluster

Run the following command to scan the specified image

```
roxctl image scan \
  --image=<image_registry>/<image_name>
```

- For `<image_registry>`, specify the registry where the image is located, e.g., `image-registry.alauda-image-registry.svc:5000/`.

Example Output

```
{
  "Id": "sha256:3f439d7d71adb0a0c8e05257c091236ab00c6343bc44388d091450ff58664
  "name": {
    "registry": "image-registry.alauda-image-registry.svc:5000",
    "remote": "default/image-stream",
    "tag": "latest",
    "fullName": "image-registry.alauda-image-registry.svc:5000/default/image-
  }
  // ...
}
```

- `Id`: A unique identifier for the image, serving as a fingerprint for integrity and authenticity.
- `name.registry`: The image registry location.
- `name.remote`: The remote path to the image.
- `name.tag`: The version or tag of the image.
- `name.fullName`: The complete name of the image (registry, path, tag).

roxctl image scan Command Options

Option Descriptions

Option	Description
`--cluster string`	Delegate image scanning to a specific cluster.
`--compact-output`	Print the JSON output in a compact format. Default: `false`.
`-f, --force`	Ignore Central's cache for the scan and force a fresh re-pull from Scanner. Default: `false`.

Option	Description
`--headers strings`	Print the headers in a tabular format. Default: `COMPONENT`, `VERSION`, `CVE`, `SEVERITY`, `LINK`.
`--headers-as-comments`	Print the headers as comments in a CSV tabular output. Default: `false`.
`-h, --help`	View the help text for the `roxctl image scan` command.
`-i, --image string`	Specify the image name and reference you want to scan.
`-a, --include-snoozed`	Return both snoozed and unsnoozed CVEs. Default: `false`.
`--merge-output`	Merge duplicate cells in a tabular output. Default: `true`.
`--no-header`	Do not print headers for tabular format. Default: `false`.
`-o, --output string`	Specify the output format: `table`, `CSV`, `JSON`, or `SARIF`.
`-r, --retries int`	Set the number of retries before aborting with an error. Default: `3`.
`-d, --retry-delay int`	Set the time in seconds to wait between retries. Default: `3`.
`--row-jsonpath-expressions string`	Use JSON path expressions to create rows from the JSON object. See `roxctl image scan --help` for details.

Risk

Introduction

Introduction

Guides

Evaluating Security Risks

Risk View

Risk Details Panel

Using Process Baseline

What is a Process Baseline?

Baseline States

Managing Process Baselines

Introduction

Alauda Container Security for Kubernetes is a platform designed to help you identify, assess, and manage security risks across your containerized environments. It provides visibility into vulnerabilities, misconfigurations, and risky runtime activities, enabling you to prioritize and address the most critical security issues in your deployments.

Menu

Guides

Evaluating Security Risks

Risk View

Risk Details Panel

Using Process Baseline

What is a Process Baseline?

Baseline States

Managing Process Baselines

Evaluating Security Risks

Alauda Container Security assesses and ranks your deployments by security risk, highlighting vulnerabilities, configurations, and runtime activities needing attention.

TOC

- Risk View
  - Creating Policies from Risk View
    - Filter Mapping Table
- Risk Details Panel
  - Risk Indicators Tab
  - Deployment Details Tab
  - Process Discovery Tab
  - Event Timeline

Risk View

The **Risk** view lists all deployments, sorted by a multi-factor risk metric (policy violations, image contents, configuration, etc.). Deployments at the top are the most at risk.

Each deployment shows:

- **Name**
- **Created**
- **Cluster**
- **Namespace**

- **Priority**

Features:

- Sort and filter violations
- Create new policies from filtered results

To see more details, select a deployment.

Creating Policies from Risk View

You can create security policies based on your filters in the **Risk** view.

Steps:

1. Go to **Risk** in the portal.
2. Apply filters.
3. Click **New Policy** and fill required fields.

Note: Only Cluster, Namespace, Deployment, and Label filters are converted to policy scopes. Other filters may be dropped or modified.

Filter Mapping Table

Search Attribute	Policy Criteria
Add Capabilities	Add Capabilities
Annotation	Disallowed Annotation
CPU Cores Limit	Container CPU Limit
CPU Cores Request	Container CPU Request
CVE	CVE
CVE Published On	✕ Dropped
CVE Snoozed	✕ Dropped
CVSS	CVSS

Search Attribute	Policy Criteria
Cluster	⟳ Converted to scope
Component	Image Component (name)
Component Version	Image Component (version)
Deployment	⟳ Converted to scope
Deployment Type	✕ Dropped
Dockerfile Instruction Keyword	Dockerfile Line (key)
Dockerfile Instruction Value	Dockerfile Line (value)
Drop Capabilities	✕ Dropped
Environment Key	Environment Variable (key)
Environment Value	Environment Variable (value)
Environment Variable Source	Environment Variable (source)
Exposed Node Port	✕ Dropped
Exposing Service	✕ Dropped
Exposing Service Port	✕ Dropped
Exposure Level	Port Exposure
External Hostname	✕ Dropped
External IP	✕ Dropped
Image	✕ Dropped
Image Command	✕ Dropped
Image Created Time	Days since image was created
Image Entrypoint	✕ Dropped
Image Label	Disallowed Image Label

Search Attribute	Policy Criteria
Image OS	Image OS
Image Pull Secret	✕ Dropped
Image Registry	Image Registry
Image Remote	Image Remote
Image Scan Time	Days since image was last scanned
Image Tag	Image Tag
Image Top CVSS	✕ Dropped
Image User	✕ Dropped
Image Volumes	✕ Dropped
Label	⟳ Converted to scope
Max Exposure Level	✕ Dropped
Memory Limit (MB)	Container Memory Limit
Memory Request (MB)	Container Memory Request
Namespace	⟳ Converted to scope
Namespace ID	✕ Dropped
Pod Label	✕ Dropped
Port	Port
Port Protocol	Protocol
Priority	✕ Dropped
Privileged	Privileged
Process Ancestor	Process Ancestor
Process Arguments	Process Arguments

Search Attribute	Policy Criteria
Process Name	Process Name
Process Path	✕ Dropped
Process Tag	✕ Dropped
Process UID	Process UID
Read Only Root Filesystem	Read-Only Root Filesystem
Secret	✕ Dropped
Secret Path	✕ Dropped
Service Account	✕ Dropped
Service Account Permission Level	Minimum RBAC Permission Level
Toleration Key	✕ Dropped
Toleration Value	✕ Dropped
Volume Destination	Volume Destination
Volume Name	Volume Name
Volume ReadOnly	Writable Volume
Volume Source	Volume Source
Volume Type	Volume Type

Scope Conversion Example: Filtering by `Cluster:A,B` and `Namespace:Z` creates:

- (Cluster=A AND Namespace=Z)
- (Cluster=B AND Namespace=Z)

Risk Details Panel

Selecting a deployment opens the **Risk Details** panel with multiple tabs.

