Red Hat Advanced Cluster Security for Kubernetes 4.4

Operating

Operating Red Hat Advanced Cluster Security for Kubernetes

Last Updated: 2024-04-22

Red Hat Advanced Cluster Security for Kubernetes 4.4 Operating

Operating Red Hat Advanced Cluster Security for Kubernetes

Abstract

This document describes how to perform common operating tasks in Red Hat Advanced Cluster Security for Kubernetes, including using the dashboards, managing compliance, evaluating security risks, managing security policies and network policies, examining images for vulnerabilities, and responding to violations.

Table of Contents

CHAPTER 1. VIEWING THE DASHBOARD 9
1.1. STATUS BAR 9
1.2. DASHBOARD FILTER 9
1.3. WIDGET OPTIONS 9
1.4. ACTIONABLE WIDGETS 10
1.4.1. Policy violations by severity 10
1.4.2. Images at most risk 10
1.4.3. Deployments at most risk 10
1.4.4. Aging images 10
1.4.5. Policy violations by category 11
1.4.6. Compliance by standard 11

CHAPTER 2. USING THE COMPLIANCE OPERATOR 12
2.1. USING THE COMPLIANCE OPERATOR WITH RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES 12
2.1.1. Installing the Compliance Operator 12
2.1.2. Restarting Sensor in the secured cluster 12
2.1.3. Configuring the ScanSettingBinding object 13

CHAPTER 3. MANAGING COMPLIANCE 15
3.1. MANAGING THE COMPLIANCE 1.0 FEATURE 15
3.1.1. Viewing the compliance dashboard 15
3.1.2. Running a compliance scan 15
3.1.3. Viewing compliance scan results 17
Compliance status 17
3.1.3.1. Viewing compliance status for clusters 17
3.1.3.2. Viewing compliance status for namespaces 18
3.1.3.3. Viewing compliance status for a specific standard 18
3.1.3.4. Viewing compliance status for a specific control 19
3.1.4. Filtering compliance status 19
3.1.5. Generating compliance reports 19
3.1.5.1. Evidence reports 20
3.1.6. Supported benchmark versions 21
3.2. MANAGING THE COMPLIANCE 2.0 FEATURE (TECHNOLOGY PREVIEW) 21
3.2.1. Viewing the cluster compliance page 22
3.2.2. Cluster compliance page overview 22
3.2.3. Creating a compliance scan schedule 22
3.2.4. Viewing the compliance scan status 23
3.2.5. Compliance scan status overview 23

CHAPTER 4. EVALUATING SECURITY RISKS 25
4.1. RISK VIEW 25
4.1.1. Opening the risk view 25
4.2. CREATING A SECURITY POLICY FROM THE RISK VIEW 25
4.2.1. Understanding how Red Hat Advanced Cluster Security for Kubernetes transforms the filtering criteria into policy criteria 26
4.3. VIEWING RISK DETAILS 29
4.3.1. Risk Indicators tab 29
4.4. DEPLOYMENT DETAILS TAB 30
4.4.1. Overview section 30
4.4.2. Container configuration section 30
4.4.3. Security context section 31

4.5. PROCESS DISCOVERY TAB 31
4.5.1. Event timeline section 31
4.6. USING PROCESS BASELINES 33
Process baselines 33
Process baseline states 33
4.6.1. Viewing the process baselines 33
4.6.2. Adding a process to the baseline 34
4.6.3. Removing a process from the baseline 34
4.6.4. Locking and unlocking the process baselines 34

CHAPTER 5. USING ADMISSION CONTROLLER ENFORCEMENT 36
5.1. UNDERSTANDING ADMISSION CONTROLLER ENFORCEMENT 36
5.2. ENABLING ADMISSION CONTROLLER ENFORCEMENT 37
5.3. BYPASSING ADMISSION CONTROLLER ENFORCEMENT 38
5.4. DISABLING ADMISSION CONTROLLER ENFORCEMENT 38
5.4.1. Disabling associated policies 39
5.4.2. Disabling the webhook 39
5.5. VALIDATINGWEBHOOKCONFIGURATION YAML FILE CHANGES 40
If Central or Sensor is unavailable 40
Make the admission controller more reliable 41
Using with the roxctl CLI 41

CHAPTER 6. MANAGING SECURITY POLICIES 42
6.1. USING DEFAULT SECURITY POLICIES 42
6.2. MODIFYING EXISTING SECURITY POLICIES 43
6.3. CREATING AND MANAGING POLICY CATEGORIES 43
6.3.1. Creating policy categories by using the Policy categories tab 43
6.3.2. Modifying policy categories by using the Policy categories tab 44
6.4. CREATING CUSTOM POLICIES 44
6.4.1. Creating a security policy from the system policies view 45
6.4.1.1. Security policy enforcement for the deploy stage 47
6.4.1.1.1. Hard enforcement 48
6.4.1.1.2. Soft enforcement 48
6.4.1.1.3. Namespace exclusions 48
6.4.1.1.4. Enforcement on existing deployments 49
6.4.2. Creating a security policy from the risk view 49
6.4.3. Policy criteria 49
6.4.3.1. Adding logical conditions for the policy criteria 70
6.5. SHARING SECURITY POLICIES 71
6.5.1. Exporting a security policy 71
6.5.2. Importing a security policy 71

CHAPTER 7. DEFAULT SECURITY POLICIES 73
7.1. CRITICAL SEVERITY SECURITY POLICIES 73
7.2. HIGH SEVERITY SECURITY POLICIES 74
7.3. MEDIUM SEVERITY SECURITY POLICIES 77
7.4. LOW SEVERITY SECURITY POLICIES 83

CHAPTER 8. MANAGING NETWORK POLICIES 86
8.1. NETWORK GRAPH 86
8.1.1. About the network graph 86
8.1.1.1. Displays, navigation, and the user interface in the network graph 87
8.1.1.2. External entities and connections in the network graph 89
8.1.1.3. Connections involving internal entities 90

8.1.2. Access control and permissions 91
8.1.3. Viewing deployment information 91
8.1.4. Viewing network policies in the network graph 91
8.1.5. Configuring CIDR blocks in the network graph 92
8.2. USING THE NETWORK GRAPH TO GENERATE AND SIMULATE NETWORK POLICIES 93
8.2.1. About generating policies from the network graph 93
8.2.2. Generating network policies in the network graph 93
8.2.3. Saving generated policies in the network graph 95
8.2.4. Testing generated policies in the network graph 95
8.2.5. Reverting to a previously applied policy in the network graph 95
8.2.6. Deleting all policies autogenerated in the network graph 96
8.2.7. Simulating network policies from the network graph 96
8.3. ABOUT NETWORK BASELINING IN THE NETWORK GRAPH 97
8.3.1. Viewing network baselines from the network graph 98
8.3.2. Downloading network baselines from the network graph 98
8.3.3. Configuring network baselining time frame 98
8.3.4. Enabling alerts on baseline violations in the network graph 99

CHAPTER 9. BUILD-TIME NETWORK POLICY TOOLS 101
9.1. USING THE BUILD-TIME NETWORK POLICY GENERATOR 101
9.1.1. Generating build-time network policies 101
9.1.2. roxctl netpol generate command options 103
9.2. CONNECTIVITY MAPPING USING THE ROXCTL NETPOL CONNECTIVITY MAP COMMAND 103
9.2.1. Retrieving connectivity mapping information from a Kubernetes manifest directory 104
9.2.2. Connectivity map output formats and visualizations 104
9.2.3. Generating svg graphs from the dot output using Graphviz 104
9.2.4. roxctl netpol connectivity map command options 105
9.3. IDENTIFYING THE DIFFERENCES IN ALLOWED CONNECTIONS BETWEEN PROJECT VERSIONS 105
9.3.1. Generating connectivity difference reports with the roxctl netpol connectivity diff command 106
9.3.2. roxctl netpol connectivity diff command options 107
9.3.3. Distinguishing between syntactic and semantic difference outputs 107
9.3.3.1. Syntactic difference output 108
9.3.3.2. Semantic difference output 108

CHAPTER 10. AUDITING LISTENING ENDPOINTS 110

CHAPTER 11. REVIEWING CLUSTER CONFIGURATION 111
11.1. USING THE CONFIGURATION MANAGEMENT VIEW 111
11.2. IDENTIFYING MISCONFIGURATIONS IN KUBERNETES ROLES 111
11.2.1. Finding Kubernetes roles and their assignment 112
11.2.2. Finding service accounts and their permissions 112
11.2.3. Finding unused Kubernetes roles 112
11.3. VIEWING KUBERNETES SECRETS 112
11.4. FINDING POLICY VIOLATIONS 113
11.5. FINDING FAILING CIS CONTROLS 113

CHAPTER 12. EXAMINING IMAGES FOR VULNERABILITIES 115
12.1. ABOUT RHACS SCANNER V4 (TECHNOLOGY PREVIEW) 116
12.2. SCANNING IMAGES 117
12.2.1. Understanding and addressing common Scanner warning messages 118
12.2.2. Supported package formats 119
12.2.3. Supported programming languages 119
12.2.4. Supported runtimes and frameworks 120
12.2.5. Supported operating systems 120

12.3. ACCESSING DELEGATED IMAGE SCANNING 121
12.3.1. Enhancing image scanning by accessing delegated image scanning 121
12.3.2. Configuring delegated image scanning 122
12.3.3. Installing and configuring Scanner-slim on secured clusters 123
12.3.3.1. Using the Operator 123
12.3.3.2. Using Helm 123
12.3.3.3. Verifying after installation 123
12.3.3.4. Using image scanning 123
12.4. SETTING UP SCANNING 124
12.4.1. Automatic scanning of active images 124
12.4.2. Scanning inactive images 124
12.5. ABOUT VULNERABILITIES 125
12.5.1. Fetching vulnerability definitions 125
12.5.2. Understanding vulnerability scores 125
12.6. DISABLING LANGUAGE-SPECIFIC VULNERABILITY SCANNING 126
12.7. ADDITIONAL RESOURCES 126

CHAPTER 13. VERIFYING IMAGE SIGNATURES 127
13.1. CONFIGURING SIGNATURE INTEGRATION 127
13.2. USING SIGNATURE VERIFICATION IN A POLICY 127
13.3. ENFORCING SIGNATURE VERIFICATION 128

CHAPTER 14. MANAGING VULNERABILITIES 129
14.1. VULNERABILITY MANAGEMENT 129
14.1.1. Vulnerability management process 129
14.1.1.1. Performing asset assessment 129
14.1.2. Viewing vulnerabilities 130
14.1.2.1. Viewing application vulnerabilities 130
14.1.2.2. Viewing image vulnerabilities 130
14.1.2.3. Viewing workload CVEs in Vulnerability Management (2.0) 131
14.1.2.3.1. Viewing infrastructure vulnerabilities 132
14.1.2.3.2. Viewing node vulnerabilities 133
14.1.2.4. Prioritizing the vulnerabilities 133
14.1.2.5. Assessing the exposure 133
14.1.2.6. Taking action 134
14.1.2.6.1. Finding a new component version 134
14.1.2.7. Accepting risks 134
14.1.2.7.1. Marking vulnerabilities as false positive 135
14.1.2.7.2. Reviewing a false positive or deferred CVE 136
14.1.2.8. Reporting vulnerabilities to teams 136
14.1.3. Vulnerability reporting 136
14.1.3.1. Creating vulnerability management report configurations 136
14.1.3.1.1. Configuring delivery destinations and scheduling 137
14.1.3.1.2. Reviewing and creating the report configuration 138
14.1.3.2. Vulnerability report permissions 138
14.1.3.3. Editing vulnerability report configurations 138
14.1.3.4. Downloading vulnerability reports 139
14.1.3.5. Sending vulnerability reports on-demand 140
14.1.3.6. Cloning vulnerability report configurations 140
14.1.3.7. Deleting vulnerability report configurations 140
14.1.3.8. Configuring vulnerability management report job retention settings 140
14.1.3.9. Migration of vulnerability reports when upgrading to RHACS version 4.3 and later 141
14.1.4. Additional resources 142

14.2. COMMON VULNERABILITY MANAGEMENT TASKS 142
14.2.1. Finding critical CVEs impacting your infrastructure 142
14.2.2. Finding the most vulnerable image components 143
14.2.3. Identifying the container image layer that introduces vulnerabilities 143
14.2.4. Identifying Dockerfile lines in images that introduced components with CVEs 143
14.2.5. Viewing details only for fixable CVEs 144
14.2.6. Identifying the operating system of the base image 144
14.2.7. Identifying top risky objects 145
14.2.8. Identifying top riskiest images and components 145
14.2.9. Viewing the Dockerfile for an image 146
14.2.10. Disabling identifying vulnerabilities in nodes 146
14.2.11. Scanning inactive images 147
14.2.12. Creating policies to block specific CVEs 147
14.2.13. Viewing recently detected vulnerabilities 148
14.2.14. Viewing the most common vulnerabilities 148
14.2.15. Identifying deployments with most severe policy violations 149
14.2.16. Finding clusters with most Kubernetes and Istio vulnerabilities 149
14.2.17. Identifying vulnerabilities in nodes 149
14.3. SCANNING RHCOS NODE HOSTS 150
14.3.1. Enabling RHCOS node scanning 151
14.3.2. Analysis and detection 152
14.3.3. Vulnerability matching 152
14.3.4. Related environment variables 153
14.3.5. Identifying vulnerabilities in nodes 153

