Red Hat Advanced Cluster Security for Kubernetes 4.3

roxctl CLI

roxctl CLI
Abstract

This document describes how to install and use the roxctl command-line interface, including the roxctl syntax and operations. It provides some common command examples.
# Table of Contents

## CHAPTER 1. INSTALLING THE ROXCTL CLI

1.1. INSTALLING THE ROXCTL CLI BY DOWNLOADING THE BINARY

1.1.1. Installing the roxctl CLI on Linux

1.1.2. Installing the roxctl CLI on macOS

1.1.3. Installing the roxctl CLI on Windows

1.2. RUNNING THE ROXCTL CLI FROM A CONTAINER

## CHAPTER 2. USING THE ROXCTL CLI

2.1. PREREQUISITES

2.2. GETTING AUTHENTICATION INFORMATION

2.3. AUTHENTICATING BY USING THE ROXCTL CLI

2.3.1. Creating an API token

2.3.2. Exporting and saving the API token

2.3.3. Using an authentication provider to authenticate with roxctl

2.4. CONFIGURING AND USING THE ROXCTL CLI IN RHACS CLOUD SERVICE

## CHAPTER 3. MANAGING SECURED CLUSTERS

3.1. PREREQUISITES

3.2. GENERATING SENSOR DEPLOYMENT FILES

   Generating files for Kubernetes systems

   Generating files for OpenShift Container Platform systems

3.3. INSTALLING SENSOR BY USING THE SENSOR.SH SCRIPT

3.4. DOWNLOADING SENSOR BUNDLES FOR EXISTING CLUSTERS

3.5. DELETING CLUSTER INTEGRATION

## CHAPTER 4. CHECKING POLICY COMPLIANCE

4.1. PREREQUISITES

4.2. CONFIGURING OUTPUT FORMAT

4.3. CHECKING DEPLOYMENT YAML FILES

4.4. CHECKING IMAGES

4.5. CHECKING IMAGE SCAN RESULTS

## CHAPTER 5. DEBUGGING ISSUES

5.1. PREREQUISITES

5.2. VIEWING THE LOGS

   Kubernetes

   OpenShift Container Platform

5.3. VIEWING THE CURRENT LOG LEVEL

5.4. CHANGING THE LOG LEVEL

5.5. RETRIEVING DEBUGGING INFORMATION

## CHAPTER 6. GENERATING BUILD-TIME NETWORK POLICIES

6.1. USING THE BUILD-TIME NETWORK POLICY GENERATOR

## CHAPTER 7. IMAGE SCANNING BY USING THE ROXCTL CLI

7.1. SCANNING IMAGES BY USING A REMOTE CLUSTER

7.2. ROXCTL IMAGE SCAN COMMAND OPTIONS
CHAPTER 1. INSTALLING THE ROXCTL CLI

roxctl is a command-line interface (CLI) for running commands on Red Hat Advanced Cluster Security for Kubernetes (RHACS). You can install the roxctl CLI by downloading the binary or you can run the roxctl CLI from a container image.

1. INSTALLING THE ROXCTL CLI BY DOWNLOADING THE BINARY

You can install the roxctl CLI to interact with RHACS from a command-line interface. You can install roxctl on Linux, Windows, or macOS.

1.1. Installing the roxctl CLI on Linux

You can install the roxctl CLI binary on Linux by using the following procedure.

NOTE

roxctl CLI for Linux is available for amd64, ppc64le, and s390x architectures.

Procedure

1. Determine the roxctl architecture for the target operating system:

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

2. Download the roxctl CLI:

   $ curl -f -o roxctl "https://mirror.openshift.com/pub/rhacs/assets/4.3.1/bin/Linux/roxctl${arch}"

3. Make the roxctl binary executable:

   $ chmod +x roxctl

4. Place the roxctl binary in a directory that is on your PATH:

   To check your PATH, execute the following command:

   $ echo $PATH

Verification

- Verify the roxctl version you have installed:

  $ roxctl version

1.1.2. Installing the roxctl CLI on macOS

You can install the roxctl CLI binary on macOS by using the following procedure.