Risk Indicators Tab

Shows:

- **Policy Violations**
- **Suspicious Process Executions**
- **Image Vulnerabilities**
- **Service Configurations**
- **Service Reachability**
- **Components Useful for Attackers**
- **Number of Components in Image**
- **Image Freshness**
- **RBAC Configuration**

Only relevant sections are shown for the selected deployment.

Deployment Details Tab

Provides:

- Deployment ID
- Namespace
- Updated (timestamp)
- Deployment Type
- Replicas
- Labels
- Cluster name
- Annotations
- Service Account

Container Configuration:

- Image Name
- Resources: CPU/Memory requests and limits
- Mounts: Name, Source, Destination, Type
- Secrets: Kubernetes secrets and X.509 certificate details

Security Context:

- Privileged: `true` if privileged

Process Discovery Tab

Lists all binaries executed in each container, summarized by deployment:

- **Binary Name**
- **Container**
- **Arguments**
- **Time** (most recent)
- **Pod ID**
- **UID**

Use `Process Name:<name>` in the filter bar to search.

Event Timeline

The **Event Timeline** shows events for the selected deployment:

- Process activities
- Policy violations
- Container restarts/terminations

Events appear as icons on a timeline. Hover for details. You can:

- Show legend for event types
- Export as PDF/CSV
- Filter event types
- Expand to see events per container

A minimap controls the visible range.

Notes:

- On container restarts, up to 10 inactive instances per container are shown; process activities for previous instances are not tracked.
- Only the most recent execution of each (process name, arguments, UID) per pod is shown.
- Events are shown only for active pods.
- Timestamps are adjusted for accuracy.

Using Process Baseline

Process baselining in Alauda Container Security helps secure your infrastructure by learning which processes normally run in your containers and enforcing that only these are allowed.

TOC

- What is a Process Baseline?
- Baseline States
  - Unlocked
  - Locked
- Managing Process Baselines
  - Viewing Baselines
  - Adding a Process
  - Removing a Process
  - Locking/Unlocking the Baseline

What is a Process Baseline?

When you deploy Alauda Container Security, there is no default process baseline. As deployments are discovered, a process baseline is automatically created for each container type, including all observed processes.

Baseline States

Unlocked

- During initial discovery (first hour), baselines are unlocked.
- New processes are automatically added to the baseline and do not trigger risks or violations.
- After one hour, new processes are marked as risks but do not trigger violations, and are not added to the baseline.

Locked

- Locking a baseline stops new processes from being added.
- Any process not in the baseline triggers a violation.
- You can always manually add or remove processes from the baseline.

If a deployment has multiple container types, each has its own baseline. If some are locked and others unlocked, the deployment status shows as **Mixed**.

Managing Process Baselines

You can view and manage process baselines in the **Risk** view of the Alauda Container Security portal.

Viewing Baselines

1. Go to **Risk** in the portal.
2. Select a deployment.
3. In the details panel, open the **Process Discovery** tab.
4. Baselines are listed under **Spec Container Baselines**.

Adding a Process

1. In **Process Discovery**, under **Running Processes**, click the **Add** icon next to a process not already in the baseline.

Removing a Process

1. In **Process Discovery**, under **Spec Container Baselines**, click the **Remove** icon next to the process you want to remove.

Locking/Unlocking the Baseline

- Click the **Lock** icon to enforce violations for unlisted processes.
- Click the **Unlock** icon to stop enforcing violations.

By managing process baselines, you ensure only approved processes run in your environment, reducing security risks.

Menu

Security Policy

Introduction

Introduction

Guides

View Security Policy

Learn how to view and manage security policies in Alauda Container Security.

Policy Categories

Policy Lifecycle Stages

Policy Criteria and Attributes

Policy Enforcement

Exporting and Importing Policies

Create Custom Policy

Learn how to create custom policies in Alauda Container Security.

Methods to Create Custom Policies

Creating Policies via the Portal

Editing and Managing Policies

Default Policies in Alauda Container Security

Overview of default and custom policies in Alauda Container Security.

Overview

Policy Table Structure

Critical Severity Policies

High Severity Policies

Medium Severity Policies

Low Severity Policies

Managing Default Policies

How To

Checking Policy Compliance with roxctl

Learn how to check policy compliance using roxctl in Alauda Container Security.

Prerequisites

Output Formats

Output Options

Checking Policy Compliance for Deployments

Checking Policy Compliance for Images

Viewing Image Scan Results

Use Policy to Verify Image Signature in Alauda Container Security

Learn how to use policies to verify image signatures in Alauda Container Security.

Supported Signature Verification Methods

Prerequisites

Configure Signature Integration

Create and Enforce Image Signature Verification Policies

Introduction

Alauda Container Security is a security solution designed for containerized environments. It helps prevent high-risk service deployments and enables timely response to runtime security incidents, ensuring the safety and compliance of your container infrastructure.

Guides

View Security Policy

Learn how to view and manage security policies in Alauda Container Security.

Policy Categories

Policy Lifecycle Stages

Policy Criteria and Attributes

Policy Enforcement

Exporting and Importing Policies

Create Custom Policy

Learn how to create custom policies in Alauda Container Security.

Methods to Create Custom Policies

Creating Policies via the Portal

Editing and Managing Policies

Default Policies in Alauda Container Security

Overview of default and custom policies in Alauda Container Security.

Overview

Policy Table Structure

Critical Severity Policies

High Severity Policies

Medium Severity Policies

Low Severity Policies

Managing Default Policies

Viewing and Managing Security Policies

Alauda Container Security offers both default and customizable security policies to help you prevent high-risk deployments and respond to runtime incidents in your container environment.

TOC

- Policy Categories
- Policy Lifecycle Stages
- Policy Criteria and Attributes
  - Image Registry and Contents
  - Container Configuration
  - Deployment Metadata
  - Storage and Networking
  - Process Activity (Runtime Only)
  - Kubernetes Access and Events
- Policy Enforcement
- Exporting and Importing Policies
  - Exporting a Policy
  - Importing a Policy

Policy Categories

Policies are organized by type and function for easier management and search. Default categories include:

- Anomalous Activity
- Cryptocurrency Mining
- DevOps Best Practices
- Docker CIS
- Kubernetes
- Kubernetes Events
- Network Tools
- Package Management
- Privileges
- Security Best Practices
- Supply Chain Security
- System Modification
- Vulnerability Management
- Zero Trust

To manage categories:

1. Go to **Platform Configuration** > **Policy Management**.
2. Click the **Policy Categories** tab.
3. Create, view, or manage categories as needed.

Policy Lifecycle Stages

When creating or editing a policy, you can specify one or more lifecycle stages:

- **Build**: Checks image fields (e.g. CVEs, Dockerfile instructions).
- **Deploy**: Includes build-time checks and cluster configuration (e.g. privileged mode).