CHAPTER 15. RESPONDING TO VIOLATIONS 155
15.1. VIOLATIONS VIEW 155
15.1.1. Marking violations as resolved 155
15.2. VIEWING VIOLATION DETAILS 156
15.2.1. Violation tab 156
15.2.2. Deployment tab 156
Overview section 156
Container configuration section 156
Port configuration section 157
Security context section 158
Network policy section 158
15.2.3. Policy tab 158
Policy overview section 158
Policy behavior 158
Policy criteria section 159
15.2.3.1. Security policy enforcement for the deploy stage 159
15.2.3.1.1. Hard enforcement 159
15.2.3.1.2. Soft enforcement 160
15.2.3.1.3. Namespace exclusions 160
15.2.3.1.4. Enforcement on existing deployments 160

CHAPTER 16. CREATING AND USING DEPLOYMENT COLLECTIONS 161
16.1. PREREQUISITES 161
16.2. UNDERSTANDING DEPLOYMENT COLLECTIONS 161
16.3. ACCESSING DEPLOYMENT COLLECTIONS 164
16.4. CREATING DEPLOYMENT COLLECTIONS 164
16.4.1. Creating collection rules 165
16.4.2. Adding attached collections 166

16.5. MIGRATION OF ACCESS SCOPES TO COLLECTIONS 166
16.6. MANAGING COLLECTIONS BY USING THE API 167

CHAPTER 17. SEARCHING AND FILTERING 168
17.1. SEARCH SYNTAX 168
17.2. SEARCH AUTOCOMPLETE 169
17.3. USING GLOBAL SEARCH 169
17.4. USING LOCAL PAGE FILTERING 170
17.5. COMMON SEARCH QUERIES 170
Finding deployments that are affected by a specific CVE 170
Finding privileged running deployments 170
Finding deployments that have external network exposure 170
Finding deployments that are running specific processes 170
Finding deployments that have serious but fixable vulnerabilities 171
Finding deployments that use passwords exposed through environment variables 171
Finding running deployments that have particular software components in them 171
Finding users or groups 171
Finding who owns a particular deployment 171
Finding who is deploying images from public registries 171
Finding who is deploying into the default namespace 171
17.6. SEARCH ATTRIBUTES 171

CHAPTER 18. MANAGING USER ACCESS 177
18.1. MANAGING RBAC IN RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES 177
18.1.1. System roles 177
18.1.1.1. Viewing the permission set and access scope for a system role 178
18.1.1.2. Creating a custom role 178
18.1.1.3. Assigning a role to a user or a group 179
18.1.2. System permission sets 179
18.1.2.1. Viewing the permissions for a system permission set 180
18.1.2.2. Creating a custom permission set 180
18.1.3. System access scopes 181
18.1.3.1. Viewing the details for a system access scope 182
18.1.3.2. Creating a custom access scope 182
18.1.4. Resource definitions 183
18.1.5. Declarative configuration for authentication and authorization resources 186
18.2. ENABLING PKI AUTHENTICATION 186
18.2.1. Configuring PKI authentication by using the RHACS portal 187
18.2.2. Configuring PKI authentication by using the roxctl CLI 187
18.2.3. Updating authentication keys and certificates 188
18.2.4. Logging in by using a client certificate 188
18.3. UNDERSTANDING AUTHENTICATION PROVIDERS 188
18.3.1. Claim mappings 189
18.3.1.1. OIDC default claim mappings 189
18.3.1.2. Auth0 default claim mappings 189
18.3.1.3. SAML 2.0 default claim mappings 189
18.3.1.4. Google IAP default claim mappings 190
18.3.1.5. User certificates default claim mappings 190
18.3.1.6. OpenShift Auth default claim mappings 190
18.3.2. Rules 190
18.3.3. Minimum access role 190
18.3.4. Required attributes 191
18.4. CONFIGURING IDENTITY PROVIDERS 191

18.4.1. Configuring Okta Identity Cloud as a SAML 2.0 identity provider 191
18.4.1.1. Creating an Okta app 191
18.4.1.2. Configuring a SAML 2.0 identity provider 192
18.4.2. Configuring Google Workspace as an OIDC identity provider 194
18.4.2.1. Setting up OAuth 2.0 credentials for your GCP project 194
18.4.2.2. Specifying a client secret 195
18.4.2.3. Configuring an OIDC identity provider 196
18.4.3. Configuring OpenShift Container Platform OAuth server as an identity provider 198
18.4.3.1. Configuring OpenShift Container Platform OAuth server as an identity provider 198
18.4.3.2. Creating additional routes for OpenShift Container Platform OAuth server 200
18.4.4. Connecting Azure AD to RHACS using SSO configuration 202
18.4.4.1. Adding group claims to tokens for SAML applications using SSO configuration 202
18.5. CONFIGURING SHORT-LIVED ACCESS 202
18.5.1. Configure short-lived access for an OIDC identity token issuer 203
18.5.2. Exchanging an identity token 204

CHAPTER 19. USING THE SYSTEM HEALTH DASHBOARD 206
19.1. SYSTEM HEALTH DASHBOARD DETAILS 206
Cluster health section 206
Vulnerabilities definition section 207
Integrations section 207
19.2. VIEWING PRODUCT USAGE DATA 207
19.3. GENERATING A DIAGNOSTIC BUNDLE BY USING THE RHACS PORTAL 208
19.3.1. Additional resources 209

CHAPTER 20. USING THE ADMINISTRATION EVENTS PAGE 210
20.1. VIEWING THE ADMINISTRATION EVENTS PAGE 210
20.2. UNDERSTANDING THE ADMINISTRATION EVENTS PAGE 210
20.3. VIEWING THE ADMINISTRATION EVENT DETAILS 210
20.4. UNDERSTANDING THE ADMINISTRATION EVENT DETAILS 210
20.5. CONFIGURING ADMINISTRATION EVENTS RETENTION SETTINGS 211

CHAPTER 1. VIEWING THE DASHBOARD

The Red Hat Advanced Cluster Security for Kubernetes (RHACS) Dashboard provides quick access to the data you need. It contains additional navigation shortcuts and actionable widgets that are easy to filter and customize so that you can focus on the data that matters most to you. You can view information about levels of risk in your environment, compliance status, policy violations, and common vulnerabilities and exposures (CVEs) in images.

NOTE

When you open the RHACS portal for the first time, the Dashboard might be empty. After you deploy Sensor in at least one cluster, the Dashboard reflects the status of your environment.

The following sections describe the Dashboard components.

1.1. STATUS BAR

The **Status Bar** provides at-a-glance numerical counters for key resources. The counters reflect what is visible with your current access scope that is defined by the roles associated with your user profile. These counters are clickable, providing fast access to desired list view pages as follows:

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

1.2. DASHBOARD FILTER

The Dashboard includes a top-level filter that applies simultaneously to all widgets. You can select one or more clusters, and one or more namespaces within selected clusters. When no clusters or namespaces are selected, the view automatically switches to **All**. Any change to the filter is immediately reflected by all widgets, limiting the data they present to the selected scope. The Dashboard filter does not affect the **Status Bar**.

1.3. WIDGET OPTIONS

Some widgets are customizable to help you focus on specific data. Widgets offer different controls that you can use to change how the data is sorted, filter the data, and customize the output of the widget.

Widgets offer two ways to customize different aspects:

- An **Options** menu, when present, provides specific options applicable to that widget.
- A **dynamic axis legend**, when present, provides a method to filter data by hiding one or more of the axis categories. For example, in the **Policy violations by category** widget, you can click on a severity to include or exclude violations of a selected severity from the data.

NOTE

Individual widget customization settings are short-lived and are reset to the system default upon leaving the Dashboard.

1.4. ACTIONABLE WIDGETS

The following sections describe the actionable widgets available in the Dashboard.

1.4.1. Policy violations by severity

This widget shows the distribution of violations across severity levels for the Dashboard-filtered scope. Clicking a **severity level** in the chart takes you to the **Violations** page, filtered for that severity and scope. It also lists the three most recent violations of a **Critical** level policy within the scope you defined in the Dashboard filter. Clicking a specific violation takes you directly to the **Violations** detail page for that violation.

1.4.2. Images at most risk

This widget lists the top six vulnerable images within the Dashboard-filtered scope, sorted by their computed risk priority, along with the number of critical and important CVEs they contain. Click on an image name to go directly to the **Image Findings** page under **Vulnerability Management**. Use the **Options** menu to focus on fixable CVEs, or further focus on active images.

NOTE

When clusters or namespaces have been selected in the Dashboard filter, the data displayed is already filtered to active images, or images that are used by deployments within the filtered scope.

1.4.3. Deployments at most risk

This widget provides information about the top deployments at risk in your environment. It displays additional information such as the resource location (cluster and namespace) and the risk priority score. Additionally, you can click on a deployment to view risk information about the deployment; for example, its policy violations and vulnerabilities.

1.4.4. Aging images

Older images present a higher security risk because they can contain vulnerabilities that have already been addressed. If older images are active, they can expose deployments to exploits. You can use this widget to quickly assess your security posture and identify offending images. You can use the default ranges or customize the age intervals with your own values. You can view both inactive and active images or use the Dashboard filter to focus on a particular area for active images. You can then click on an age group in this widget to view only those images in the **Vulnerability Management → Images** page.

1.4.5. Policy violations by category

This widget can help you gain insights about the challenges your organization is facing in complying with security policies, by analyzing which types of policies are violated more than others. The widget shows the five policy categories of highest interest. Explore the **Options** menu for different ways to slice the data. You can filter the data to focus exclusively on deploy or runtime violations.

You can also change the sorting mode. By default, the data is sorted by the number of violations within the highest severity first. Therefore, all categories with critical policies will appear before categories without critical policies. The other sorting mode considers the total number of violations regardless of severity. Because some categories contain no critical policies (for example, “Docker CIS”), the two sorting modes can provide significantly different views, offering additional insight.

Click on a severity level at the bottom of the graph to include or exclude that level from the data. Selecting different severity levels can result in a different top five selection or ranking order. Data is filtered to the scope selected by the Dashboard filter.

1.4.6. Compliance by standard

You can use the **Compliance by standard** widget with the Dashboard filter to focus on areas that matter to you the most. The widget lists the top or bottom six compliance benchmarks, depending on sort order. Select **Options** to sort by the coverage percentage. Click on one of the benchmark labels or graphs to go directly to the **Compliance Controls** page, filtered by the Dashboard scope and the selected benchmark.

NOTE

The **Compliance** widget shows details only after you run a compliance scan.

CHAPTER 2. USING THE COMPLIANCE OPERATOR

2.1. USING THE COMPLIANCE OPERATOR WITH RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES

You can configure RHACS to use the Compliance Operator for compliance reporting and remediation with OpenShift Container Platform clusters. Results from the Compliance Operator can be reported in the RHACS Compliance Dashboard.

2.1.1. Installing the Compliance Operator

Install the Compliance Operator using Operator Hub.

IMPORTANT

If you install the Compliance Operator after Sensor is fully operational, you must restart Sensor in the secured cluster.

For more information about restarting Sensor, see "Restarting Sensor in the secured cluster" in the "Additional resources" section.

Procedure

1. In the web console, go to the **Operators → OperatorHub** page.
2. Enter **compliance operator** into the **Filter by keyword** box to find the Compliance Operator.
3. Select the **Compliance Operator** to view the details page.
4. Read the information about the Operator, and then click **Install**.

Next steps

- Configuring the ScanSettingBinding object

Additional resources

- Restarting Sensor in the secured cluster

2.1.2. Restarting Sensor in the secured cluster

If you installed the Compliance Operator before installing RHACS, you need to restart Sensor in the secured cluster either by using the command-line interface (CLI) or the user interface (UI).

Procedure

- To restart Sensor from the CLI, run the following command:

```
$ oc -n stackrox delete pod -lapp=sensor
```

- To restart Sensor from the UI, perform the following steps:
    1. Change the active project to **stackrox**.

2. Go to **Workloads → Pods**.

3. Locate the pod with the name starting with **sensor-**, and then click **Actions → Delete Pod**.

2.1.3. Configuring the ScanSettingBinding object

Create a **ScanSettingBinding** object in the **openshift-compliance** namespace to scan the cluster by using the **cis** and **cis-node** profiles.

IMPORTANT

- If you are using the compliance 2.0 feature, you can schedule your scan by using RHACS to create a compliance scan schedule instead of creating the **ScanSettingBinding** on the Compliance Operator.
  For more information about scheduling a compliance scan by using the compliance 2.0 feature, see "Creating a compliance scan schedule" in the "Additional resources” section.

- This example uses **ocp4-cis** and **ocp4-cis-node** profiles, but OpenShift Container Platform provides additional profiles. See "Understanding the Compliance Operator" in the "Additional resources" section for more information.

IMPORTANT

Compliance 2.0 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 .

Procedure

Select one of the following options:

- Use the CLI to create the YAML file and object. For example:

  a. Create a file called **sscan.yaml** using the following text:

```
apiVersion: compliance.openshift.io/v1alpha1
kind: ScanSettingBinding
metadata:
  name: cis-compliance
profiles:
  - name: ocp4-cis-node
    kind: Profile
    apiGroup: compliance.openshift.io/v1alpha1
  - name: ocp4-cis
    kind: Profile
    apiGroup: compliance.openshift.io/v1alpha1
settingsRef:
  name: default
  kind: ScanSetting
  apiGroup: compliance.openshift.io/v1alpha1
```

b. Create the **ScanSettingBinding** object by running the following command:

```
$ oc create -f sscan.yaml -n openshift-compliance
```

If successful, the following message is displayed:

```
$ scansettingbinding.compliance.openshift.io/cis-compliance created
```

- Use the web console to create the object by performing the following steps:

  a. Change the active project to **openshift-compliance**.

  b. Click **+** to open the **Import YAML** page.

  c. Paste the YAML from the previous example and then click **Create**.

Verification

1. Run a compliance scan in RHACS.
   For more information about running a compliance scan by using the compliance 1.0 feature, see "Running a compliance scan" in the "Additional resources" section.