NOTE

roxctl CLI for macOS is available for the amd64 architecture.
1. Download the `roxctl` CLI:
   ```bash
   $ curl -f -O https://mirror.openshift.com/pub/rhacs/assets/4.3.1/bin/Darwin/roxctl
   ```
2. Remove all extended attributes from the binary:
   ```bash
   $ xattr -c roxctl
   ```
3. Make the `roxctl` binary executable:
   ```bash
   $ chmod +x roxctl
   ```
4. Place the `roxctl` binary in a directory that is on your `PATH`:
   To check your `PATH`, execute the following command:
   ```bash
   $ echo $PATH
   ```

**Verification**

- Verify the `roxctl` version you have installed:
  ```bash
  $ roxctl version
  ```

### 1.1.3. Installing the roxctl CLI on Windows

You can install the `roxctl` CLI binary on Windows by using the following procedure.

**NOTE**

`roxctl` CLI for Windows is available for the `amd64` architecture.

**Procedure**

- Download the `roxctl` CLI:
  ```bash
  $ curl -f -O https://mirror.openshift.com/pub/rhacs/assets/4.3.1/bin/Windows/roxctl.exe
  ```

**Verification**

- Verify the `roxctl` version you have installed:
  ```bash
  $ roxctl version
  ```

### 1.2. RUNNING THE ROXCTL CLI FROM A CONTAINER

The `roxctl` client is the default entry point in the RHACS `roxctl` image. To run the `roxctl` client in a container image:
Prerequisites

- You must first generate an authentication token from the RHACS portal.

Procedure

1. Log in to the `registry.redhat.io` registry.
   
   ```
   $ docker login registry.redhat.io
   ```

2. Pull the latest container image for the `roxctl` CLI.
   
   ```
   $ docker pull registry.redhat.io/advanced-cluster-security/rhacs-roxctl-rhel8:4.3.1
   ```

After you install the CLI, you can run it by using the following command:

```
$ docker run -e ROX_API_TOKEN=$ROX_API_TOKEN \
  -it registry.redhat.io/advanced-cluster-security/rhacs-roxctl-rhel8:4.3.1 \
  -e $ROX_CENTRAL_ADDRESS <command>
```

NOTE

In Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service), when using `roxctl` commands that require the Central address, use the Central instance address as displayed in the Instance Details section of the Red Hat Hybrid Cloud Console. For example, use `acs-ABCD12345.acs.rhcloud.com` instead of `acs-data-ABCD12345.acs.rhcloud.com`.

Verification

- Verify the `roxctl` version you have installed.

  ```
  $ docker run -it registry.redhat.io/advanced-cluster-security/rhacs-roxctl-rhel8:4.3.1 version
  ```
CHAPTER 2. USING THE ROXCTL CLI

2.1. PREREQUISITES

- You have configured the ROX_ENDPOINT environment variable using the following command:

  ```
  $ export ROX_ENDPOINT=<host:port> 1
  The host and port information that you want to store in the ROX_ENDPOINT environment variable.
  ```

2.2. GETTING AUTHENTICATION INFORMATION

The following procedure describes how to use the roxctl central whoami command to retrieve information about your authentication status and user profile in Central. The example output illustrates the data you can expect to see, including user roles, access permissions, and various administrative functions. This step allows you to review your access and roles within Central.

**Procedure**

- Run the following command to get information about your current authentication status and user information in Central:

  ```
  $ roxctl central whoami
  ```

**Example output**

<table>
<thead>
<tr>
<th>UserID:</th>
<th>&lt;redacted&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name:</td>
<td>&lt;redacted&gt;</td>
</tr>
<tr>
<td>Roles:</td>
<td>APIToken creator, Admin, Analyst, Continuous Integration, Network Graph Viewer, None, Sensor Creator, Vulnerability Management Approver, Vulnerability Management Requester, Vulnerability Manager, Vulnerability Report Creator</td>
</tr>
<tr>
<td>Access:</td>
<td>rw Access, rw Administration, rw Alert, rw CVE, rw Cluster, rw Compliance, rw Deployment, rw DeploymentExtension, rw Detection, rw Image, rw Integration, rw K8sRole, rw K8sRoleBinding, rw K8sSubject, rw Namespace, rw NetworkGraph</td>
</tr>
</tbody>
</table>
2.3. AUTHENTICATING BY USING THE ROXCTL CLI

For authentication, you can use an API token, your administrator password, or the `roxctl central login` command.