- **Runtime**: Adds process execution and runtime event checks.

Policy Criteria and Attributes

Policies are triggered by specific criteria (attributes). The following tables summarize common attributes and their descriptions. For details on allowed values, operators, and applicable phases, see the notes below each table.

Image Registry and Contents

Attribute	Description	Allowed Values
Image Registry	Name of the image registry	String
Image Name	Full image name in registry	String
Image Tag	Image identifier	String
Image Signature	Signature integration for image	Integration ID
Fixable	Image has fixable CVE	Boolean
Days Since CVE	Days since CVE discovered	Integer
Image Age	Days since image creation	Integer
Image Scan Age	Days since last image scan	Integer
Image User	USER directive in Dockerfile	String
Dockerfile Line	Dockerfile instruction/argument	LABEL/RUN/etc.
Unscanned Image	Image scan status	Boolean
CVSS	Vulnerability score	Number
Severity	Vulnerability severity	Level

Attribute	Description	Allowed Values
Fixed By	Version that fixes vulnerability	String
CVE	Specific CVE number	String
Image Component	Software component in image	key=value
Image OS	Base OS of the image	String
Required Image Label	Required Docker image label	key=value
Disallowed Image Label	Disallowed Docker image label	key=value

Operators: Regex, NOT, AND, OR, OR only, AND only, None, etc.

Phases: Build, Deploy, Runtime

Container Configuration

Attribute	Description	Allowed Values
Environment Variable	Check environment variables	RAW=key=value
Container CPU Request	CPU cores requested	Number
Container CPU Limit	CPU cores limit	Number
Container Memory Request	Memory requested (MB)	Number
Container Memory Limit	Memory limit (MB)	Number

Attribute	Description	Allowed Values
Privileged Container	Privileged mode enabled	Boolean
Read-Only Root Filesystem	Root filesystem is read-only	Boolean
Seccomp Profile Type	Seccomp profile type	UNCONFINED/RUNTIME_DEFAULT/LOCALHOST
Allow Privilege Escalation	Privilege escalation allowed	Boolean
Drop Capabilities	Linux capabilities to drop	List
Add Capabilities	Linux capabilities not allowed	List
Container Name	Name of the container	String
AppArmor Profile	AppArmor profile used	String
Liveness Probe	Liveness probe defined	Boolean
Readiness Probe	Readiness probe defined	Boolean

Operators: Regex, AND, OR, None, etc.

Phases: Deploy, Runtime

Deployment Metadata

Attribute	Description	Allowed Values
Disallowed Annotation	Annotation not allowed	key=value
Required Label	Required Kubernetes label	key=value
Required Annotation	Required Kubernetes annotation	key=value
Runtime Class	RuntimeClass of the deployment	String
Host Network	Host network enabled	Boolean
Host PID	Host PID namespace shared	Boolean
Host IPC	Host IPC namespace shared	Boolean
Namespace	Namespace of the deployment	String
Replicas	Number of deployment replicas	Number

Operators: Regex, AND, OR, NOT, None, etc.

Phases: Deploy, Runtime

Storage and Networking

Attribute	Description	Allowed Values
Volume Name	Name of the storage	String
Volume Source	Volume provision type	String
Volume Destination	Path where volume is mounted	String

Attribute	Description	Allowed Values
Volume Type	Type of volume	String
Writable Mounted Volume	Volume mounted as writable	Boolean
Mount Propagation	Mount propagation mode	NONE/HOSTTOCONTAINER/BIDIRECTIONAL
Writable Host Mount	Host path mounted writable	Boolean
Exposed Port Protocol	Protocol used by exposed port	String
Exposed Port	Port numbers exposed	Number
Exposed Node Port	Node port exposed externally	Number
Port Exposure Method	Service exposure method	UNSET/EXTERNAL/NODE/HOST/INTERNAL/ROUTE
Unexpected Network Flow	Detected network traffic not in baseline	Boolean
Has Ingress Network	Presence of ingress	Boolean

Attribute	Description	Allowed Values
Policy	network policy	
Has Egress Network Policy	Presence of egress network policy	Boolean

Operators: Regex, AND, OR, NOT, None, etc.

Phases: Deploy, Runtime, Runtime (Network)

Process Activity (Runtime Only)

Attribute	Description	Allowed Values
Process Name	Name of executed process	String
Process Ancestor	Parent process name	String
Process Arguments	Command arguments	String
Process UID	Unix user ID	Integer
Unexpected Process Executed	Not in locked baseline	Boolean

Operators: Regex, AND, OR, NOT, None, etc.

Phases: Runtime (Process)

Kubernetes Access and Events

Attribute	Description	Allowed Values
Service Account	Name of the service account	String
Automount Service Account Token	Auto-mount service account token	Boolean
Minimum RBAC Permissions	Minimum RBAC permission level	DEFAULT/ELEVATED_IN_NAMESPACE/ELEVATED_(
Kubernetes Action	Name of Kubernetes action	PODS_EXEC/PODS_PORTFORWARD
Kubernetes User Name	Name of user accessing resource	String
Kubernetes User Groups	User group name	String
Kubernetes Resource Type	Type of accessed resource	String
Kubernetes API Verb	API verb used	CREATE/DELETE/GET/PATCH/UPDATE
Kubernetes Resource Name	Name of accessed resource	String
User Agent	User agent used	String

Attribute	Description	Allowed Values
Source IP Address	Source IP address	IPv4/IPv6
Is Impersonated User	Request made by impersonated user	Boolean

Operators: Regex, AND, OR, NOT, None, etc.

Phases: Deploy, Runtime, Runtime (K8s Events), Runtime (Audit Log)

Policy Enforcement

Alauda Container Security supports multiple enforcement types depending on the policy phase:

- **Build-time enforcement**: Fails CI builds if images violate policy. The API returns a non-zero exit code, which can be used to fail the build pipeline.
- **Deploy-time enforcement**: Integrates with Kubernetes admission controllers and Alauda Container Platform admission plugins to block noncompliant workloads. Enforcement can be:
  - **Hard enforcement**: Admission controller blocks creation or update of violating deployments.
  - **Soft enforcement**: Sensor scales violating deployments to zero replicas, preventing pods from being scheduled.
- **Runtime enforcement**: When enabled, any runtime activity within a pod that violates this policy will cause the pod to be automatically deleted. Violations triggered via the API server will also be blocked.

> **Note:** By default, administrative namespaces such as `stackrox`, `kube-system`, `cpaas-system`, and `istio-system` are excluded from enforcement blocking. Requests from service accounts in system namespaces are also bypassed.

To apply policy changes to existing deployments, use **Policy Management** > **Reassess All** to trigger enforcement on all deployments.