2. Ensure that **ocp4-cis** and **ocp4-cis-node** results are displayed.

Additional resources

- Understanding the Compliance Operator
- Compliance Operator scans
- Running a compliance scan in RHACS
- Creating a compliance scan schedule

CHAPTER 3. MANAGING COMPLIANCE

3.1. MANAGING THE COMPLIANCE 1.0 FEATURE

By using Red Hat Advanced Cluster Security for Kubernetes you can assess, check, and report on the compliance status of your containerized infrastructure. You can run out-of-the-box compliance scans based on industry standards including:

- **CIS Benchmarks** (Center for Internet Security) for **Docker** and **Kubernetes**
- **HIPAA** (Health Insurance Portability and Accountability Act)
- **NIST Special Publication 800-190**and **800-53** (National Institute of Standards and Technology)
- **PCI DSS** (Payment Card Industry Data Security Standard)
- **OpenSCAP** (Open Security Content Automation Protocol): Available in RHACS for OpenShift Container Platform clusters when the Compliance Operator is installed and configured to provide results to RHACS

By scanning your environment based on these standards you can:

- Evaluate your infrastructure for regulatory compliance.
- Harden your Docker Engine and Kubernetes orchestrator.
- Understand and manage the overall security posture of your environment.
- Get a detailed view of compliance status for clusters, namespaces, and nodes.

3.1.1. Viewing the compliance dashboard

The compliance dashboard provides a high-level view of the compliance standards across all clusters, namespaces, and nodes in your environment.

The compliance dashboard includes charts and provides options to investigate a potential problem with compliance mandates. You can go to compliance scan results for a single cluster, namespace, or a node. Moreover, you can generate reports on the state of compliance within your containerized environment.

Procedure

- In the RHACS portal, select **Compliance (1.0)** from the navigation menu.

NOTE

The first time you open the Compliance dashboard you will see a blank dashboard. You must run a compliance scan to populate the dashboard.

3.1.2. Running a compliance scan

Running a compliance scan checks the compliance status for your entire infrastructure across all compliance standards. When you run a compliance scan, Red Hat Advanced Cluster Security for Kubernetes takes a data snapshot of your environment. The data snapshot includes alerts, images, network policies, deployments, and related host-based data. Central collects the host-based data from the Sensors running in your clusters. After that, Central collects more data from the compliance container running in each collector pod. The compliance container collects the following data about your environment:

- Configurations for Docker Daemon, Docker image, and Docker container.
- Information about Docker networks.
- Command-line arguments and processes for Docker, Kubernetes, and OpenShift Container Platform.
- Permissions of specific file paths.
- Configuration files for the core Kubernetes and OpenShift Container Platform services.

After the data collection is complete, Central performs checks on the data to determine results. You can view the results from the compliance dashboard and also generate compliance reports based on the results.

NOTE

In a compliance scan:

- **Control** describes a single line item in an industry or regulatory compliance standard against which an auditor evaluates an information system for compliance with said standard. Red Hat Advanced Cluster Security for Kubernetes checks the evidence of compliance with a single control by completing one or more checks.
- **Check** is the single test performed during a single control assessment.
- Some controls have multiple checks associated with them. If any of the associated check fails for a control, the entire control state is marked as **Fail**.

Procedure

1. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
2. Optional: By default, information under all standards is displayed in the compliance results. To view only information from specific standards, perform the following steps:
   a. Click **Manage standards**.
   b. By default, all standards are selected. Clear the checkbox for any specific standard that you do not want to display, and then click **Save**. Standards that are not selected do not appear in the dashboard display (including widgets), compliance results tables that are accessed from the dashboard, and PDF files created using the **Export** button. However, all default standards are included when results are exported as a CSV file.
3. Click **Scan environment**.

NOTE

Scanning the entire environment takes about 2 minutes to complete. This time might vary depending on the number of clusters and nodes in your environment.

Verification

1. In the RHACS portal, go to **Configuration Management**.
2. In the **CIS Kubernetes v1.5** widget, click **Scan.**
3. RHACS displays a message which indicates that a compliance scan is in progress.

3.1.3. Viewing compliance scan results

After you run a compliance scan, the compliance dashboard displays the results as the compliance status for your environment. You can view compliance violations directly from the dashboard, filter the details view, and drill down compliance standards to understand if your environment is compliant against specific benchmarks. This section explains how to view and filter compliance scan results.

You can use shortcuts to check the compliance status of clusters, namespaces, and nodes. Look for these shortcuts on the top of your compliance dashboard. By clicking these shortcuts you can view the compliance snapshot and generate reports on the overall compliance of your clusters, namespaces, or nodes.

Compliance status

Status	Description
Fail	The compliance check failed.
Pass	The compliance check passed.
N/A	Red Hat Advanced Cluster Security for Kubernetes skipped the check because it was not applicable.
Info	The compliance check gathered data, but Red Hat Advanced Cluster Security for Kubernetes could not make a **Pass** or **Fail** determination.
Error	The compliance check failed due to a technical issue.

3.1.3.1. Viewing compliance status for clusters

You can view compliance status for all clusters or a single cluster from the compliance dashboard.

Procedure

- To view compliance status for all clusters in your environment:
  a. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
  b. Click **Clusters** on the compliance dashboard.

- To view compliance status for a specific cluster in your environment:
    a. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
    b. On the compliance dashboard, look for the **Passing standards by cluster** widget.
    c. In this widget, click on a cluster name to view its compliance status.

3.1.3.2. Viewing compliance status for namespaces

You can view compliance status for all namespaces or a single namespace from the compliance dashboard.

Procedure

- To view compliance status for all namespaces in your environment:
    1. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
    2. Click **Namespaces** on the compliance dashboard.
- To view compliance status for a specific namespace in your environment:
    1. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
    2. Click **Namespaces** to open the namespaces details page.
    3. From the **Namespaces** table, click on a namespace. A side panel opens on the right.
    4. In the side panel, click on the name of the namespace to view its compliance status.

3.1.3.3. Viewing compliance status for a specific standard

Red Hat Advanced Cluster Security for Kubernetes supports NIST, PCI DSS, NIST, HIPAA, CIS for Kubernetes and CIS for Docker compliance standards. You can view all the compliance controls for a single compliance standard.

Procedure

1. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
2. On the compliance dashboard, look for the **Passing standards across clusters cluster** widget.
3. In this widget, click on a standard to view information about all the controls associated with that standard.

NOTE

Many of the controls in CIS Docker refer to the configuration of the Docker engine on each Kubernetes node. Many CIS Docker controls are also best practices for building and using containers, and RHACS has policies to enforce their use. See "Managing security policies" in "Additional resources" for more information.

Additional resources

- Managing security policies

3.1.3.4. Viewing compliance status for a specific control

You can view compliance status for a specific control for a selected standard.

Procedure

1. In the RHACS portal, go to **Compliance (1.0)**.
2. On the compliance dashboard, look for the **Passing standards by cluster** widget.
3. In this widget, click on a standard to view information about all the controls associated with that standard.
4. From the **Controls** table, click on a control. A side panel opens on the right.
5. In the side panel, click on the name of the control to view its details.

3.1.4. Filtering compliance status

Red Hat Advanced Cluster Security for Kubernetes search makes it easy to filter different combinations of data from the compliance dashboard. To focus your attention on a subset of clusters, industry standards, passing or failing controls, you can narrow the scope of the data visible on the compliance dashboard.

Procedure

1. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
2. On the compliance dashboard, select either **Clusters**, or **Namespaces**, or **Nodes** to open the details page.
3. Enter your filtering criteria in the search bar and then press **Enter**.

3.1.5. Generating compliance reports

Red Hat Advanced Cluster Security for Kubernetes enables you to generate reports to keep track of the compliance status of your environment. You can use these reports to convey compliance status across various industry mandates to other stakeholders.

You can generate:

- **Executive reports** that focuses on the business aspect and includes charts and summary of compliance status in PDF format.
- **Evidence reports** that focuses on the technical aspect and includes detailed information in CSV format.

Procedure

1. Go to the RHACS portal and open the compliance dashboard by selecting **Compliance (1.0)** from the navigation menu.
2. On the compliance dashboard, click **Export**.
   - To generate an executive report, select **Download page as PDF**.
   - To generate an evidence report, select **Download Evidence as CSV**.

TIP

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

3.1.5.1. Evidence reports

You can export comprehensive compliance-related data from Red Hat Advanced Cluster Security for Kubernetes in CSV format as an evidence report. This evidence report contains detailed information about the compliance assessment, and it is tailored towards technical roles, such as compliance auditors, DevOps engineers, or security practitioners.

An evidence report contains the following information:

CSV field	Description
Standard	The compliance standard, for example, CIS Kubernetes.
Cluster	The name of the assessed cluster.
Namespace	The name of the namespace or project where the deployment exists.
Object Type	The Kubernetes entity type of the object. For example, **node**, **cluster**, **DaemonSet**, **Deployment**, or **StaticPod**.
Object Name	The name of the object which is a Kubernetes systems-generated string that uniquely identify objects. For example, **gke-setup-dev21380-default-pool-8e086a77-1jfq**.
Control	The control number as it appears in the compliance standard.
Control Description	Description about the compliance check that the control carries out.
State	Whether the compliance check passed or failed. For example, **Pass** or **Fail**.
Evidence	The explanation about why a specific compliance check failed or passed.

CSV field	Description
Assessment Time	The time and date when you ran the compliance scan.

3.1.6. Supported benchmark versions

Red Hat Advanced Cluster Security for Kubernetes supports compliance checks against the following industry standards and regulatory frameworks:

Benchmark	Supported version
CIS Benchmarks (Center for Internet Security) for Docker and Kubernetes	CIS Kubernetes v1.5.0 and CIS Docker v1.2.0
HIPAA (Health Insurance Portability and Accountability Act)	HIPAA 164
NIST (National Institute of Standards and Technology)	NIST Special Publication 800-190 and 800-53 Rev. 4
PCI DSS (Payment Card Industry Data Security Standard)	PCI DSS 3.2.1

3.2. MANAGING THE COMPLIANCE 2.0 FEATURE (TECHNOLOGY PREVIEW)

IMPORTANT

Compliance 2.0 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 .

You can view the compliance results associated with your cluster by using the compliance 2.0 feature in the Red Hat Advanced Cluster Security for Kubernetes (RHACS) portal. The feature collects compliance information gathered by the Compliance Operator into a single interface.

For more information about using the Compliance Operator, see Using the Compliance Operator with Red Hat Advanced Cluster Security for Kubernetes.

NOTE

Currently, the compliance 2.0 feature and the Compliance Operator evaluate only infrastructure and platform compliance.

3.2.1. Viewing the cluster compliance page

By viewing the cluster compliance page, you can get a comprehensive overview of the compliance status of your clusters.

Procedure

- In the RHACS portal, go to the **Compliance (2.0) → Cluster Compliance → Coverage**tab.

3.2.2. Cluster compliance page overview

The cluster compliance page organizes information in the following groups:

- **Cluster**: Gives the details of your cluster and provides a snapshot of its current state and configurations.
- **Operator status**: Assesses the health and operational status of the Compliance Operator instance within your cluster and ensures that the Operator is running optimally and functioning seamlessly.
- **Compliance**: Shows the percentage of checks that have been passed for the scanned profiles.

3.2.3. Creating a compliance scan schedule

By creating a compliance scan schedule, you can customize and automate your compliance scans to align with your operational requirements.

Procedure

1. In the RHACS portal, go to the **Compliance (2.0) → Cluster Compliance → Schedules**tab.
2. Click **Create scan schedule**.
3. In the **Configuration options** page, provide the following information:
   - **Name**: Enter a name to identify different compliance scans.
   - **Description**: Specify the reason for each compliance scan.
   - **Configure schedule**: Adjust the scan schedule to fit your required schedule:
     - **Frequency**: From the drop-down list, select how often you want to perform the scan. Frequencies include **Daily**, **Weekly**, and **Monthly**.
     - **On day(s)**: From the list, select one or more days of the week on which you want to perform the scan. Valid values include **Monday**, **Tuesday**, **Wednesday**, **Thursday**, **Friday**, **Saturday**, **Sunday**, **The first of the month**, and **The middle of the month**.

       NOTE

       These values are only applicable if you specify the frequency of scan as **Weekly** or **Monthly**.
     - **Time**: Start to type the time in **hh:mm** at which you want to run the scan. From the list that is displayed, select a time.

4. Click **Next**.
5. In the **Clusters** page, select one or more clusters that you want to include in the scan.
6. Click **Next**.
7. In the **Profiles** page, select one or more profiles that you want to include in the scan.
8. Click **Next**.
9. Review your scan configuration, and then click **Create**.

Verification

1. In the RHACS portal, go to the **Compliance (2.0) → Cluster Compliance → Schedules**tab.
2. Select the compliance scan you have created.
3. In the **Clusters** section, verify that the operator status is healthy.
4. Optional: To edit the scan schedule, click **Edit scan schedule**, make your changes, and then click **Save.**

3.2.4. Viewing the compliance scan status

By viewing the status of a compliance scan, you can efficiently monitor and analyze the health of your clusters.

IMPORTANT

Wait until the Compliance Operator returns the scan results. It might take a few minutes.

Procedure

1. In the RHACS portal, go to the **Compliance (2.0) → Cluster Compliance → Coverage**tab.
2. Select a cluster to view the details of the individual scans.
3. Optional: Enter the name of the compliance check in the **Filter by keyword box** to view the status.
4. Optional: From the **Compliance status** drop-down list, select one or more statuses by using which you want to filter the scan details. Compliance statuses include **Pass**, **Fail**, **Error**, **Info**, **Manual**, **Not Applicable**, and **Inconsistent**.