Follow these guidelines for the effective use of API tokens:

- Use an API token in a production environment with continuous integration (CI). Each token is assigned specific access permissions, providing control over the actions it can perform. In addition, API tokens do not require interactive processes, such as browser-based logins, making them ideal for automated processes. These tokens have a time-to-live (TTL) value of 1 year, providing a longer validity period for seamless integration and operational efficiency.

- Use your administrator password only for testing purposes. Do not use it in the production environment.

- Use the `roxctl central login` command only for interactive, local uses.

2.3.1. Creating an API token

Procedure

1. In the RHACS portal, navigate to Platform Configuration → Integrations.

2. Scroll to the Authentication Tokens category, and then 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 securely store it. You will not be able to view it again.

2.3.2. Exporting and saving the API token

Procedure

- [ ]
1. After you have generated the authentication token, export it as the `ROX_API_TOKEN` variable by entering the following command:

```bash
$ export ROX_API_TOKEN=<api_token>
```

2. (Optional): You can also save the token in a file and use it with the `--token-file` option by entering the following command:

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

Note the following guidelines:

- You cannot use both the `--password (-p)` and the `--token-file` options simultaneously.

- If you have already set the `ROX_API_TOKEN` variable, and specify the `--token-file` option, the `roxctl` CLI uses the specified token file for authentication.

- If you have already set the `ROX_API_TOKEN` variable, and specify the `--password` option, the `roxctl` CLI uses the specified password for authentication.

2.3.3. Using an authentication provider to authenticate with roxctl

You can configure an authentication provider in Central and initiate the login process with the `roxctl` CLI. Set the `ROX_ENDPOINT` variable, initiate the login process with the `roxctl central login` command, select the authentication provider in a browser window, and retrieve the token information from the `roxctl` CLI as described in the following procedure.

Prerequisite

- You selected an authentication provider of your choice, such as OpenID Connect (OIDC) with fragment or query mode.

Procedure

1. Run the following command to set the `ROX_ENDPOINT` variable to Central hostname and port:

```bash
export ROX_ENDPOINT=<central_hostname:port>
```

2. Run the following command to initiate the login process to Central:

```bash
$ roxctl central login
```

3. Within the `roxctl` CLI, a URL is printed as output and you are redirected to a browser window where you can select the authentication provider you want to use.

4. Log in with your authentication provider.

   After you have successfully logged in, the browser window indicates that authentication was successful and you can close the browser window.

5. The `roxctl` CLI displays your token information including details such as the access token, the expiration time of the access token, the refresh token if one has been issued, and notification that these values are stored locally.

Example output
Please complete the authorization flow in the browser with an auth provider of your choice. If no browser window opens, please click on the following URL:
http://127.0.0.1:xxxxx/login

INFO: Received the following after the authorization flow from Central:
INFO: Access token: <redacted> 1
INFO: Refresh token: <redacted> 3
INFO: Storing these values under $HOME/.roxctl/login… 4

1. The access token.
2. The expiration time of the access token.
3. The refresh token.
4. The directory where values of the access token, the access token expiration time, and the refresh token are stored locally.

**IMPORTANT**

Ensure that you set the environment to determine the directory where the configuration is stored. By default, the configuration is stored in the $HOME/.roxctl/roxctl-config directory.

- If you set the $ROX_CONFIG_DIR environment variable, the configuration is stored in the $ROX_CONFIG_DIR/roxctl-config directory. This option has the highest priority.

- If you set the $XDG_RUNTIME_DIR environment variable and the $ROX_CONFIG_DIR variable is not set, the configuration is stored in the $XDG_RUNTIME_DIR /roxctl-config directory.