Exporting and Importing Policies

You can share security policies between different Central instances by exporting and importing policies as JSON files.

Exporting a Policy

1. Go to **Platform Configuration** > **Policy Management**.
2. Select the policy to export.
3. Click **Actions** > **Export policy to JSON**.

Importing a Policy

1. Go to **Platform Configuration** > **Policy Management**.
2. Click **Import Policy**.
3. Upload the JSON file and click **Begin Import**.

Import Handling:

- If the imported policy UID and name are unique, a new policy is created.
- If the UID matches but the name differs, you can keep both (new UID) or replace the existing policy.
- If the name matches but the UID differs, you can keep both (rename) or replace the existing policy.

- If both UID and name match, Alauda Container Security checks if the criteria match. If so, the existing policy is kept; otherwise, you can keep both (rename) or replace.

> **Important:**
>
> - When importing into the same Central instance, all exported fields are used.
> - When importing into a different Central instance, certain fields (e.g. cluster scopes exclusions notifications) are omitted and cannot be migrated.

Creating Custom Policies in Alauda Container Security

Alauda Container Security allows you to create custom security policies in addition to using the default ones. You can create and manage policies through the web portal or as code using Kubernetes custom resources (CRs).

TOC

Methods to Create Custom Policies
Creating Policies via the Portal
  Enter Policy Details
  Configure Policy Lifecycle
  Define Policy Rules and Criteria
  Set Policy Scope
  Configure Policy Actions
  Review and Save Policy
Editing and Managing Policies

Methods to Create Custom Policies

- In the Alauda Container Security portal, go to **Platform Configuration > Policy Management** and click **Create Policy**.
- In the **Risk** section, use filters to select criteria and click **Create Policy**.
- Manage policies as code by saving them as Kubernetes CRs and applying them to clusters using tools like Argo CD.

Creating Policies via the Portal

Enter Policy Details

- **Name**: Enter a name for the policy.
- **Severity**: Select a severity level.
- **Category**: Choose a policy category (required).
- **Description**: Provide details about the policy.
- **Rationale**: Explain the reason for the policy.
- **Guidance**: Add steps to resolve violations.
- **MITRE ATT&CK**: Select relevant tactics and techniques.

Configure Policy Lifecycle

- Select applicable **Lifecycle Stages**: **Build**, **Deploy**, or **Runtime**.
- For **Runtime**, choose an **Event Source**: **Deployment** or **Audit logs**.

Define Policy Rules and Criteria

- In the **Rules** section, set conditions to trigger the policy.
- Drag and drop policy fields to build rules. Available fields depend on the selected lifecycle stage.
- Combine multiple values or rules using logical operators (**AND**/**OR**).

Set Policy Scope

- **Inclusion Scope**: Restrict policy to specific clusters, namespaces, or deployment labels. Supports RE2 regex for namespaces and labels.
- **Exclusion Scope**: Exclude specific deployments, clusters, namespaces, or labels. Regex supported for namespaces and labels (not for deployments).
- For **Build** stage, you can exclude images from the policy.

Configure Policy Actions

- **Activation State**: Set the policy as active or inactive.
- **Enforcement**:
  - **Inform**: Only report violations.
  - **Inform and enforce**: Enforce actions based on lifecycle stage:
    - **Build**: Fails CI builds for noncompliant images.
    - **Deploy**: Blocks or edits noncompliant deployments if admission controller is enabled.
    - **Runtime**: Deletes pods matching policy criteria.
- **Notifiers**: Attach notifiers to send alerts to email or external tools (e.g., Jira, Splunk, webhooks). Notifiers must be pre-configured in **Platform Configuration > Integrations**.

Review and Save Policy

- Review all settings and preview potential violations.
- Click **Save** to create the policy.

Editing and Managing Policies

- To edit a policy, go to **Platform Configuration > Policy Management**, select a policy, and click **Actions > Edit Policy**.
- Default policies cannot be edited directly; clone them first.

Default Policies in Alauda Container Security

Alauda Container Security offers a set of default policies to help you prevent high-risk deployments and respond to runtime incidents in your Kubernetes environment. These policies are designed to identify security issues and enforce best practices across your clusters.

TOC

Overview

Viewing Policies

Policy Table Structure

Critical Severity Policies

High Severity Policies

Medium Severity Policies

Low Severity Policies

Managing Default Policies

Overview

Default policies cover the entire container lifecycle: build, deploy, and runtime. You can view, clone, and edit these policies in the Alauda Container Security portal. **Default policies cannot be deleted or directly modified.**

Viewing Policies

1. Go to **Platform Configuration** > **Policy Management** in the portal.

2. The **Policies** view lists all default and custom policies, including their status, severity, and lifecycle stage.

Policy Table Structure

- **Policy**: Policy name
- **Description**: What the policy detects or enforces
- **Status**: **Enabled** or **Disabled**
- **Severity**: Critical, High, Medium, or Low
- **Lifecycle**: Build, Deploy, or Runtime

Critical Severity Policies

Lifecycle Stage	Policy Name	Description	Status
Build/Deploy	Apache Struts: CVE-2017-5638	Alerts on images with the CVE-2017-5638 Apache Struts vulnerability.	Enabled
Build/Deploy	Log4Shell: log4j Remote Code Execution	Alerts on images with CVE-2021-44228 and CVE-2021-45046 vulnerabilities.	Enabled
Build/Deploy	Spring4Shell & Spring Cloud Function	Alerts on images with CVE-2022-22965 (Spring MVC) or CVE-2022-22963 (Spring Cloud).	Enabled

Lifecycle Stage	Policy Name	Description	Status
Runtime	Iptables Executed in Privileged Container	Alerts when privileged pods run iptables.	Enabled

High Severity Policies

Lifecycle Stage	Policy Name	Description	Status
Build/Deploy	Fixable CVSS >= 7	Alerts on fixable vulnerabilities with CVSS ≥ 7.	Disabled
Build/Deploy	Fixable Severity at least Important	Alerts on fixable vulnerabilities rated Important or higher.	Enabled
Build/Deploy	Rapid Reset: HTTP/2 DoS Vulnerability	Alerts on images susceptible to HTTP/2 Rapid Reset DoS.	Disabled
Build/Deploy	Secure Shell (ssh) Port Exposed in Image	Alerts when port 22 is exposed in images.	Enabled
Deploy	Emergency Deployment Annotation	Alerts on deployments using emergency annotations to bypass admission checks.	Enabled
Deploy	Environment Variable Contains Secret	Alerts when environment variables contain 'SECRET'.	Enabled
Deploy	Fixable CVSS >= 6 and Privileged	Alerts on privileged deployments with fixable CVSS ≥ 6 vulnerabilities.	Disabled