3.2.5. Compliance scan status overview

By understanding the compliance scan status, you can manage the overall security posture of your environment.

Status	Description
Fail	The compliance check failed.

Status	Description
Pass	The compliance check passed.
Not Applicable	Skipped the compliance check because it was not applicable.
Info	The compliance check gathered data, but RHACS could not make a pass or fail determination.
Error	The compliance check failed due to a technical issue.
Manual	Manual intervention is required to ensure compliance.
Inconsistent	The compliance scan data is inconsistent, and requires closer inspection and targeted resolution.

CHAPTER 4. EVALUATING SECURITY RISKS

Red Hat Advanced Cluster Security for Kubernetes assesses risk across your entire environment and ranks your running deployments according to their security risk. It also provides details about vulnerabilities, configurations, and runtime activities that require immediate attention.

4.1. RISK VIEW

The **Risk** view lists all deployments from all clusters, sorted by a multi-factor risk metric based on policy violations, image contents, deployment configuration, and other similar factors. Deployments at the top of the list present the most risk.

The **Risk** view shows list of deployments with following attributes for each row:

- **Name**: The name of the deployment.
- **Created**: The creation time of the deployment.
- **Cluster**: The name of the cluster where the deployment is running.
- **Namespace**: The namespace in which the deployment exists.
- **Priority**: A priority ranking based on severity and risk metrics.

In the **Risk** view, you can:

- Select a column heading to sort the violations in ascending or descending order.
- Use the filter bar to filter violations.
- Create a new policy based on the filtered criteria.

To view more details about the risks for a deployment, select a deployment in the **Risk** view.

4.1.1. Opening the risk view

You can analyze all risks in the **Risk** view and take corrective action.

Procedure

- Go to the RHACS portal and select **Risk** from the navigation menu.

4.2. CREATING A SECURITY POLICY FROM THE RISK VIEW

While evaluating risks in your deployments in the **Risk** view, when you apply local page filtering, you can create new security policies based on the filtering criteria you are using.

Procedure

1. Go to the RHACS portal and select **Risk** from the navigation menu.
2. Apply local page filtering criteria that you want to create a policy for.
3. Select **New Policy** and fill in the required fields to create a new policy.

4.2.1. Understanding how Red Hat Advanced Cluster Security for Kubernetes transforms the filtering criteria into policy criteria

When you create new security policies from the **Risk** view, based on the filtering criteria you use, not all criteria are directly applied to the new policy.

- Red Hat Advanced Cluster Security for Kubernetes converts the **Cluster**, **Namespace**, and **Deployment** filters to equivalent policy scopes.
  - Local page filtering on the **Risk** view combines the search terms by using the following methods:
    - Combines the search terms within the same category with an **OR** operator. For example, if the search query is **Cluster:A,B**, the filter matches deployments in **cluster A** or **cluster B**.
    - Combines the search terms from different categories with an **AND** operator. For example, if the search query is **Cluster:A+Namespace:Z**, the filter matches deployments in **cluster A** and in **namespace Z**.
  - When you add multiple scopes to a policy, the policy matches violations from any of the scopes.
    - For example, if you search for **(Cluster A OR Cluster B) AND (Namespace Z)** it results in two policy scopes, **(Cluster=A AND Namespace=Z)** OR **(Cluster=B AND Namespace=Z)**.
- Red Hat Advanced Cluster Security for Kubernetes drops or modifies filters that do not directly map to policy criteria and reports the dropped filters.

The following table lists how the filtering search attributes map to the policy criteria:

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
Cluster	⟳ Converted to scope

Search attribute	Policy criteria
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
Image OS	Image OS
Image Pull Secret	✕ Dropped

Search attribute	Policy criteria
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
Process Name	Process Name
Process Path	✕ Dropped
Process Tag	✕ Dropped

Search attribute	Policy criteria
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

4.3. VIEWING RISK DETAILS

When you select a deployment in the **Risk** view, the **Risk Details** open in a panel on the right. The **Risk Details** panel shows detailed information grouped by multiple tabs.

4.3.1. Risk Indicators tab

The **Risk Indicators** tab of the **Risk Details** panel explains the discovered risks.

The **Risk Indicators** tab includes the following sections:

- **Policy Violations**: The names of the policies that are violated for the selected deployment.
- **Suspicious Process Executions**: Suspicious processes, arguments, and container names that the process ran in.
- **Image Vulnerabilities**: Images including total CVEs with their CVSS scores.
- **Service Configurations**: Aspects of the configurations that are often problematic, such as read-write (RW) capability, whether capabilities are dropped, and the presence of privileged containers.

- **Service Reachability**: Container ports exposed inside or outside the cluster.
- **Components Useful for Attackers** Discovered software tools that are often used by attackers.
- **Number of Components in Image**: The number of packages found in each image.
- **Image Freshness**: Image names and age, for example, **285 days old**.
- **RBAC Configuration**: The level of permissions granted to the deployment in Kubernetes role-based access control (RBAC).

NOTE

Not all sections are visible in the **Risk Indicators** tab. Red Hat Advanced Cluster Security for Kubernetes displays only relevant sections that affect the selected deployment.

4.4. DEPLOYMENT DETAILS TAB

The sections in the **Deployment Details** tab of the **Deployment Risk** panel provide more information so you can make appropriate decisions on how to address the discovered risk.

4.4.1. Overview section

The **Overview** section shows details about the following:

- **Deployment ID**: An alphanumeric identifier for the deployment.
- **Namespace**: The Kubernetes or OpenShift Container Platform namespace in which the deployment exists.
- **Updated**: A timestamp with date for when the deployment was updated.
- **Deployment Type**: The type of deployment, for example, **Deployment** or **DaemonSet**.
- **Replicas**: The number of pods deployed for this deployment.
- **Labels**: The key-value labels attached to the Kubernetes or OpenShift Container Platform application.
- **Cluster**: The name of the cluster where the deployment is running.
- **Annotations**: The Kubernetes annotations for the deployment.
- **Service Account**: Represents an identity for processes that run in a pod. When a process is authenticated through a service account, it can contact the Kubernetes API server and access cluster resources. If a pod does not have an assigned service account, it gets the default service account.

4.4.2. Container configuration section

The container configuration section shows details about the following:

- **Image Name**: The name of the image that is deployed.
- **Resources**

    - **CPU Request (cores)**: The number of CPUs requested by the container.
    - **CPU Limit (cores)**: The maximum number of CPUs the container can use.
    - **Memory Request (MB)**: The memory size requested by the container.
    - **Memory Limit (MB)**: The maximum amount of memory the container can use without being killed.
- **Mounts**
    - **Name**: The name of the mount.
    - **Source**: The path from where the data for the mount comes.
    - **Destination**: The path to which the data for the mount goes.
    - **Type**: The type of the mount.
- **Secrets**: The names of Kubernetes secrets used in the deployment, and basic details for secret values that are X.509 certificates.

4.4.3. Security context section

The **Security Context** section shows details about the following:

- **Privileged**: Lists **true** if the container is privileged.

4.5. PROCESS DISCOVERY TAB

The **Process Discovery** tab provides a comprehensive list of all binaries that have been executed in each container in your environment, summarized by deployment.

The process discovery tab shows details about the following:

- **Binary Name**: The name of the binary that was executed.
- **Container**: The container in the deployment in which the process executed.
- **Arguments**: The specific arguments that were passed with the binary.
- **Time**: The date and time of the most recent time the binary was executed in a given container.
- **Pod ID**: The identifier of the pod in which the container resides.
- **UID**: The Linux user identity under which the process executed.

Use the **Process Name:<name>** query in the filter bar to find specific processes.

4.5.1. Event timeline section

The **Event Timeline** section in the **Process Discovery** tab provides an overview of events for the selected deployment. It shows the number of policy violations, process activities, and container termination or restart events.

You can select **Event Timeline** to view more details.

The **Event Timeline** modal box shows events for all pods for the selected deployment.

The events on the timeline are categorized as:

- Process activities
- Policy violations
- Container restarts
- Container terminations

The events appear as icons on a timeline. To see more details about an event, hold your mouse pointer over the event icon. The details appear in a tooltip.

- Click **Show Legend** to see which icon corresponds to which type of event.
- Select **Export → Download PDF** or **Export → Download CSV** to download the event timeline information.
- Select the **Show All** drop-down menu to filter which type of events are visible on the timeline.
- Click on the expand icon to see events separately for each container in the selected pod.

All events in the timeline are also visible in the minimap control at the bottom. The minimap controls the number of events visible in the event timeline. You can change the events shown in the timeline by modifying the highlighted area on the minimap. To do this, decrease the highlighted area from left or right sides (or both), and then drag the highlighted area.

NOTE

- When containers restart, Red Hat Advanced Cluster Security for Kubernetes:
  - Shows information about container termination and restart events for up to 10 inactive container instances for each container in a pod. For example, for a pod with two containers **app** and **sidecar**, Red Hat Advanced Cluster Security for Kubernetes keeps activity for up to 10 **app** instances and up to 10 **sidecar** instances.
  - Does not track process activities associated with the previous instances of the container.
- Red Hat Advanced Cluster Security for Kubernetes only shows the most recent execution of each (process name, process arguments, UID) tuple for each pod.
- Red Hat Advanced Cluster Security for Kubernetes shows events only for the active pods.
- Red Hat Advanced Cluster Security for Kubernetes adjusts the reported timestamps based on time reported by Kubernetes and the Collector. Kubernetes timestamps use second-based precision, and it rounds off the time to the nearest second. However, the Collector uses more precise timestamps. For example, if Kubernetes reports the container start time as **10:54:48**, and the Collector reports a process in that container started at **10:54:47.5349823**, Red Hat Advanced Cluster Security for Kubernetes adjusts the container start time to **10:54:47.5349823**.

4.6. USING PROCESS BASELINES

You can minimize risk by using process baselining for infrastructure security. With this approach, Red Hat Advanced Cluster Security for Kubernetes first discovers existing processes and creates a baseline. Then it operates in the default deny-all mode and only allows processes listed in the baseline to run.

Process baselines

When you install Red Hat Advanced Cluster Security for Kubernetes, there is no default process baseline. As Red Hat Advanced Cluster Security for Kubernetes discovers deployments, it creates a process baseline for every container type in a deployment. Then it adds all discovered processes to their own process baselines.

Process baseline states

During the process discovery phase, all baselines are in an unlocked state.

In an **unlocked** state:

- When Red Hat Advanced Cluster Security for Kubernetes discovers a new process, it adds that process to the process baseline.
- Processes do not show up as risks and do not trigger any violations.

After an hour from when Red Hat Advanced Cluster Security for Kubernetes receives the first process indicator from a container in a deployment, it finishes the process discovery phase. At this point:

- Red Hat Advanced Cluster Security for Kubernetes stops adding processes to the process baselines.
- New processes that are not in the process baseline show up as risks, but they do not trigger any violations.

To generate violations, you must manually lock the process baseline.

In a **locked** state:

- Red Hat Advanced Cluster Security for Kubernetes stops adding processes to the process baselines.
- New processes that are not in the process baseline trigger violations.

Independent of the locked or unlocked baseline state, you can always add or remove processes from the baseline.

NOTE

For a deployment, if each pod has multiple containers in it, Red Hat Advanced Cluster Security for Kubernetes creates a process baseline for each container type. For such a deployment, if some baselines are locked and some are unlocked, the baseline status for that deployment shows up as **Mixed**.

4.6.1. Viewing the process baselines

You can view process baselines from the **Risk** view.

Procedure

1. In the RHACS portal, select **Risk** from the navigation menu.
2. Select a deployment from the list of deployments in the default **Risk** view. Deployment details open in a panel on the right.
3. In the **Deployment** details panel, select the **Process Discovery** tab.
4. The process baselines are visible under the **Spec Container Baselines** section.

4.6.2. Adding a process to the baseline

You can add a process to the baseline.

Procedure

1. In the RHACS portal, select **Risk** from the navigation menu.
2. Select a deployment from the list of deployments in the default **Risk** view. Deployment details open in a panel on the right.
3. In the **Deployment** details panel, select the **Process Discovery** tab.
4. Under the **Running Processes** section, click the **Add** icon for the process you want to add to the process baseline.

NOTE

The **Add** icon is available only for the processes that are not in the process baseline.

4.6.3. Removing a process from the baseline

You can remove a process from the baseline.

Procedure

1. In the RHACS portal, select **Risk** from the navigation menu.
2. Select a deployment from the list of deployments in the default **Risk** view. Deployment details open in a panel on the right.
3. In the **Deployment** details panel, select the **Process Discovery** tab.
4. Under the **Spec Container baselines** section, click the **Remove** icon for the process you want to remove from the process baseline.

4.6.4. Locking and unlocking the process baselines

You can **Lock** the baseline to trigger violations for all processes not listed in the baseline and **Unlock** the baseline to stop triggering violations.

Procedure

1. In the RHACS portal, select **Risk** from the navigation menu.

2. Select a deployment from the list of deployments in the default **Risk** view. Deployment details open in a panel on the right.
3. In the **Deployment** details panel, select the **Process Discovery** tab.
4. Under the **Spec Container baselines** section:
   - Click the **Lock** icon to trigger violations for processes that are not in the baseline.
   - Click the **Unlock** icon to stop triggering violations for processes that are not in the baseline.

CHAPTER 5. USING ADMISSION CONTROLLER ENFORCEMENT

Red Hat Advanced Cluster Security for Kubernetes works with Kubernetes admission controllers and OpenShift Container Platform admission plugins to allow you to enforce security policies before Kubernetes or OpenShift Container Platform creates workloads, for example, deployments, daemon sets or jobs.