- If you do not set the $ROX_CONFIG_DIR or $XDG_RUNTIME_DIR environment variable, the configuration is stored in the $HOME/.roxctl/roxctl-config directory.

### 2.4. CONFIGURING AND USING THE ROXCTL CLI IN RHACS CLOUD SERVICE

**Procedure**

- Export the following variables before using these commands:

  $ export ROX_API_TOKEN=<api_token>

  $ export ROX_ENDPOINT=<address>:<port_number>

- You can use the **--help** option to get more information about the commands.

- In Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service), when using roxctl commands that require the Central address, use the **Central instance address** as displayed in
the **Instance Details** section of the Red Hat Hybrid Cloud Console. For example, use `acs-ABCD12345.acs.rhcloud.com` instead of `acs-data-ABCD12345.acs.rhcloud.com`.
CHAPTER 3. MANAGING SECURED CLUSTERS

To secure a Kubernetes or an OpenShift Container Platform cluster, you must deploy Red Hat Advanced Cluster Security for Kubernetes (RHACS) services into the cluster. You can generate deployment files in the RHACS portal by navigating to the Platform Configuration → Clusters view, or you can use the roxctl CLI.

3.1. PREREQUISITES

- You have configured the `ROX_ENDPOINT` environment variable using the following command:

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

  The host and port information that you want to store in the `ROX_ENDPOINT` environment variable.

3.2. GENERATING SENSOR DEPLOYMENT FILES

Generating files for Kubernetes systems

Procedure

- Generate the required sensor configuration for your Kubernetes cluster and associate it with your Central instance by running the following command:

  ```bash
  $ roxctl sensor generate k8s --name <cluster_name> --central "$ROX_ENDPOINT"
  ```

Generating files for OpenShift Container Platform systems

Procedure

- Generate the required sensor configuration for your OpenShift Container Platform cluster and associate it with your Central instance by running the following command:

  ```bash
  $ roxctl sensor generate openshift --openshift-version <ocp_version> --name <cluster_name> --central "$ROX_ENDPOINT"
  ```

  For the `--openshift-version` option, specify the major OpenShift Container Platform version number for your cluster. For example, specify 3 for OpenShift Container Platform version 3.x and specify 4 for OpenShift Container Platform version 4.x.

  Read the `--help` output to see other options that you might need to use depending on your system architecture.

  Verify that the endpoint you provide for `--central` can be reached from the cluster where you are deploying Red Hat Advanced Cluster Security for Kubernetes services.
IMPORTANT

If you are using a non-gRPC capable load balancer, such as HAProxy, AWS Application Load Balancer (ALB), or AWS Elastic Load Balancing (ELB), follow these guidelines:

- Use the WebSocket Secure (wss) protocol. To use wss, prefix the address with wss://, and
- Add the port number after the address, for example:

```
$ roxctl sensor generate k8s --central wss://stackrox-central.example.com:443
```

3.3. INSTALLING SENSOR BY USING THE SENSOR.SH SCRIPT

When you generate the Sensor deployment files, `roxctl` creates a directory called `sensor-<cluster_name>` in your working directory. The script to install Sensor is located in this directory.

Procedure

- Run the sensor installation script to install Sensor:

```
$ ./sensor-<cluster_name>/sensor.sh
```

If you get a warning that you do not have the required permissions to install Sensor, follow the on-screen instructions, or contact your cluster administrator for help.

3.4. DOWNLOADING SENSOR BUNDLES FOR EXISTING CLUSTERS

Procedure

- Run the following command to download Sensor bundles for existing clusters by specifying a cluster name or ID:

```
$ roxctl sensor get-bundle <cluster_name_or_id>
```

3.5. DELETING CLUSTER INTEGRATION

Procedure

- Before deleting the cluster, ensure you have the correct cluster name that you want to remove from Central:

```
$ roxctl cluster delete --name=<cluster_name>
```

IMPORTANT

Deleting the cluster integration does not remove the RHACS services running in the cluster, depending on the installation method. You can remove the services by running the `delete-sensor.sh` script from the Sensor installation bundle.
CHAPTER 4. CHECKING POLICY COMPLIANCE

You can use the roxctl CLI to check deployment YAML files and images for policy compliance.