Lifecycle Stage	Policy Name	Description	Status
Deploy	Privileged Containers with Important and Critical Fixable CVEs	Alerts on privileged containers with important/critical fixable vulnerabilities.	Enabled
Deploy	Secret Mounted as Environment Variable	Alerts when secrets are mounted as environment variables.	Disabled
Deploy	Secure Shell (ssh) Port Exposed	Alerts when port 22 is exposed in deployments.	Enabled
Runtime	Cryptocurrency Mining Process Execution	Detects crypto-currency mining processes.	Enabled
Runtime	iptables Execution	Detects iptables usage in containers.	Enabled
Runtime	Kubernetes Actions: Exec into Pod	Alerts on exec commands run in containers via Kubernetes API.	Enabled
Runtime	Linux Group Add Execution	Detects groupadd/addgroup usage.	Enabled
Runtime	Linux User Add Execution	Detects useradd/adduser usage.	Enabled
Runtime	Login Binaries	Detects login attempts.	Disabled
Runtime	Network Management Execution	Detects network configuration commands.	Enabled
Runtime	nmap Execution	Alerts on nmap process execution.	Enabled
Runtime	OpenShift: Kubeadmin Secret Accessed	Alerts on kubeadmin secret access.	Enabled

Lifecycle Stage	Policy Name	Description	Status
Runtime	Password Binaries	Detects password change attempts.	Disabled
Runtime	Process Targeting Cluster Kubelet Endpoint	Detects misuse of kubelet/heapster endpoints.	Enabled
Runtime	Process Targeting Cluster Kubernetes Docker Stats Endpoint	Detects misuse of docker stats endpoint.	Enabled
Runtime	Process Targeting Kubernetes Service Endpoint	Detects misuse of Kubernetes Service API endpoint.	Enabled
Runtime	Process with UID 0	Alerts on processes running as UID 0.	Disabled
Runtime	Secure Shell Server (sshd) Execution	Detects SSH daemon execution in containers.	Enabled
Runtime	SetUID Processes	Detects setuid binary usage.	Disabled
Runtime	Shadow File Modification	Detects shadow file modifications.	Disabled
Runtime	Shell Spawned by Java Application	Detects shell spawned as a subprocess of Java apps.	Enabled
Runtime	Unauthorized Network Flow	Alerts on anomalous network flows.	Enabled
Runtime	Unauthorized Processed Execution	Alerts on unauthorized process execution in locked baselines.	Enabled

Medium Severity Policies

Lifecycle Stage	Policy Name	Description	Status
Build	Docker CIS 4.4: Ensure images are scanned and rebuilt	Alerts if images are not scanned and rebuilt with security patches.	Disabled
Deploy	30-Day Scan Age	Alerts if a deployment hasn't been scanned in 30 days.	Enabled
Deploy	CAP_SYS_ADMIN capability added	Alerts if containers escalate with CAP_SYS_ADMIN.	Enabled
Deploy	Container using read-write root filesystem	Alerts if containers have read-write root filesystems.	Disabled
Deploy	Container with privilege escalation allowed	Alerts if containers allow privilege escalation.	Enabled
Deploy	Deployments should have at least one Ingress Network Policy	Alerts if deployments lack an Ingress Network Policy.	Disabled
Deploy	Deployments with externally exposed endpoints	Alerts if deployments have externally exposed services.	Disabled
Deploy	Docker CIS 5.1: AppArmor profile enabled	Alerts if AppArmor is not enabled.	Enabled
Deploy	Docker CIS 5.15: Host's process namespace not shared	Alerts if host's process namespace is shared.	Enabled
Deploy	Docker CIS 5.16: Host's IPC namespace not shared	Alerts if host's IPC namespace is shared.	Enabled

Lifecycle Stage	Policy Name	Description	Status
Deploy	Docker CIS 5.19: Mount propagation mode not enabled	Alerts if mount propagation mode is enabled.	Enabled
Deploy	Docker CIS 5.21: Default seccomp profile not disabled	Alerts if seccomp profile is disabled.	Disabled
Deploy	Docker CIS 5.7: Privileged ports mapped within containers	Alerts if privileged ports (<1024) are mapped.	Enabled
Deploy	Docker CIS 5.9/5.20: Host's network namespace not shared	Alerts if host's network namespace is shared.	Enabled
Deploy	Images with no scans	Alerts if images in deployments are not scanned.	Disabled
Runtime	Kubernetes Actions: Port Forward to Pod	Alerts on port forward requests via Kubernetes API.	Enabled
Deploy	Mount Container Runtime Socket	Alerts if container runtime socket is mounted.	Enabled
Deploy	Mounting Sensitive Host Directories	Alerts if sensitive host directories are mounted.	Enabled
Deploy	No resource requests or limits specified	Alerts if containers lack resource requests/limits.	Enabled
Deploy	Pod Service Account Token Automatically Mounted	Alerts if default service account token is mounted unnecessarily.	Enabled

Lifecycle Stage	Policy Name	Description	Status
Deploy	Privileged Container	Alerts if containers run in privileged mode.	Enabled
Runtime	crontab Execution	Detects crontab usage.	Enabled
Runtime	Netcat Execution Detected	Detects netcat usage.	Enabled
Runtime	OpenShift: Central Admin Secret Accessed	Alerts on access to Central Admin secret.	Enabled
Runtime	OpenShift: Secret Accessed by Impersonated User	Alerts on secret access by impersonated users.	Enabled
Runtime	Remote File Copy Binary Execution	Alerts on remote file copy tool execution.	Enabled

Low Severity Policies

Lifecycle Stage	Policy Name	Description	Status
Build/Deploy	90-Day Image Age	Alerts if a deployment hasn't been updated in 90 days.	Enabled
Build/Deploy	ADD Command used instead of COPY	Alerts if ADD command is used in Dockerfile.	Disabled
Build/Deploy	Alpine Linux Package Manager (apk) in Image	Alerts if apk is present in images.	Enabled

Lifecycle Stage	Policy Name	Description	Status
Build/Deploy	Curl in Image	Alerts if curl is present in images.	Disabled
Build/Deploy	Docker CIS 4.1: User for the Container Created	Ensures containers run as non-root users.	Enabled
Build/Deploy	Docker CIS 4.7: Alert on Update Instruction	Ensures update instructions are not used alone in Dockerfile.	Enabled
Build/Deploy	Insecure specified in CMD	Alerts if 'insecure' is used in command.	Enabled
Build/Deploy	Latest tag	Alerts if images use the 'latest' tag.	Enabled
Build/Deploy	Red Hat Package Manager in Image	Alerts if Red Hat, Fedora, or CentOS package managers are present.	Enabled
Build/Deploy	Required Image Label	Alerts if images are missing required labels.	Disabled
Build/Deploy	Ubuntu Package Manager Execution	Detects Ubuntu package manager usage.	Enabled
Build/Deploy	Ubuntu Package Manager in Image	Alerts if Debian/Ubuntu package managers are present in images.	Enabled
Build/Deploy	Wget in Image	Alerts if wget is present in images.	Disabled
Deploy	Drop All Capabilities	Alerts if deployments do not drop all capabilities.	Disabled