The RHACS admission controller prevents users from creating workloads that violate policies you configure in RHACS. Beginning from the RHACS version 3.0.41, you can also configure the admission controller to prevent updates to workloads that violate policies.

RHACS uses the **ValidatingAdmissionWebhook** controller to verify that the resource being provisioned complies with the specified security policies. To handle this, RHACS creates a **ValidatingWebhookConfiguration** which contains multiple webhook rules.

When the Kubernetes or OpenShift Container Platform API server receives a request that matches one of the webhook rules, the API server sends an **AdmissionReview** request to RHACS. RHACS then accepts or rejects the request based on the configured security policies.

NOTE

To use admission controller enforcement on OpenShift Container Platform, you need the Red Hat Advanced Cluster Security for Kubernetes version 3.0.49 or newer.

5.1. UNDERSTANDING ADMISSION CONTROLLER ENFORCEMENT

If you intend to use admission controller enforcement, consider the following:

- **API latency**: Using admission controller enforcement increases Kubernetes or OpenShift Container Platform API latency because it involves additional API validation requests. Many standard Kubernetes libraries, such as fabric8, have short Kubernetes or OpenShift Container Platform API timeouts by default. Also, consider API timeouts in any custom automation you might be using.
- **Image scanning**: You can choose whether the admission controller scans images while reviewing requests by setting the **Contact Image Scanners** option in the cluster configuration panel.
  - If you enable this setting, Red Hat Advanced Cluster Security for Kubernetes contacts the image scanners if the scan or image signature verification results are not already available, which adds considerable latency.
  - If you disable this setting, the enforcement decision only considers image scan criteria if cached scan and signature verification results are available.
- You can use admission controller enforcement for:
  - Options in the pod **securityContext**.
  - Deployment configurations.
  - Image components and vulnerabilities.
- You cannot use admission controller enforcement for:
  - Any runtime behavior, such as processes.

  - Any policies based on port exposure.

- The admission controller might fail if there are connectivity issues between the Kubernetes or OpenShift Container Platform API server and RHACS Sensor. To resolve this issue, delete the **ValidatingWebhookConfiguration** object as described in the disabling admission controller enforcement section.

- If you have deploy-time enforcement enabled for a policy and you enable the admission controller, RHACS attempts to block deployments that violate the policy. If a noncompliant deployment is not rejected by the admission controller, for example, in case of a timeout, RHACS still applies other deploy-time enforcement mechanisms, such as scaling to zero replicas.

5.2. ENABLING ADMISSION CONTROLLER ENFORCEMENT

You can enable admission controller enforcement from the **Clusters** view when you install Sensor or edit an existing cluster configuration.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Clusters**.
2. Select an existing cluster from the list or secure a new cluster by selecting **Secure a cluster → Legacy installation method**.
3. If you are securing a new cluster, in the **Static Configuration** section of the cluster configuration panel, enter the details for your cluster.
4. Red Hat recommends that you only turn on the **Configure Admission Controller Webhook to listen on Object Creates** toggle if you are planning to use the admission controller to enforce on object create events.
5. Red Hat recommends that you only turn on the **Configure Admission Controller Webhook to listen on Object Updates** toggle if you are planning to use the admission controller to enforce on update events.
6. Red Hat recommends that you only turn on the **Enable Admission Controller Webhook to listen on exec and port-forward events** toggle if you are planning to use the admission controller to enforce on pod execution and pod port forwards events.
7. Configure the following options in the **Dynamic Configuration** section:
   - **Enforce on Object Creates**: This toggle controls the behavior of the admission control service. You must have the **Configure Admission Controller Webhook to listen on Object Creates** toggle turned on for this to work.
   - **Enforce on Object Updates**: This toggle controls the behavior of the admission control service. You must have the **Configure Admission Controller Webhook to listen on Object Updates** toggle turned on for this to work.
8. Select **Next**.
9. In the **Download files** section, select **Download YAML files and keys**

NOTE

When enabling admission controller for an existing cluster, follow this guidance:

- If you make any changes in the **Static Configuration** section, you must download the YAML files and redeploy the Sensor.
- If you make any changes in the **Dynamic Configuration** section, you can skip downloading the files and deployment, as RHACS automatically synchronizes the Sensor and applies the changes.

10. Select **Finish**.

Verification

- After you provision a new cluster with the generated YAML, run the following command to verify if admission controller enforcement is configured correctly:

```
$ oc get ValidatingWebhookConfiguration (1)
```

(1) If you use Kubernetes, enter **kubectl** instead of **oc**.

Example output

```
NAME      CREATED AT
stackrox  2019-09-24T06:07:34Z
```

5.3. BYPASSING ADMISSION CONTROLLER ENFORCEMENT

To bypass the admission controller, add the **admission.stackrox.io/break-glass** annotation to your configuration YAML. Bypassing the admission controller triggers a policy violation which includes deployment details. Red Hat recommends providing an issue-tracker link or some other reference as the value of this annotation so that others can understand why you bypassed the admission controller.

5.4. DISABLING ADMISSION CONTROLLER ENFORCEMENT

You can disable admission controller enforcement from the **Clusters** view on the Red Hat Advanced Cluster Security for Kubernetes (RHACS) portal.

Procedure

1. In the RHACS portal, select **Platform Configuration** → **Clusters**.
2. Select an existing cluster from the list.
3. Turn off the **Enforce on Object Creates** and **Enforce on Object Updates** toggles in the **Dynamic Configuration** section.
4. Select **Next**.
5. Select **Finish**.

5.4.1. Disabling associated policies

You can turn off the enforcement on relevant policies, which in turn instructs the admission controller to skip enforcements.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Policy Management**.
2. Disable enforcement on the default policies:
    - In the policies view, locate the **Kubernetes Actions: Exec into Pod** policy. Click the overflow menu, ⋮, and then select **Disable policy**.
    - In the policies view, locate the **Kubernetes Actions: Port Forward to Pod** policy. Click the overflow menu, ⋮, and then select **Disable policy**.
3. Disable enforcement on any other custom policies that you have created by using criteria from the default **Kubernetes Actions: Port Forward to Pod** and **Kubernetes Actions: Exec into Pod** policies.

5.4.2. Disabling the webhook

You can disable admission controller enforcement from the **Clusters** view in the RHACS portal.

IMPORTANT

If you disable the admission controller by turning off the webhook, you must redeploy the Sensor bundle.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Clusters**.
2. Select an existing cluster from the list.
3. Turn off the **Enable Admission Controller Webhook to listen on exec and port-forward events** toggle in the **Static Configuration** section.
4. Select **Next** to continue with Sensor setup.
5. Click **Download YAML file and keys**
6. From a system that has access to the monitored cluster, extract and run the **sensor** script:

```
$ unzip -d sensor sensor-<cluster_name>.zip
```

```
$ ./sensor/sensor.sh
```

NOTE

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

After the sensor is deployed, it contacts Central and provides cluster information.

7. Return to the RHACS portal and check if the deployment is successful. If it is successful, a green checkmark appears under section #2. If you do not see a green checkmark, use the following command to check for problems:

   - On OpenShift Container Platform:

     ```
     $ oc get pod -n stackrox -w
     ```

   - On Kubernetes:

     ```
     $ kubectl get pod -n stackrox -w
     ```

8. Select **Finish**.

NOTE

When you disable the admission controller, RHACS does not delete the **ValidatingWebhookConfiguration** parameter. However, instead of checking requests for violations, it accepts all **AdmissionReview** requests.

To remove the **ValidatingWebhookConfiguration** object, run the following command in the secured cluster:

- On OpenShift Container Platform:

  ```
  $ oc delete ValidatingWebhookConfiguration/stackrox
  ```

- On Kubernetes:

  ```
  $ kubectl delete ValidatingWebhookConfiguration/stackrox
  ```

5.5. VALIDATINGWEBHOOKCONFIGURATION YAML FILE CHANGES

With Red Hat Advanced Cluster Security for Kubernetes you can enforce security policies on:

- Object creation
- Object update
- Pod execution
- Pod port forward

If Central or Sensor is unavailable

The admission controller requires an initial configuration from Sensor to work. Kubernetes or OpenShift Container Platform saves this configuration, and it remains accessible even if all admission control service replicas are rescheduled onto other nodes. If this initial configuration exists, the admission controller enforces all configured deploy-time policies.

If Sensor or Central becomes unavailable later:

- you will not be able to run image scans, or query information about cached image scans. However, admission controller enforcement still functions based on the available information gathered before the timeout expires, even if the gathered information is incomplete.
- you will not be able to disable the admission controller from the RHACS portal or modify enforcement for an existing policy as the changes will not get propagated to the admission control service.

NOTE

If you need to disable admission control enforcement, you can delete the validating webhook configuration by running the following command:

- On OpenShift Container Platform:

  ```
  $ oc delete ValidatingWebhookConfiguration/stackrox
  ```

- On Kubernetes:

  ```
  $ kubectl delete ValidatingWebhookConfiguration/stackrox
  ```

Make the admission controller more reliable

Red Hat recommends that you schedule the admission control service on the control plane and not on worker nodes. The deployment YAML file includes a soft preference for running on the control plane, however it is not enforced.

By default, the admission control service runs 3 replicas. To increase reliability, you can increase the replicas by running the following command:

```
$ oc -n stackrox scale deploy/admission-control --replicas=<number_of_replicas> (1)
```

(1) If you use Kubernetes, enter **kubectl** instead of **oc**.

Using with the roxctl CLI

You can use the following options when you generate a Sensor deployment YAML file:

- **--admission-controller-listen-on-updates**: If you use this option, Red Hat Advanced Cluster Security for Kubernetes generates a Sensor bundle with a **ValidatingWebhookConfiguration** pre-configured to receive update events from the Kubernetes or OpenShift Container Platform API server.
- **--admission-controller-enforce-on-updates**: If you use this option, Red Hat Advanced Cluster Security for Kubernetes configures Central such that the admission controller also enforces security policies object updates.

Both these options are optional, and are **false** by default.

CHAPTER 6. MANAGING SECURITY POLICIES

Red Hat Advanced Cluster Security for Kubernetes allows you to use out-of-the-box security policies and define custom multi-factor policies for your container environment. Configuring these policies enables you to automatically prevent high-risk service deployments in your environment and respond to runtime security incidents.

6.1. USING DEFAULT SECURITY POLICIES

Red Hat Advanced Cluster Security for Kubernetes includes a set of default policies that provide broad coverage to identify security issues and ensure best practices for security in your environment.

To view the default policies:

- In the RHACS portal, go to **Platform Configuration** → **Policy Management**.

The **Policies** view lists the default policies and includes the following parameters for each policy:

- **Policy**: A name for the policy.
- **Description**: A longer, more detailed description of the alert for the policy.
- **Status**: The current status of the policy, either **Enabled** or **Disabled**.
- **Notifiers**: The list of notifiers that are configured for the policy.
- **Severity**: A ranking of the policy, either critical, high, medium, or low, for the amount of attention required.
- **Lifecycle**: The phase of the container lifecycle (build, deploy, or runtime) that this policy applies to, and the phase at which enforcement applies, when the policy is enabled.

The default policies have preconfigured parameters and belong to categories such as:

- Anomalous Activity
- Cryptocurrency Mining
- DevOps Best Practices
- Kubernetes
- Network Tools
- Package Management
- Privileges
- Security Best Practices
- System Modification
- Vulnerability Management

You can edit these categories and create your own categories.

NOTE

You cannot delete default policies or edit policy criteria for default policies.

6.2. MODIFYING EXISTING SECURITY POLICIES

You can edit the policies you have created and the existing default policies provided by Red Hat Advanced Cluster Security for Kubernetes.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Policy Management**.
2. From the **Policies** page, select the policy you want to edit.
3. Select **Actions → Edit policy**.
4. Modify the **Policy details**. You can modify the policy name, severity, categories, description, rationale, and guidance. You can also attach notifiers to the policy by selecting from the available **Notifiers** under the **Attach notifiers** section.
5. Click **Next**.
6. In the **Policy behavior** section, select the **Lifecycle stages** and **Event sources** for the policy.
7. Select a **Response method** to address violations for the policy.
8. Click **Next**.
9. In the **Policy criteria** section, expand the categories under the **Drag out policy fields** section. Use the drag-and-drop policy fields to specify logical conditions for the policy criteria.

   NOTE

   You cannot edit policy criteria for default policies.

10. Click **Next**.
11. In the **Policy scope** section, modify **Restrict by scope**, **Exclude by scope**, and **Exclude images** settings.
12. Click **Next**.
13. In the **Review policy** section, preview the policy violations.
14. Click **Save**.

Additional resources

- Creating a security policy from the system policies view

6.3. CREATING AND MANAGING POLICY CATEGORIES

6.3.1. Creating policy categories by using the Policy categories tab

Beginning with version 3.74, RHACS provides a new method to create and manage policy categories in Red Hat Advanced Cluster Security Cloud Service or in RHACS if you have the PostgreSQL database enabled. All policy workflows other than policy creation remain unchanged when using this feature.

You can also configure policy categories by using the **PolicyCategoryService** API object. For more information, go to **Help** → **API reference** in the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Policy Management**.
2. Click the **Policy categories** tab. This tab provides a list of existing categories and allows you to filter the list by category name. You can also click **Show all categories** and select the checkbox to remove default or custom categories from the displayed list.
3. Click **Create category**.
4. Enter a category name and click **Create**.

6.3.2. Modifying policy categories by using the Policy categories tab

Beginning with version 3.74, RHACS provides a new method to create and manage policy categories in Red Hat Advanced Cluster Security Cloud Service or in RHACS if you have the PostgreSQL database enabled. All policy workflows other than policy creation remain unchanged when using this feature.