4.1. PREREQUISITES

- You have configured the **ROX_ENDPOINT** environment variable using the following command:

  $ export ROX_ENDPOINT=<host:port>

  The host and port information that you want to store in the **ROX_ENDPOINT** environment variable.

4.2. CONFIGURING OUTPUT FORMAT

When you check policy compliance by using the roxctl deployment check or roxctl image check commands, you can specify the output format by using the -o option to the command and specifying the format as json, table, csv, or junit. This option determines how the output of a command is displayed in the terminal.

For example, the following command checks a deployment and then displays the result in csv format:

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

**NOTE**

When you do not specify the -o option for the output format, the following default behavior is used:

- The format for the deployment check and the image check commands is table.

- The default output format for the image scan command is json. This is the old JSON format output for compatibility with older versions of the CLI. To get the output in the new JSON format, specify the option with format, as -o json. Use the old JSON format output when gathering data for troubleshooting purposes.

Different options are available to configure the output. The following table lists the options and the format in which they are available.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>--compact-output</td>
<td>Use this option to display the JSON output in a compact format.</td>
<td>json</td>
</tr>
<tr>
<td>--headers</td>
<td>Use this option to specify custom headers.</td>
<td>table, csv</td>
</tr>
<tr>
<td>--no-header</td>
<td>Use this option to omit the header row from the output.</td>
<td>table, csv</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Formats</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>--row-jsonpath-expressions</td>
<td>Use this option to specify GJSON paths to select specific items from the output. For example, to get the Policy name and Severity for a deployment check, use the following command:</td>
<td></td>
</tr>
</tbody>
</table>
|                                | ```bash
$ roxctl deployment check --file=<yaml_filename> \
   -o table --headers POLICY-NAME,SEVERITY \ 
   --row-jsonpath-expressions="{results..violatedPolicies..name,results..violatedPolicies..severity}"
``` | table and csv |
| --merge-output                 | Use this option to merge table cells that have the same value.                                                                                                                                              | table   |
| headers-as-comment             | Use this option to include the header row as a comment in the output.                                                                                                                                       | csv     |
| --junit-suite-name             | Use this option to specify the name of the JUnit test suite.                                                                                                                                               | junit   |

### 4.3. CHECKING DEPLOYMENT YAML FILES

**Procedure**

- Run the following command to check the build-time and deploy-time violations of your security policies in YAML deployment files:

  ```bash
  $ roxctl deployment check --file=<yaml_filename>
  ```

  The format is defined in the API reference. To cause Red Hat Advanced Cluster Security for Kubernetes (RHACS) to re-pull image metadata and image scan results from the associated registry and scanner, add the **--force** option.

  **NOTE**

  To check specific image scan results, you must have a token with both **read** and **write** permissions for the **Image** resource. The default **Continuous Integration** system role already has the required permissions.

  This command validates the following items:

  - Configuration options in a YAML file, such as resource limits or privilege options
  - Aspects of the images used in a YAML file, such as components or vulnerabilities

### 4.4. CHECKING IMAGES
Procedure

- Run the following command to check the build-time violations of your security policies in images:

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

  The format is defined in the API reference. To cause Red Hat Advanced Cluster Security for Kubernetes (RHACS) to re-pull image metadata and image scan results from the associated registry and scanner, add the `--force` option.

  **NOTE**

  To check specific image scan results, you must have a token with both **read** and **write** permissions for the `Image` resource. The default **Continuous Integration** system role already has the required permissions.

### 4.5. CHECKING IMAGE SCAN RESULTS

You can also check the scan results for specific images.

Procedure

- Run the following command to return the components and vulnerabilities found in the image in JSON format:

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

  The format is defined in the API reference. To cause Red Hat Advanced Cluster Security for Kubernetes (RHACS) to re-pull image metadata and image scan results from the associated registry and scanner, add the `--force` option.

  **NOTE**

  To check specific image scan results, you must have a token with both **read** and **write** permissions for the `Image` resource. The default **Continuous Integration** system role already has the required permissions.
CHAPTER 5. DEBUGGING ISSUES

Central saves information to its container logs.

5.1. PREREQUISITES

- You have configured the `ROX_ENDPOINT` environment variable using the following command:

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

  The host and port information that you want to store in the `ROX_ENDPOINT` environment variable.

5.2. VIEWING THE LOGS

**Kubernetes**

**Procedure**

- Run the following command to view the logs for the Central pod:

  ```bash
  $ kubectl logs -n stackrox <central_pod>
  ```

**OpenShift Container Platform**

**Procedure**

- Run the following command to view the logs for the Central pod:

  ```bash
  $ oc logs -n stackrox <central_pod>
  ```

5.3. VIEWING THE CURRENT LOG LEVEL

You can change the log level to see more or less information in Central logs.

**Procedure**

- Run the following command to view the current log level:

  ```bash
  $ roxctl central debug log
  ```

5.4. CHANGING THE LOG LEVEL

**Procedure**

- Run the following command to change the log level:

  ```bash
  $ roxctl central debug log --level=<log_level>
  ```
The acceptable values for `<log_level>` are Panic, Fatal, Error, Warn, Info, and Debug.

5.5. RETRIEVING DEBUGGING INFORMATION

Procedure

- Run the following command to gather the debugging information for investigating issues:

  ```bash
  $ roxctl central debug dump
  ```

- To generate a diagnostic bundle with the RHACS administrator password or API token and central address, follow the procedure in Generating a diagnostic bundle by using the roxctl CLI.
CHAPTER 6. GENERATING BUILD-TIME NETWORK POLICIES

The build-time network policy generator is included in the roxctl CLI. For the build-time network policy generation feature, roxctl CLI does not need to communicate with RHACS Central so you can use it in any development environment.

6.1. USING THE BUILD-TIME NETWORK POLICY GENERATOR

IMPORTANT

Build-time network policy generation is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope.

Prerequisites

1. The build-time network policy generator recursively scans the directory you specify when you run the command. Therefore, before you run the command, you must already have service manifests, config maps, and workload manifests such as Pod, Deployment, ReplicaSet, Job, DaemonSet, and StatefulSet as YAML files in the specified directory.

2. Verify that you can apply these YAML files as-is using the kubectl apply -f command. The build-time network policy generator does not work with files that use Helm-style templating.

3. Verify that the service network addresses are not hardcoded. Every workload that needs to connect to a service must specify the service network address as a variable. You can specify this variable by using the workload’s resource environment variable or in a config map.

   ● Example 1: using an environment variable
   
   ● Example 2: using a config map
   
   ● Example 3: using a config map

4. Service network addresses must match the following official regular expression pattern:

   (http(s)?://)?<svc>(.<ns>(.svc.cluster.local)?)?(:<portNum>)?

   In this pattern,
   
   ● <svc> is the service name.
   
   ● <ns> is the namespace where you defined the service.
   
   ● <portNum> is the exposed service port number.

Following are some examples that match the pattern:
- wordpress-mysql:3306
- redis-follower.redis.svc.cluster.local:6379
- redis-leader.redis

Procedure

1. Verify that the build-time network policy generation feature is available by running the help command:

   $ roxctl netpol generate -h

2. Generate the policies by using the netpol generate command:

   $ roxctl netpol generate <folder-path>

   Specify the path of the folder that has the Kubernetes manifests.

The roxctl netpol generate command supports the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h, --help</td>
<td>View the help text for the netpol command.</td>
</tr>
<tr>
<td>-d, --output-dir &lt;dir&gt;</td>
<td>Save the generated policies into a target folder. One file per policy.</td>
</tr>
<tr>
<td>-f, --output-file &lt;filename&gt;</td>
<td>Save and merge the generated policies into a single YAML file.</td>
</tr>
<tr>
<td>--fail</td>
<td>Fail on the first encountered error. The default value is false.</td>
</tr>
<tr>
<td>--remove</td>
<td>Remove the output path if it already exist.</td>
</tr>
<tr>
<td>--strict</td>
<td>Treat warnings as errors. The default value is false.</td>
</tr>
</tbody>
</table>
CHAPTER 7. IMAGE SCANNING BY USING THE ROXCTL CLI

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

7.1. SCANNING IMAGES BY USING A REMOTE CLUSTER

By specifying the appropriate cluster in the delegated scanning configuration or through the cluster parameter described in the following procedure, you can scan images from cluster local registries by using a remote cluster.

IMPORTANT

For more information about how to configure delegated image scanning, see Configuring delegated image scanning.

Procedure

- Run the following command to scan the specified image in a remote cluster:

  ```
  $ roxctl image scan \
  --image=<image_registry>/<image_name> 1 \n  --cluster=<cluster_detail> 2 \n  [flags] 3
  ```

1 For `<image_registry>`, specify the registry where the image is located, for example, `image-registry.openshift-image-registry.svc:5000/`. For `<image_name>`, specify the name of the image you want to scan, for example, `default/image-stream:latest`.

2 For `<cluster_detail>`, specify the name or ID of the remote cluster. For example, specify the name `remote`.

3 Optional: For `[flags]`, you can specify parameters to modify the behavior of the command.

   For more information about optional parameters, see roxctl image scan command options.

Example output

```json
{
  "Id": "sha256:3f439d7d71adb0a0c8e05257c091236ab00c6343bc44388d091450ff58664bf9", 1
  "name": { 2
    "registry": "image-registry.openshift-image-registry.svc:5000/", 3
    "remote": "default/image-stream",
    "tag": "latest",
    "fullName": "image-registry.openshift-image-registry.svc:5000/default/image-stream:latest" 6
  },
  [...]
}
```

1 A unique identifier for the image that serves as a fingerprint for the image. It helps ensure the integrity and authenticity of the image.
7.2. ROXCTL IMAGE SCAN COMMAND OPTIONS

The `roxctl image scan` command supports the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--cluster string</code></td>
<td>Delegate image scanning to a specific cluster.</td>
</tr>
<tr>
<td><code>--compact-output</code></td>
<td>Print the JSON output in a compact format. The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>-f, --force</code></td>
<td>Ignore Central’s cache for the scan and force a fresh re–pull from Scanner. The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>--headers strings</code></td>
<td>Print the headers in a tabular format. Default values include <code>COMPONENT,VERSION,CVE,SEVERITY, and LINK</code>.</td>
</tr>
<tr>
<td><code>--headers-as-comments</code></td>
<td>Print the headers as comments in a CSV tabular output. The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>-h, --help</code></td>
<td>View the help text for the <code>roxctl image scan</code> command.</td>
</tr>
<tr>
<td><code>-i, --image string</code></td>
<td>Specify the image name and reference you want to scan.</td>
</tr>
<tr>
<td><code>-a, --include-snoozed</code></td>
<td>Return both snoozed and unsnoozed common vulnerabilities and exposures (CVEs). The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>--merge-output</code></td>
<td>Merge duplicate cells in a tabular output. The default value is <code>true</code>.</td>
</tr>
<tr>
<td><code>--no-header</code></td>
<td>Do not print headers for tabular format. The default value is <code>false</code>.</td>
</tr>
</tbody>
</table>
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-o, --output string</code></td>
<td>Specify the output format. You can select a format to customize the display of results. Formats include <code>table, CSV, JSON, and SARIF</code>.</td>
</tr>
<tr>
<td><code>-r, --retries int</code></td>
<td>Set the number of retries before the operation is aborted with an error. The default value is 3.</td>
</tr>
<tr>
<td><code>-d, --retry-delay int</code></td>
<td>Set the time in seconds to wait between retries. The default value is 3.</td>
</tr>
<tr>
<td><code>--row-jsonpath-expressions string</code></td>
<td>Use the JSON path expressions to create rows from the JSON object. For more details, run the <code>roxctl image scan --help</code> command.</td>
</tr>
</tbody>
</table>