Lifecycle Stage	Policy Name	Description	Status
Deploy	Improper Usage of Orchestrator Secrets Volume	Alerts if Dockerfile uses 'VOLUME /run/secrets'.	Enabled
Deploy	Kubernetes Dashboard Deployed	Alerts if a Kubernetes dashboard service is detected.	Enabled
Deploy	Required Annotation: Email	Alerts if 'email' annotation is missing.	Disabled
Deploy	Required Annotation: Owner/Team	Alerts if 'owner' or 'team' annotation is missing.	Disabled
Deploy	Required Label: Owner/Team	Alerts if 'owner' or 'team' label is missing.	Disabled
Runtime	Alpine Linux Package Manager Execution	Alerts if apk is run at runtime.	Enabled
Runtime	chkconfig Execution	Detects chkconfig usage.	Enabled
Runtime	Compiler Tool Execution	Alerts if compiler binaries are run at runtime.	Enabled
Runtime	Red Hat Package Manager Execution	Alerts if Red Hat, Fedora, or CentOS package managers are run at runtime.	Enabled
Runtime	Shell Management	Alerts on shell add/remove commands.	Disabled
Runtime	systemctl Execution	Detects systemctl usage.	Enabled
Runtime	systemd Execution	Detects systemd usage.	Enabled

Managing Default Policies

- Default policies provide broad security coverage.
- You can **view**, **clone**, and **edit** cloned default policies in the portal.
- **Default policies cannot be deleted or directly modified.**

> **Note:** Default policies are not supported with the policies-as-code feature.

Menu

How To

Checking Policy Compliance with roxctl

Learn how to check policy compliance using roxctl in Alauda Container Security.

Prerequisites

Output Formats

Output Options

Checking Policy Compliance for Deployments

Checking Policy Compliance for Images

Viewing Image Scan Results

Use Policy to Verify Image Signature in Alauda Container Security

Learn how to use policies to verify image signatures in Alauda Container Security.

Supported Signature Verification Methods

Prerequisites

Configure Signature Integration

Create and Enforce Image Signature Verification Policies

Checking Policy Compliance with roxctl

Alauda Container Security provides the `roxctl` CLI to help you check deployment YAML files and container images for policy compliance. This guide explains how to use `roxctl` for these checks and interpret the results.

TOC

Prerequisites
Output Formats
Example
Output Options
Example: Custom Headers and JSONPath
Checking Policy Compliance for Deployments
Checking Policy Compliance for Images
Viewing Image Scan Results

Prerequisites

- Set the `ROX_ENDPOINT` environment variable:

  ```
  export ROX_ENDPOINT=<host:port>
  ```

  Replace `<host:port>` with the address of your Alauda Container Security Central instance.

Output Formats

When running `roxctl deployment check` or `roxctl image check`, you can specify the output format using the `-o` option. Supported formats are `json`, `table`, `csv`, and `junit`. If not specified, the default is `table` for deployment and image checks, and `json` for image scans.

Example

```
roxctl deployment check --file=<yaml_filename> -o csv
```

Output Options

The following table summarizes the available output options:

Option	Description	Formats
`--compact-output`	Display JSON output in a compact format.	`json`
`--headers`	Specify custom headers.	`table`, `csv`
`--no-header`	Omit the header row from the output.	`table`, `csv`
`--row-jsonpath-expressions`	Use GJSON paths to select specific data.	`table`, `csv`
`--merge-output`	Merge table cells with the same value.	`table`
`headers-as-comment`	Include the header row as a comment in the output.	`csv`

Option	Description	Formats
`--junit-suite-name`	Specify the name of the JUnit test suite.	`junit`

Example: Custom Headers and JSONPath

```
roxctl deployment check --file=<yaml_filename> \
  -o table --headers POLICY-NAME,SEVERITY \
  --row-jsonpath-expressions="{results..violatedPolicies..name,results..viola
```

Checking Policy Compliance for Deployments

To check build-time and deploy-time policy violations in your deployment YAML files, run:

```
roxctl deployment check --file=<yaml_filename> \
  --namespace=<cluster_namespace> \
  --cluster=<cluster_name_or_id> \
  --verbose
```

- `<yaml_filename>`: Path to the deployment YAML file(s). You can specify multiple files by repeating the `--file` flag.
- `<cluster_namespace>`: (Optional) Namespace for context. Default is `default`.
- `<cluster_name_or_id>`: (Optional) Cluster name or ID for context.
- `--verbose`: (Optional) Show additional information, such as RBAC permissions and network policies.

> **Note:** Additional deployment information is included in JSON output, regardless of the `--verbose` flag.

To force Alauda Container Security to re-pull image metadata and scan results, add the `--force` option.

> **Permission Requirement:** To check specific image scan results, your token must have both `read` and `write` permissions for the `Image` resource. The default **Continuous Integration** system role includes these permissions.

The deployment check validates:

- Configuration options in the YAML file (e.g., resource limits, privilege settings)
- Image aspects (e.g., components, vulnerabilities)

Checking Policy Compliance for Images

To check build-time policy violations in images, run:

```
roxctl image check --image=<image_name>
```

To force Alauda Container Security to re-pull image metadata and scan results, add the `--force` option.

> **Permission Requirement:** To check specific image scan results, your token must have both `read` and `write` permissions for the `Image` resource. The default **Continuous Integration** system role includes these permissions.

Viewing Image Scan Results

To view the components and vulnerabilities found in an image in JSON format, run:

```
roxctl image scan --image=<image_name>
```

To force Alauda Container Security to re-pull image metadata and scan results, add the `--force` option.

> **Permission Requirement:** To check specific image scan results, your token must have both `read` and `write` permissions for the `Image` resource. The default **Continuous Integration** system role includes these permissions.

Use Policy to Verify Image Signature

Alauda Container Security allows you to ensure the integrity of container images in your clusters by verifying image signatures against pre-configured keys. You can create policies to block unsigned images or images without a verified signature, and enforce these policies using the admission controller to prevent unauthorized deployments.

TOC

Supported Signature Verification Methods

Prerequisites

Configure Signature Integration

  Using Cosign Public Keys

  Using Cosign Certificates

Create and Enforce Image Signature Verification Policies

  Prerequisites

  Procedure

Supported Signature Verification Methods

Alauda Container Security supports the following signature verification methods:

- Cosign public keys
- Cosign certificates

> **Note:**

> - Only Cosign signatures and Cosign Public Keys/Certificates verification are supported. For more information, see Cosign overview.
> - Communication with the transparency log Rekor is not supported.
> - At least one Cosign verification method must be configured for signature verification.
> - For all deployed and watched images:
>   - Signatures are fetched and verified every 4 hours.
>   - Signatures are verified whenever you update signature integration verification data.