You can also configure policy categories by using the **PolicyCategoryService** API object. For more information, go to **Help** → **API reference** in the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Policy Management**.
2. Click the **Policy categories** tab. This tab provides a list of existing categories and allows you to filter the list by category name. You can also click **Show all categories** and select the checkbox to remove default or custom categories from the displayed list.
3. Click a policy name to edit or delete it. Default policy categories cannot be selected, edited, or deleted.

Additional resources

- Creating a security policy from the system policies view

6.4. CREATING CUSTOM POLICIES

In addition to using the default policies, you can also create custom policies in Red Hat Advanced Cluster Security for Kubernetes.

To build a new policy, you can clone an existing policy or create a new one from scratch.

- You can also create policies based on the filter criteria in the **Risk** view in the RHACS portal.
- You can also use **AND**, **OR**, and **NOT** logical operators for policy criteria to create advanced policies.

6.4.1. Creating a security policy from the system policies view

You can create new security policies from the system policies view.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Policy Management**.
2. Click **Create policy**.
3. Enter the following details about your policy in the **Policy details** section:
   - Enter a **Name** for the policy.
   - Optional: Attach notifiers to the policy by selecting from the available **Notifiers** under the **Attach notifiers** section.

     NOTE

     Before you can forward alerts, you must integrate RHACS with your notification provider, such as webhooks, Jira, PagerDuty, Splunk, or others.
   - Select a **Severity** level for this policy, either **Critical**, **High**, **Medium**, or **Low**.
   - Select policy **Categories** you want to apply to this policy. For information about creating categories, see "Creating and managing policy categories" later in this document.
   - Enter details about the policy in the **Description** field.
   - Enter an explanation about why the policy exists in the **Rationale** field.
   - Enter steps to resolve violations of this policy in the **Guidance** field.
   - Optional: Under the **MITRE ATT&CK** section, select the tactics and the techniques you want to specify for the policy.
     a. Click **Add tactic**, and then select a tactic from the drop-down list.
     b. Click the **Add technique** to add techniques for the selected tactic. You can specify multiple techniques for a tactic.
4. Click **Next**.
5. In the **Policy behavior** section, take the following steps:
   a. Select the **Lifecycle stages** to which your policy is applicable: **Build**, **Deploy**, or **Runtime**. You can select more than one stage.
      - Build-time policies apply to image fields such as CVEs and Dockerfile instructions.
      - Deploy-time policies can include all build-time policy criteria but they can also include data from your cluster configurations, such as running in privileged mode or mounting the Docker socket.
      - Runtime policies can include all build-time and deploy-time policy criteria but they can also include data about process executions during runtime.

b. Optional: If you selected the **Runtime** lifecycle stage, select one of the following **Event sources**:

   - Deployment: RHACS triggers policy violations when event sources include process and network activity, pod exec and pod port forwarding.
   - RHACS triggers policy violations when event sources match Kubernetes audit log records.

6. For **Response method**, select one of the following options:

   a. **Inform**: include the violation in the violations list.

   b. **Inform and enforce**: enforce actions.

7. Optional: If you selected **Inform and enforce**, in **Configure enforcement behavior**, select the enforcement behavior for the policy by using the toggle for each lifecycle. It is only available for the stages you select when configuring **Lifecycle stages**. The enforcement behavior is different for each lifecycle stage.

   - **Build**: RHACS fails your continuous integration (CI) builds when images match the criteria of the policy.
   - **Deploy**: For the **Deploy** stage, RHACS blocks the creation and update of deployments that match the conditions of the policy if the RHACS admission controller is configured and running.
     - In clusters with admission controller enforcement, the Kubernetes or OpenShift Container Platform API server blocks all noncompliant deployments. In other clusters, RHACS edits noncompliant deployments to prevent pods from being scheduled.
     - For existing deployments, policy changes only result in enforcement at the next detection of the criteria, when a Kubernetes event occurs. For more information about enforcement, see "Security policy enforcement for the deploy stage".
   - **Runtime** - RHACS deletes all pods when an event in the pods matches the criteria of the policy.

     WARNING

     Policy enforcement can impact running applications or development processes. Before you enable enforcement options, inform all stakeholders and plan about how to respond to automated enforcement actions.

8. Click **Next**.

9. In the **Policy Criteria** section, configure the attributes that you want to trigger the policy for.

   a. Click and drag policy fields into the **Policy Section** to add criteria.

NOTE

The policy fields that are available depend on the lifecycle stage you chose for the policy. For example, criteria under **Kubernetes access policies** or **Networking** are available when creating a policy for the runtime lifecycle, but not when creating a policy for the build lifecycle. See "Policy criteria" in the "Additional resources" section for more information about policy criteria, including information about criteria and the lifecycle phase in which they are available.

b. Optional: Click **Add condition** to add policy sections that contain additional criteria that will trigger the policy (for example, to trigger on old, stale images, you can configure that **image tag is not latest** or **image age** and specify a minimum number of days since an image is built).

10. Click **Next**.

11. In the **Policy scope** section, configure the following:

    - Click **Add inclusion scope** to use **Restrict by scope** to enable this policy only for a specific cluster, a namespace, or a label. You can add multiple scopes and also use regular expression in RE2 Syntax for namespaces and labels.

    - Click **Add exclusion scope** to use **Exclude by scope** to exclude deployments, clusters, namespaces, and labels you specify. The policy will not apply to the entities that you select. You can add multiple scopes and also use regular expression in RE2 Syntax for namespaces and labels. However, you cannot use regular expression for selecting deployments.

    - For **Excluded Images (Build Lifecycle only)**, select all images that you do not want to trigger a violation for.

NOTE

The **Excluded Images** setting only applies when you check images in a continuous integration system with the **Build** lifecycle stage. It does not have any effect if you use this policy to check running deployments in the **Deploy** lifecycle stage or runtime activities in the **Runtime** lifecycle stage.

12. Click **Next**.

13. In the **Review policy** section, preview the policy violations.

14. Click **Save**.

6.4.1.1. Security policy enforcement for the deploy stage

Red Hat Advanced Cluster Security for Kubernetes supports two forms of security policy enforcement for deploy-time policies: hard enforcement through the admission controller and soft enforcement by RHACS Sensor. The admission controller blocks creation or updating of deployments that violate policy. If the admission controller is disabled or unavailable, Sensor can perform enforcement by scaling down replicas for deployments that violate policy to **0**.

WARNING

Policy enforcement can impact running applications or development processes. Before you enable enforcement options, inform all stakeholders and plan how to respond to the automated enforcement actions.

6.4.1.1.1. Hard enforcement

Hard enforcement is performed by the RHACS admission controller. In clusters with admission controller enforcement, the Kubernetes or OpenShift Container Platform API server blocks all noncompliant deployments. The admission controller blocks **CREATE** and **UPDATE** operations. Any pod create or update request that satisfies a policy configured with deploy-time enforcement enabled will fail.

NOTE

Kubernetes admission webhooks support only **CREATE**, **UPDATE**, **DELETE**, or **CONNECT** operations. The RHACS admission controller supports only **CREATE** and **UPDATE** operations. Operations such as **kubectl patch**, **kubectl set**, and **kubectl scale** are PATCH operations, not UPDATE operations. Because PATCH operations are not supported in Kubernetes, RHACS cannot perform enforcement on PATCH operations.

For blocking enforcement, you must enable the following settings for the cluster in RHACS:

- **Enforce on Object Creates**: This toggle in the **Dynamic Configuration** section controls the behavior of the admission control service. You must have the **Configure Admission Controller Webhook to listen on Object Creates** toggle in the **Static Configuration** section turned on for this to work.
- **Enforce on Object Updates**: This toggle in the **Dynamic Configuration** section controls the behavior of the admission control service. You must have the **Configure Admission Controller Webhook to listen on Object Updates** toggle in the **Static Configuration** section turned on for this to work.

If you make changes to settings in the **Static Configuration** setting, you must redeploy the secured cluster for those changes to take effect.

6.4.1.1.2. Soft enforcement

Soft enforcement is performed by RHACS Sensor. This enforcement prevents an operation from being initiated. With soft enforcement, Sensor scales the replicas to 0, and prevents pods from being scheduled. In this enforcement, a non-ready deployment is available in the cluster.

If soft enforcement is configured, and Sensor is down, then RHACS cannot perform enforcement.

6.4.1.1.3. Namespace exclusions

By default, RHACS excludes certain administrative namespaces, such as the **stackrox**, **kube-system**, and **istio-system** namespaces, from enforcement blocking. The reason for this is that some items in these namespaces must be deployed for RHACS to work correctly.

6.4.1.1.4. Enforcement on existing deployments

For existing deployments, policy changes only result in enforcement at the next detection of the criteria, when a Kubernetes event occurs. If you make changes to a policy, you must reassess policies by selecting **Policy Management** and clicking **Reassess All**. This action applies deploy policies on all existing deployments regardless of whether there are any new incoming Kubernetes events. If a policy is violated, then RHACS performs enforcement.

Additional resources

- Policy criteria
- Using admission controller enforcement

6.4.2. Creating a security policy from the risk view

While evaluating risks in your deployments in the **Risk** view, when you apply local page filtering, you can create new security policies based on the filtering criteria you are using.

Procedure

1. Go to the RHACS portal and select **Risk** from the navigation menu.
2. Apply local page filtering criteria that you want to create a policy for.
3. Select **New Policy** and fill in the required fields to create a new policy.

Additional resources

- Using local page filtering
- Creating a security policy from the system policies view

6.4.3. Policy criteria

In the **Policy Criteria** section you can configure the data on which you want to trigger a policy.

You can configure the policy based on the attributes listed in the following table.

In this table:

- The **Regular expressions**, **AND, OR**, and **NOT** columns indicate whether you can use regular expressions and other logical operators along with the specific attribute.
  - **!** for **Regex** (Regular expressions) indicates that you can only use regular expressions for the listed fields.
  - **!** for **AND**, or **OR** indicates that you can only use the mentioned logical operator for the attribute.
  - ✕ in the **Regex** / **NOT** / **AND, OR** column indicates that the attribute does not support any of those (regex, negation, logical operators).
- The **RHACS version** column indicates the version of Red Hat Advanced Cluster Security for Kubernetes that you must have to use the attribute.

- You cannot use logical combination operators **AND** and **OR** for attributes that have:
  - Boolean values **true** and **false**
  - Minimum-value semantics, for example:
    - **Minimum RBAC permissions**
    - **Days since image was created**
- You cannot use the **NOT** logical operator for attributes that have:
  - Boolean values **true** and **false**
  - Numeric values that already use comparison, such as the **<**, **>**, **<=**, **>=** operators.
  - Compound criteria that can have multiple values, for example:
    - **Dockerfile Line**, which includes both instructions and arguments.
    - **Environment Variable**, which consists of both name and value.
  - Other meanings, including **Add Capabilities**, **Drop Capabilities**, **Days since image was created**, and **Days since image was last scanned**

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Section: Image registry					
Image Registry	The name of the image registry.	Image Registry	String	Regex, NOT, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image Name	The full name of the image in registry, for example **library/nginx**.	Image Remote	String	Regex, NOT, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image Tag	Identifier for an image.	Image Tag	String	Regex, NOT, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Image Signature	The list of signature integrations you can use to verify an image's signature. Create alerts on images that either do not have a signature or their signature is not verifiable by at least one of the provided signature integrations.	Image Signature Verified By	A valid ID of an already configured image signature integration	! **OR** only	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Section: Image contents					
CVE is fixable	This criterion results in a violation only if the image in the deployment you are evaluating has a fixable CVE.	Fixable	Boolean	✕	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Days Since CVE Was First Discovered In Image	This criterion results in a violation only if it has been more than a specified number of days since RHACS discovered the CVE in a specific image.	Days Since CVE Was First Discovered In Image	Integer	✕	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Days Since CVE Was First Discovered In System	This criterion results in a violation only if it has been more than a specified number of days since RHACS discovered the CVE across all deployed images in all clusters that RHACS monitors.	Days Since CVE Was First Discovered In System	Integer	×	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image age	The minimum number of days from image creation date.	Image Age	Integer	×	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image scan age	The minimum number of days since the image was last scanned.	Image Scan Age	Integer	×	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image User	Matches the USER directive in the Dockerfile. See https://docs.docker.com/engine/reference/builder/#user for details .	Image User	String	Regex, NOT, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Dockerfile Line	A specific line in the Dockerfile, including both instructions and arguments.	Dockerfile Line	One of: LABEL, RUN, CMD, EXPOSE, ENV, ADD, COPY, ENTRYPOINT, VOLUME, USER, WORKDIR, ONBUILD	! Regex only for values, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image scan status	Check if an image was scanned.	Unscanned Image	Boolean	×	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
CVSS	Common Vulnerability Scoring System, use it to match images with vulnerabilities whose scores are greater than **>**, less than **<**, or equal to **=** the specified CVSS.	CVSS	<, >, <=, >= or nothing (which implies equal to) — and — a decimal (a number with an optional fractional value). Examples: >=5, or 9.5	AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Severity	The severity of the vulnerability based on the CVSS or the vendor. Can be one of Low, Moderate, Important or Critical.	Severity	<, >, ⇐, >= or nothing (which implies equal to) — and — One of: UNKNOWN LOW MODERATE IMPORTANT CRITICAL Examples: >=IMPORTANT , or CRITICAL	AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Fixed By	The version string of a package that fixes a flagged vulnerability in an image. This criterion may be used in addition to other criteria that identify a vulnerability, for example using the CVE criterion.	Fixed By	String	Regex, NOT, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
CVE	Common Vulnerabilities and Exposures, use it with specific CVE numbers.	CVE	String	Regex, NOT, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image Component	Name and version number of a specific software component present in an image.	Image Component	key=value Value is optional. If value is missing, it must be in format "key=".	Regex, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Image OS	Name and version number of the base operating system of the image. For example, **alpine:3.17.3**	Image OS	String	Regex, NOT, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Require image label	Ensure the presence of a Docker image label. The policy triggers if any image in the deployment does not have the specified label. You can use regular expressions for both key and value fields to match labels. The **Require Image Label** policy criteria only works when you integrate with a Docker registry. For details about Docker labels see Docker documentation , https://docs.docker.com/config/labels-custom-metadata/.	Required Image Label	key=value Value is optional. If value is missing, it must be in format "key=".	Regex, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Disallow image label	Ensure that a particular Docker image label is NOT used. The policy triggers if any image in the deployment has the specified label. You can use regular expressions for both key and value fields to match labels. The 'Disallow Image Label policy' criteria only works when you integrate with a Docker registry. For details about Docker labels see Docker documentation , https://docs.docker.com/config/labels-custom-metadata/.	Disallowed Image Label	key=value Value is optional. If value is missing, it must be in format "key=".	Regex, AND, OR	**Build**, **Deploy**, **Runtime** (when used with a Runtime criterion)
Section: Container configuration					

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Environment Variable	Check environment variables by name or value.	Environment Variable	RAW=key=value to match an environment variable as directly specified in the deployment configuration with a specific key and value. **value** can be omitted to match on only the key. If the environment variable is not directly defined in the configuration, then the format SOURCE=KEY can be used, where SOURCE is one of SECRET_KEY, CONFIG_MAP_KEY, FIELD or RESOURCE_FIELD. In this case, criteria can only match the key and not the value.	! Regex only for key and value (if using RAW) AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Container CPU Request	Check for the number of cores reserved for a given resource.	Container CPU Request	<, >, ⇐, >= or nothing (which implies equal to) – and – A decimal (a number with an optional fractional value) Examples: >=5, or 9.5	AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Container CPU Limit	Check for the maximum number of cores a resource is allowed to use.	Container CPU Limit	(Same as Container CPU Request)	AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Container Memory Request	Check for the amount of memory reserved for a given resource.	Container Memory Request	(Same as Container CPU Request)	AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Container Memory Limit	Check for the maximum amount of memory a resource is allowed to use.	Container Memory Limit	(Same as Container CPU Request)	AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Privileged container	Privileged running deployments.	Privileged Container	Boolean	×	**Deploy**, **Runtime** (when used with a Runtime criterion)
Root filesystem writeability	Containers running with the root file system configured as read only.	Read-Only Root Filesystem	Boolean	×	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Seccomp Profile Type	The type of seccomp profile allowed for the container.	Seccomp Profile Type	One of: UNCONFINED RUNTIME_DEFAULT LOCALHOST	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)
Privilege escalation	Provides alerts when a development is configured to allow a container process to gain more privileges than its parent process.	Allow Privilege Escalation	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Drop Capabilities	Linux capabilities that must be dropped from the container. Provides alerts when the specified capabilities are not dropped. For example, if configured with **SYS_ADMIN** AND **SYS_BOOT**, and the deployment drops only *one* or *neither* of these two capabilities, the alert occurs.	Drop Capabilities	One of: ALL AUDIT_CONTROL AUDIT_READ AUDIT_WRITE BLOCK_SUSPEND CHOWN DAC_OVERRIDE DAC_READ_SEARCH FOWNER FSETID IPC_LOCK IPC_OWNER KILL LEASE LINUX_IMMUTABLE MAC_ADMIN MAC_OVERRIDE MKNOD NET_ADMIN NET_BIND_SERVICE NET_BROADCAST NET_RAW SETGID SETFCAP SETPCAP SETUID SYS_ADMIN SYS_BOOT SYS_CHROOT SYS_MODULE SYS_NICE SYS_PACCT SYS_PTRACE SYS_RAWIO SYS_RESOURCE SYS_TIME SYS_TTY_CONFIG SYSLOG WAKE_ALARM	AND	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Add Capabilities	Linux capabilities that must not be added to the container, such as the ability to send raw packets or override file permissions. Provides alerts when the specified capabilities are added. For example, if configured with **NET_ADMIN** or **NET_RAW**, and the deployment manifest YAML file includes at least one of these two capabilities, the alert occurs.	Add Capabilities	AUDIT_CONTROL AUDIT_READ AUDIT_WRITE BLOCK_SUSPEND CHOWN DAC_OVERRIDE DAC_READ_SEARCH FOWNER FSETID IPC_LOCK IPC_OWNER KILL LEASE LINUX_IMMUTABLE MAC_ADMIN MAC_OVERRIDE MKNOD NET_ADMIN NET_BIND_SERVICE NET_BROADCAST NET_RAW SETGID SETFCAP SETPCAP SETUID SYS_ADMIN SYS_BOOT SYS_CHROOT SYS_MODULE SYS_PACCT SYS_PTRACE SYS_RAWIO SYS_RESOURCE SYS_TIME SYS_TTY_CONFIG SYSLOG WAKE_ALARM	OR	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Container Name	The name of the container.	Container Name	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
AppArmor Profile	The Application Armor ("AppArmor") profile used in the container.	AppArmor Profile	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Liveness Probe	Whether the container defines a liveness probe.	Liveness Probe	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)
Readiness Probe	Whether the container defines a readiness probe.	Readiness Probe	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)
Section: Deployment metadata					
Disallowed Annotation	An annotation which is not allowed to be present on Kubernetes resources in a specified environment.	Disallowed Annotation	key=value Value is optional. If value is missing, it must be in format "key=".	Regex, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Required Label	Check for the presence of a required label in Kubernetes.	Required Label	key=value Value is optional. If value is missing, it must be in format "key=".	Regex, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Required Annotation	Check for the presence of a required annotation in Kubernetes.	Required Annotation	key=value Value is optional. If value is missing, it must be in format "key=".	Regex, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Runtime Class	The **RuntimeClass** of the deployment.	Runtime Class	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Host Network	Check if **HostNetwork** is enabled which means that the container is not placed inside a separate network stack (for example, the container's networking is not containerized). This implies that the container has full access to the host's network interfaces.	Host Network	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Host PID	Check if the Process ID (PID) namespace is isolated between the containers and the host. This allows for processes in different PID namespaces to have the same PID.	Host PID	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)
Host IPC	Check if the IPC (POSIX/SysV IPC) namespace (which provides separation of named shared memory segments, semaphores and message queues) on the host is shared with containers.	Host IPC	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)
Namespace	The name of the namespace the deployment belongs to.	Namespace	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Replicas	The number of deployment replicas. If you use **oc scale** to scale the deployment replicas from 0 to a number, then the admission controller blocks this action if the deployment violates a policy.	Replicas	<, >, ⇐, >= or nothing (which implies equal to) – and – a decimal (a number with an optional fractional value). Examples: >=5, or 9.5	NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Section: Storage					
Volume Name	Name of the storage.	Volume Name	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Volume Source	Indicates the form in which the volume is provisioned. For example, **persistentVolumeClaim** or **hostPath**.	Volume Source	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Volume Destination	The path where the volume is mounted.	Volume Destination	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Volume Type	The type of volume.	Volume Type	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Mounted volume writability	Volumes that are mounted as writable.	Writable Mounted Volume	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)
Mount Propagation	Check if container is mounting volumes in **Bidirectional**, **Host to Container**, or **None** modes.	Mount Propagation	One of: NONE HOSTTOCON TAINER BIDIRECTION AL	NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Host mount writability	Resource has mounted a path on the host with write permissions.	Writable Host Mount	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)
Section: Networking					
Protocol	Protocol, such as, TCP or UDP, that is used by the exposed port.	Exposed Port Protocol	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Port	Port numbers exposed by a deployment.	Exposed Port	<, >, <=, >= or nothing (which implies equal to) – and – an integer. Examples: >=1024, or 22	NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Exposed Node Port	Port numbers exposed externally by a deployment.	Exposed Node Port	(Same as Exposed Port)	NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Port Exposure	Exposure method of the service, for example, load balancer or node port.	Port Exposure Method	One of: UNSET EXTERNAL NODE HOST INTERNAL ROUTE	NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Unexpected Network Flow Detected	Check if the detected network traffic is part of the network baseline for the deployment.	Unexpected Network Flow Detected	Boolean	✕	**Runtime** ONLY - Network
Ingress Network Policy	Check the presence or absence of ingress Kubernetes network policies.	Has Ingress Network Policy	Boolean	Regex, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Egress Network Policy	Check the presence or absence of egress Kubernetes network policies.	Has Egress Network Policy	Boolean	Regex, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Section: Process activity					
Process Name	Name of the process executed in a deployment.	Process Name	String	Regex, NOT, AND, OR	**Runtime** ONLY - Process
Process Ancestor	Name of any parent process for a process executed in a deployment.	Process Ancestor	String	Regex, NOT, AND, OR	**Runtime** ONLY - Process

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Process Arguments	Command arguments for a process executed in a deployment.	Process Arguments	String	Regex, NOT, AND, OR	**Runtime** ONLY - Process
Process UID	Unix user ID for a process executed in a deployment.	Process UID	Integer	NOT, AND, OR	**Runtime** ONLY - Process
Unexpected Process Executed	Check deployments for which process executions are not listed in the deployment's locked process baseline.	Unexpected Process Executed	Boolean	✕	**Runtime** ONLY - Process
Section: Kubernetes access					
Service Account	The name of the service account.	Service Account	String	Regex, NOT, AND, OR	**Deploy**, **Runtime** (when used with a Runtime criterion)
Automount Service Account Token	Check if the deployment configuration automatically mounts the service account token.	Automount Service Account Token	Boolean	✕	**Deploy**, **Runtime** (when used with a Runtime criterion)

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Minimum RBAC Permissions	Match if the deployment's Kubernetes service account has Kubernetes RBAC permission level equal to **=** or greater than **>** the specified level.	Minimum RBAC Permissions	One of: DEFAULT ELEVATED_IN_NAMESPACE ELEVATED_CLUSTER_WIDE CLUSTER_ADMIN	NOT	**Deploy**, **Runtime** (when used with a Runtime criterion)
Section: Kubernetes events					
Kubernetes Action	The name of the Kubernetes action, such as **Pod Exec**.	Kubernetes Resource	One of: PODS_EXEC PODS_PORTFORWARD	! **OR** only	**Runtime** ONLY - Kubernetes Events
Kubernetes User Name	The name of the user who accessed the resource.	Kubernetes User Name	Alphanumeric with hyphens (-) and colon (:) only	Regex, NOT, ! **OR** only	**Runtime** ONLY - Kubernetes Events
Kubernetes User Group	The name of the group to which the user who accessed the resource belongs to.	Kubernetes User Groups	Alphanumeric with hyphens (-) and colon (:) only	Regex, ! **OR** only	**Runtime** ONLY - Kubernetes Events
Kubernetes Resource	The name of the accessed Kubernetes resource, such as **configmaps** or **secrets**.	Kubernetes Resource	One of: SECRETS CONFIGMAPS	! **OR** only	**Runtime** ONLY - Audit Log
Kubernetes API Verb	The Kubernetes API verb that is used to access the resource, such as **GET** or **POST**.	Kubernetes API Verb	One of: CREATE DELETE GET PATCH UPDATE	! **OR** only	**Runtime** ONLY - Audit Log

Attribute	Description	JSON Attribute	Allowed Values	Regex, NOT, AND, OR	Phase
Kubernetes Resource Name	The name of the accessed Kubernetes resource.	Kubernetes Resource Name	Alphanumeric with hyphens (-) and colon (:) only	Regex, NOT, ! **OR** only	**Runtime** ONLY - Audit Log
User Agent	The user agent that the user used to access the resource. For example **oc**, or **kubectl**.	User Agent	String	Regex, NOT, ! **OR** only	**Runtime** ONLY - Audit Log
Source IP Address	The IP address from which the user accessed the resource.	Source IP Address	IPV4 or IPV6 address	Regex, NOT, ! **OR** only	**Runtime** ONLY - Audit Log
Is Impersonated User	Check if the request was made by a user that is impersonated by a service account or some other account.	Is Impersonated User	Boolean	×	**Runtime** ONLY - Audit Log

6.4.3.1. Adding logical conditions for the policy criteria

You can use the drag-and-drop policy fields panel to specify logical conditions for the policy criteria.

Prerequisites

- You must be using Red Hat Advanced Cluster Security for Kubernetes version 3.0.45 or newer.

Procedure

1. In the **Policy Criteria** section, select **Add a new condition** to add a new policy section.
   - You can click on the **Edit** icon to rename the policy section.
   - The **Drag out a policy field** section lists available policy criteria in multiple categories. You can expand and collapse these categories to view the policy criteria attributes.
2. Drag an attribute to the **Drop a policy field inside** area of the policy section.

3. Depending on the type of the attribute you select, you get different options to configure the conditions for the selected attribute. For example:
   - If you select an attribute with Boolean values **Read-Only Root Filesystem**, you will see **READ-ONLY** and **WRITABLE** options.
   - If you select an attribute with compound values **Environment variable**, you will see options to enter values for **Key**, **Value**, and **Value From** fields, and an icon to add more values for the available options.
     a. To combine multiple values for an attribute, click the **Add** icon.
     b. You can also click on the logical operator **AND** or **OR** listed in a policy section, to toggle between **AND** and **OR** operators. Toggling between operators only works inside a policy section and not between two different policy sections.
4. You can specify more than one **AND** and **OR** condition by repeating these steps. After you configure the conditions for the added attributes, click **Next** to continue with the policy creation.

6.5. SHARING SECURITY POLICIES

Beginning from Red Hat Advanced Cluster Security for Kubernetes version 3.0.44, you can share your security policies between different Central instances, by exporting and importing policies. It helps you enforce the same standards for all your clusters. To share policies, you export them as JSON files, and then import them back into another Central instance.