Prerequisites

- You must have a PEM-encoded Cosign public key or the required certificate identity and issuer. For more details, see Cosign overview and Cosign certificate verification.

Configure Signature Integration

Using Cosign Public Keys

1. In the Alauda Container Security portal, go to **Platform Configuration** > **Integrations**.
2. Scroll to **Signature Integrations** and click **Signature**.
3. Click **New integration**.
4. Enter a name for the integration.
5. Click **Cosign public Keys** and then **Add a new public key**.
6. Enter the public key name and the PEM-encoded public key value.
7. (Optional) Add more public keys as needed.
8. Click **Save**.

Using Cosign Certificates

1. In the Alauda Container Security portal, go to **Platform Configuration** > **Integrations**.
2. Scroll to **Signature Integrations** and click **Signature**.
3. Click **New integration**.
4. Enter a name for the integration.
5. Click **Cosign certificates** and then **Add a new certificate verification**.
6. Enter the **Certificate OIDC Issuer** (regular expressions in RE2 Syntax ↗ are supported).
7. Enter the **Certificate identity** (regular expressions in RE2 Syntax ↗ are supported).
8. (Optional) Enter the **Certificate Chain PEM encoded** to verify certificates. If not provided, certificates are verified against the Fulcio ↗ root.
9. (Optional) Enter the **Certificate PEM encoded** to verify the signature.
10. (Optional) Add more certificate verifications as needed.
11. Click **Save**.

Create and Enforce Image Signature Verification Policies

Prerequisites

- At least one Cosign public key must be configured in a signature integration.

Procedure

1. When creating or editing a policy, drag the **Not verified by trusted image signers** criteria into the policy field under **Policy criteria**.
2. Click **Select**.
3. Choose the trusted image signers from the list and click **Save**.

To prevent the use of unsigned images, enable the **Contact Image Scanners** feature in your cluster configuration. Then, when creating a security policy to enforce signature verification, select the **Inform and enforce** option.

For more information, refer to the official Cosign documentation ↗ .

Menu

Configuration

Managing Deployment Collections

Learn how to manage deployment collections in Alauda Container Security.

Overview

Prerequisites

Benefits of Collections

Accessing and Managing Collections

Creating a Deployment Collection

Regular Expression Examples

Attaching Collections

Migration from Access Scopes

API Usage

API Token Configuration

Key Points

Procedure

Token Expiration and Notification

Configuring Notification Settings

Integrating with a Generic Docker Registry

Prerequisites

Integration Steps

Integration with Email

Configuring Email Integration

Enabling Email Notifications for Policies

Managing Deployment Collections

Alauda Container Security allows you to define and manage deployment collections, which are logical groupings of resources based on matching patterns. Collections help you organize your infrastructure and streamline configuration management.

TOC

Overview
Prerequisites
Benefits of Collections
Accessing and Managing Collections
Creating a Deployment Collection
  Steps
  Defining Collection Rules
Regular Expression Examples
  Match Production Clusters
  Match Non-Production Clusters
  Match All Entities
  Match Specific Deployments and Labels
Attaching Collections
Migration from Access Scopes
API Usage

Overview

Collections in Alauda Container Security are user-defined, named references that group deployments, namespaces, or clusters using selection rules. These rules can be based on exact matches or regular expressions (RE2 syntax is supported). Collections can also be nested, allowing you to build complex hierarchies.

Key Points:

- Collections are currently available only for deployments.
- Collections are used with vulnerability reporting.
- Deployment collections require the PostgreSQL database backend.

Prerequisites

To use collections, your account must have the following permissions:

- **WorkflowAdministration**: Read access to view collections; Write access to add, modify, or delete collections.
- **Deployment**: Read or Read/Write access to view how rules match deployments.

These permissions are included in the `Admin` system role. For more details, see the RBAC management documentation.

Benefits of Collections

Collections provide a flexible way to:

- Group resources owned by specific teams.
- Apply different policies for development and production environments.
- Manage distributed applications spanning multiple namespaces or clusters.
- Organize production or test environments efficiently.

Accessing and Managing Collections

You can manage collections through the Alauda Container Security portal:

1. Navigate to **Platform Configuration > Collections**.
2. The page displays a list of existing collections. You can:

- Search collections by name.
- View collection details in read-only mode.
- Edit, clone, or delete collections (collections in use cannot be deleted).
- Create new deployment collections.

Creating a Deployment Collection

Steps

1. Click **Create collection**.
2. Enter a name and description.
3. In **Collection rules**, do at least one of the following:

- Define selection rules (see below).
- Attach existing collections.

4. Use the live preview panel to see matching results.
5. Click **Save**.

> **Note:** At least one rule or attached collection is required.

Defining Collection Rules

You can configure rules to select resources for the collection:

- **Deployments**

- **No deployments specified**: Ignore deployment criteria.
- **Deployments with names matching**:
  - **Exact value**: Enter the deployment name.
  - **Regex value**: Use a regular expression (RE2 syntax) for pattern matching. For example, `.*` matches all deployments.
- **Deployments with labels matching exactly**: Enter a valid Kubernetes label in the format `key=value`.

- **Namespaces**
  - **Namespaces with names matching**: Use exact or regex values.
  - **Namespaces with labels matching exactly**: Enter a label in `key=value` format.
- **Clusters**
  - **Clusters with names matching**: Use exact or regex values.

To add more criteria, use the **OR** option to combine multiple rules.

Regular Expression Examples

Alauda Container Security supports RE2 syntax for regular expressions. Here are some common examples:

Match Production Clusters

To match clusters with names starting with `prod`:

```
^prod.*
```

Match Non-Production Clusters

To match clusters where `prod` does not appear in the name (RE2 does not support negative lookahead):

```
^[^p]*(p([^r]|$|r([^o]|$|o([^d]|$))))*[^p]*$
```

Match All Entities

To match all deployments, namespaces, and clusters:

- **Deployments with names matching**: `.*`
- **Namespaces with names matching**: `.*`
- **Clusters with names matching**: `.*`

Match Specific Deployments and Labels

To include the `reporting` deployment, any deployment ending with `-db`, and namespaces labeled `kubernetes.io/metadata.name=medical`:

- **Deployments with names matching**: `reporting`
- **OR**: Regex value `.*-db`
- **Namespaces with labels matching exactly**: `kubernetes.io/metadata.name=medical`

Attaching Collections

You can build hierarchical collections by attaching existing collections:

1. Filter collections by name or select from the list.
2. Click **+Attach** to add the selected collection.
3. Attached collections extend the parent collection using an OR relationship.

Migration from Access Scopes

When migrating from `rocksdb` to PostgreSQL, existing access scopes used in vulnerability reporting are converted to collections. The migration process creates embedded and root collections to replicate the original selection logic.

- **Embedded collections**: Mimic the original access scope logic.
- **Root collection**: Attaches embedded collections and is used in report configurations.

If a scope cannot be migrated (e.g., uses unsupported label selector operators), a log message is generated. Only the `IN` operator is supported for label selectors.

API Usage

Collections can also be managed via the `CollectionService` API. For example, `CollectionService_DryRunCollection` returns results similar to the live preview in the portal. Refer to the API reference in the portal for more details.

API Token Configuration

Alauda Container Security requires API tokens for system integrations, authentication, and various system functions. You can manage tokens through the Alauda Container Security web interface.

TOC

Key Points

Procedure

Token Expiration and Notification

Configuring Notification Settings

Key Points

- To prevent privilege escalation, when you create a new token, your role's permissions limit the permissions you can assign to that token. For example, if you only have `read` permission for the Integration resource, you cannot create a token with `write` permission.
- If you want a custom role to create tokens for other users, you must assign the required permissions to that custom role.
- Use short-lived tokens for machine-to-machine communication, such as CI/CD pipelines, scripts, and automation. For human-to-machine communication, such as CLI or API access, use the `roxctl central login` command.
- Most cloud service providers support OIDC identity tokens, such as Microsoft Entra ID, Google Cloud Identity Platform, and AWS Cognito. OIDC identity tokens issued by these services can be used for Alauda Container Security short-lived access.

Procedure

1. In the Alauda Container Security portal, go to **Platform Configuration** > **Integrations**.
2. Scroll to the **Authentication Tokens** category and click **API Token**.
3. Click **Generate Token**.
4. Enter a name for the token and select a role that provides the required level of access (for example, **Continuous Integration** or **Sensor Creator**).
5. Click **Generate**.

   Important:
   Copy the generated token and store it securely. You will not be able to view it again.

Token Expiration and Notification

API tokens expire one year from the creation date. Alauda Container Security alerts you in the web interface and by sending log messages to Central when a token will expire in less than one week. The log message process runs once an hour. Once a day, the process lists the tokens that are expiring and creates a log message for each one. Log messages are issued once a day and appear in Central logs.

Log message format:

```
Warn: API Token [token name] (ID [token ID]) will expire in less than X days.
```

Configuring Notification Settings

You can change the default settings for the log message process by configuring the following environment variables:

Environment Variable	Default Value	Descriptio
ROX_TOKEN_EXPIRATION_NOTIFIER_INTERVAL	1h	Frequency at which th backgroun process checks an logs expiring tokens.
ROX_TOKEN_EXPIRATION_NOTIFIER_BACKOFF_INTERVAL	24h	Frequency at which notificatior are issued for expiring tokens.
ROX_TOKEN_EXPIRATION_DETECTION_WINDOW	168h	Time perio before token expiration that trigger a notificatior (default: 1 week).

Integrating with a Generic Docker Registry

This guide explains how to integrate Alauda Container Security with Docker Registry or Harbor.

TOC

Prerequisites

Integration Steps

Prerequisites

- Obtain a username and password for authenticating with the Docker Registry or Harbor.

Integration Steps

1. In the Alauda Container Security portal, navigate to **Platform Configuration** > **Integrations**.
2. Under the **Image Integrations** section, select **Generic Docker Registry**.
3. Click **New integration**.
4. Fill in the following fields:

- **Integration name**: Enter a name for this integration.
- **Endpoint**: Specify the registry address.
- **Username** and **Password**: Enter your credentials.

5. (Optional) If you are not using a TLS certificate to connect to the registry, select **Disable TLS certificate validation (insecure)**.
6. To verify the connection, click **Test**. If you prefer to skip testing, select **Create integration without testing**.
7. Click **Save** to complete the integration.

Integration with Email

Alauda Container Security supports sending notifications via email. You can configure your existing email provider or use the built-in email notifier to send alerts. Notifications can be sent to a default recipient or dynamically determined using annotations in your deployment or namespace.

INFO

Port `25` is blocked by default. Configure your mail server to use port `587` or `465` for sending email notifications.

TOC

Configuring Email Integration

Add a New Email Integration

Dynamic Recipient with Annotations (Optional)

TLS and StartTLS Settings

Enabling Email Notifications for Policies

Configuring Email Integration

Follow these steps to set up email notifications:

Add a New Email Integration

1. Navigate to **Platform Configuration** > **Integrations**.

2. Under **Notifier Integrations**, select **Email**.
3. Click **New Integration**.
4. Enter a name for your integration in the **Integration name** field.
5. In the **Email server** field, provide the address of your email server, including the FQDN and port (e.g., `smtp.example.com:465`).
6. (Optional) To use unauthenticated SMTP, select **Enable unauthenticated SMTP**.

   WARNING

   This is insecure and not recommended unless required for internal servers.

7. Enter the username and password for the service account used for authentication.
8. (Optional) Specify the display name for the `FROM` header in the **From** field (e.g., `Security Alerts`).
9. Enter the sender's email address in the **Sender** field.
10. Specify the default recipient's email address in the **Default recipient** field.

Dynamic Recipient with Annotations (Optional)

You can use annotations to dynamically determine the recipient of email notifications:

1. In the **Annotation key for recipient** field, enter an annotation key (e.g., `email`).
2. Add an annotation to your deployment or namespace YAML file:

   ```
   metadata:
     annotations:
       email: <recipient_email@example.com>
   ```

3. Alauda Container Security will send alerts to the email specified in the annotation. If no annotation is found, the alert is sent to the default recipient.

Recipient Resolution Rules:

- If a deployment has the annotation key, its value overrides the default recipient.

- If the namespace has the annotation key, its value overrides the default recipient.
- If neither exists, the default recipient is used.

TLS and StartTLS Settings

- (Optional) To send email without TLS, select **Disable TLS certificate validation (insecure)**.

  > **INFO**
  >
  > It is recommended to use TLS for secure email delivery.

- (Optional) To use StartTLS, select either **Login** or **Plain** from the **Use STARTTLS** drop-down menu.
  - **Login:** Credentials are sent as a base64-encoded string.
  - **Plain:** Credentials are sent in plain text.

> **WARNING**
>
> With StartTLS, credentials are transmitted in plain text before encryption is established.

Enabling Email Notifications for Policies

1. In the Alauda Container Security portal, go to **Platform Configuration** > **Policy Management**.
2. Select one or more policies to enable notifications for.
3. Under **Bulk actions**, select **Enable notification**.
4. In the **Enable notification** window, choose the **Email** notifier.
5. Click **Enable**.

> **INFO**

- Notifications are opt-in. Assign a notifier to each policy to receive alerts.
- Notifications are sent only once per alert. A new alert is generated for:
  - The first policy violation in a deployment.
  - A runtime-phase policy violation after the previous alert is resolved.

By following these steps, you can ensure that Alauda Container Security notifies the right people about important security events in your container platform.