NOTE

Currently, you cannot export multiple security policies at once by using the RHACS portal. However, you can use the API for exporting multiple security policies. In the RHACS portal, go to **Help → API reference** to see the API reference.

6.5.1. Exporting a security policy

When you export a policy, it includes all the policy contents and also includes cluster scopes, cluster exclusions, and all configured notifications.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Policy Management**.
2. From the **Policies** page, select the policy you want to edit.
3. Select **Actions → Export policy to JSON**

6.5.2. Importing a security policy

You can import a security policy from the **System Policies** view on the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Policy Management**.
2. Click **Import policy**.

3. In the **Import policy JSON** dialog, click **Upload** and select the JSON file you want to upload.

4. Click **Begin import**.

Each security policy in RHACS has a unique ID (UID) and a unique name. When you import a policy, RHACS handles the uploaded policy as follows:

- If the imported policy UID and name do not match any existing policy, RHACS creates a new policy.
- If the imported policy has the same UID as an existing policy, but a different name, you can either:
    - Keep both policies. RHACS saves the imported policy with a new UID.
    - Replace the existing policy with the imported policy.
- If the imported policy has the same name as an existing policy, but a different UID, you can either:
    - Keep both policies by providing a new name for the imported policy.
    - Replace the existing policy with the imported policy.
- If the imported policy has the same name and UID as an existing policy, the Red Hat Advanced Cluster Security for Kubernetes checks if the policy criteria match to the existing policy. If the policy criteria match, RHACS keeps the existing policy and shows a success message. If the policy criteria do not match, you can either:
    - Keep both policies by providing a new name for the imported policy.
    - Replace the existing policy with the imported policy.

IMPORTANT

- If you import into the same Central instance, RHACS uses all the exported fields.
- If you import into a different Central instance, RHACS omits certain fields, such as cluster scopes, cluster exclusions, and notifications. RHACS shows these omitted fields in a message. These fields vary for every installation, and you cannot migrate them from one Central instance to another.

CHAPTER 7. DEFAULT SECURITY POLICIES

The default security policies in Red Hat Advanced Cluster Security for Kubernetes provide broad coverage to identify security issues and ensure best practices for security in your environment. By configuring those policies, you can automatically prevent high-risk service deployments in your environment and respond to runtime security incidents.

NOTE

The severity levels for policies in Red Hat Advanced Cluster Security for Kubernetes are different from the severity levels that Red Hat Product Security assigns.

The Red Hat Advanced Cluster Security for Kubernetes policy severity levels are Critical, High, Medium, and Low. Red Hat Product Security rates vulnerability severity levels as Critical, Important, Moderate, and Low.

While a policy's severity level and the Red Hat Product Security severity levels can interact, it is important to distinguish between them. For more information about the Red Hat Product Security severity levels, see Severity Ratings.

7.1. CRITICAL SEVERITY SECURITY POLICIES

The following table lists the default security policies in Red Hat Advanced Cluster Security for Kubernetes that are of critical severity. The policies are organized by life cycle stage.

Table 7.1. Critical severity security policies

Life cycle stage	Name	Description	Status
Build or Deploy	Apache Struts: CVE-2017-5638	Alerts when deployments have images that contain the CVE-2017-5638 Apache Struts vulnerability.	Enabled
Build or Deploy	Log4Shell: log4j Remote Code Execution vulnerability	Alerts when deployments include images that contain the CVE-2021-44228 and CVE-2021-45046 Log4Shell vulnerabilities. Flaws exist in the Apache Log4j Java logging library in versions 2.0-beta9 - 2.15.0, excluding version 2.12.2.	Enabled

Life cycle stage	Name	Description	Status
Build or Deploy	Spring4Shell (Spring Framework Remote Code Execution) and Spring Cloud Function vulnerabilities	Alerts when deployments include images that contain either the CVE-2022-22965 vulnerability, which affects Spring MVC, and the CVE-2022-22963 vulnerability, which affects Spring Cloud. In versions 3.16, 3.2.2, and older unsupported versions, Spring Cloud contains flaws. Flaws exist in Spring Framework in versions 5.3.0 - 5.3.17, versions 5.2.0 - 5.2.19, and in older unsupported versions.	Enabled
Runtime	Iptables Executed in Privileged Container	Alerts when privileged pods run iptables.	Enabled

7.2. HIGH SEVERITY SECURITY POLICIES

The following table lists the default security policies in Red Hat Advanced Cluster Security for Kubernetes that are of high severity. The policies are organized by life cycle stage.

Table 7.2. High severity security policies

Life cycle stage	Name	Description	Status
Build or Deploy	Fixable CVSS >= 7	Alerts when deployments with fixable vulnerabilities have a CVSS of at least 7.	Disabled
Build or Deploy	Fixable Severity at least Important	Alerts when deployments with fixable vulnerabilities have a severity rating of at least Important.	Enabled

Life cycle stage	Name	Description	Status
Build or Deploy	Secure Shell (ssh) Port Exposed in Image	Alerts when deployments expose port 22, which is commonly reserved for SSH access.	Enabled
Deploy	Emergency Deployment Annotation	Alerts when deployments use the emergency annotation, such as "admission.stackrox.io/break-glass":"ticket-1234" to circumvent StackRox Admission controller checks.	Enabled
Deploy	Environment Variable Contains Secret	Alerts when deployments have environment variables that contain 'SECRET'.	Enabled
Deploy	Fixable CVSS >= 6 and Privileged	Alerts when deployments run in privileged mode with fixable vulnerabilities that have a CVSS of at least 6.	Disabled by default in version 3.72.0 and later
Deploy	Privileged Containers with Important and Critical Fixable CVEs	Alerts when containers that run in privileged mode have important or critical fixable vulnerabilities.	Enabled
Deploy	Secret Mounted as Environment Variable	Alerts when a deployment has a Kubernetes secret that is mounted as an environment variable.	Disabled
Deploy	Secure Shell (ssh) Port Exposed	Alerts when deployments expose port 22, which is commonly reserved for SSH access.	Enabled
Runtime	Cryptocurrency Mining Process Execution	Spawns the crypto-currency mining process.	Enabled

Life cycle stage	Name	Description	Status
Runtime	iptables Execution	Detects when someone runs iptables, which is a deprecated way of managing network states in containers.	Enabled
Runtime	Kubernetes Actions: Exec into Pod	Alerts when the Kubernetes API receives a request to run a command in a container.	Enabled
Runtime	Linux Group Add Execution	Detects when someone runs the addgroup or groupadd binary to add a Linux group.	Enabled
Runtime	Linux User Add Execution	Detects when someone runs the useradd or adduser binary to add a Linux user.	Enabled
Runtime	Login Binaries	Indicates when someone tries to log in.	Disabled
Runtime	Network Management Execution	Detects when someone runs binary files that can manipulate network configuration and management.	Enabled
Runtime	nmap Execution	Alerts when someone starts the nmap process in a container during run time.	Enabled
Runtime	OpenShift: Kubeadmin Secret Accessed	Alerts when someone accesses the kubeadmin secret.	Enabled
Runtime	Password Binaries	Indicates when someone attempts to change a password.	Disabled
Runtime	Process Targeting Cluster Kubelet Endpoint	Detects the misuse of the healthz, kubelet API, or heapster endpoint.	Enabled

Life cycle stage	Name	Description	Status
Runtime	Process Targeting Cluster Kubernetes Docker Stats Endpoint	Detects the misuse of the Kubernetes docker stats endpoint.	Enabled
Runtime	Process Targeting Kubernetes Service Endpoint	Detects the misuse of the Kubernetes Service API endpoint.	Enabled
Runtime	Process with UID 0	Alerts when deployments contain processes that run with UID 0.	Disabled
Runtime	Secure Shell Server (sshd) Execution	Detects containers that run the SSH daemon.	Enabled
Runtime	SetUID Processes	Use setuid binary files, which permit people to run certain programs with escalated privileges.	Disabled
Runtime	Shadow File Modification	Indicates when someone tries to modify shadow files.	Disabled
Runtime	Shell Spawned by Java Application	Detects when a shell, such as bash, csh, sh, or zsh, is run as a subprocess of a Java application.	Enabled
Runtime	Unauthorized Network Flow	Generates a violation for any network flows that fall outside of the baselines of the "alert on anomalous violations" setting.	Enabled
Runtime	Unauthorized Processed Execution	Generates a violation for any process execution that is not explicitly allowed by a locked process baseline for a container specification in a Kubernetes deployment.	Enabled

7.3. MEDIUM SEVERITY SECURITY POLICIES

The following table lists the default security policies in Red Hat Advanced Cluster Security for Kubernetes that are of medium severity. The policies are organized by life cycle stage.

Table 7.3. Medium severity security policies

Life cycle stage	Name	Description	Status
Build	Docker CIS 4.4: Ensure images are scanned and rebuilt to include security patches	Alerts when images are not scanned and rebuilt to include security patches. It is important to scan images often to find vulnerabilities, rebuild the images to include security patches, and then instantiate containers for the images.	Disabled
Deploy	30-Day Scan Age	Alerts when a deployment has not been scanned in 30 days.	Enabled
Deploy	CAP_SYS_ADMIN capability added	Alerts when a deployment includes containers that are escalating with CAP_SYS_ADMIN.	Enabled
Deploy	Container using read-write root filesystem	Alerts when a deployment includes containers that have read-write root file systems.	Disabled
Deploy	Container with privilege escalation allowed	Alerts when a container might be running with unintended privileges, creating a security risk. This situation can happen when a container process that has more privileges than its parent process allows the container to run with unintended privileges.	Enabled
Deploy	Deployments should have at least one Ingress Network Policy	Alerts if deployments are missing an Ingress Network Policy.	Disabled

Life cycle stage	Name	Description	Status
Deploy	Deployments with externally exposed endpoints	Detects if a deployment has any service that is externally exposed through any methods. Deployments with services exposed outside of the cluster are at a higher risk of attempted intrusions because they are reachable outside of the cluster. This policy provides an alert so that you can verify that service exposure outside of the cluster is required. If the service is only needed for intra-cluster communication, use service type ClusterIP.	Disabled
Deploy	Docker CIS 5.1: Ensure that, if applicable, an AppArmor profile is enabled	Uses AppArmor to protect the Linux operating system and applications by enforcing a security policy that is known as an AppArmor profile. AppArmor is a Linux application security system that is available on some Linux distributions by default, such as Debian and Ubuntu.	Enabled
Deploy	Docker CIS 5.15: Ensure that the host’s process namespace is not shared	Creates process-level isolation between the containers and the host. The Process ID (PID) namespace isolates the process ID space, which means that processes in different PID namespaces can have the same PID.	Enabled

Life cycle stage	Name	Description	Status
Deploy	Docker CIS 5.16: Ensure that the host’s IPC namespace is not shared	Alerts when the IPC namespace on the host is shared with containers. The IPC (POSIX/SysV IPC) namespace separates named shared memory segments, semaphores, and message queues.	Enabled
Deploy	Docker CIS 5.19: Ensure mount propagation mode is not enabled	Alerts when mount propagation mode is enabled. When mount propagation mode is enabled, you can mount container volumes in Bidirectional, Host to Container, and None modes. Do not use Bidirectional mount propagation mode unless it is explicitly needed.	Enabled
Deploy	Docker CIS 5.21: Ensure the default seccomp profile is not disabled	Alerts when the seccomp profile is disabled. The seccomp profile uses an allowlist to permit common system calls and blocks all others.	Disabled
Deploy	Docker CIS 5.7: Ensure privileged ports are not mapped within containers	Alerts when privileged ports are mapped within containers. The TCP/IP port numbers that are lower than 1024 are privileged ports. Normal users and processes can not use them for security reasons, but containers might map their ports to privileged ports.	Enabled

Life cycle stage	Name	Description	Status
Deploy	Docker CIS 5.9 and 5.20: Ensure that the host’s network namespace is not shared	Alerts when the host’s network namespace is shared. When HostNetwork is enabled, the container is not placed inside a separate network stack, and the container’s networking is not containerized. As a result, the container has full access to the host’s network interfaces, and a shared UTS namespace is enabled. The UTS namespace provides isolation between the hostname and the NIS domain name, and it sets the hostname and the domain, which are visible to running processes in that namespace. Processes that run within containers do not typically require to know the hostname or the domain name, so the UTS namespace should not be shared with the host.	Enabled
Deploy	Images with no scans	Alerts when a deployment includes images that were not scanned.	Disabled
Runtime	Kubernetes Actions: Port Forward to Pod	Alerts when the Kubernetes API receives a port forward request.	Enabled
Deploy	Mount Container Runtime Socket	Alerts when a deployment has a volume mount on the container runtime socket.	Enabled

Life cycle stage	Name	Description	Status
Deploy	Mounting Sensitive Host Directories	Alerts when a deployment mounts sensitive host directories.	Enabled
Deploy	No resource requests or limits specified	Alerts when a deployment includes containers that do not have resource requests and limits.	Enabled
Deploy	Pod Service Account Token Automatically Mounted	Protects pod default service account tokens from being compromised by minimizing the mounting of the default service account token to only those pods whose applications require interaction with the Kubernetes API.	Enabled
Deploy	Privileged Container	Alerts when a deployment includes containers that run in privileged mode.	Enabled
Runtime	crontab Execution	Detects the usage of the crontab scheduled jobs editor.	Enabled
Runtime	Netcat Execution Detected	Detects when netcat runs in a container.	Enabled
Runtime	OpenShift: Advanced Cluster Security Central Admin Secret Accessed	Alerts when someone accesses the Red Hat Advanced Cluster Security Central secret.	Enabled
Runtime	OpenShift: Kubernetes Secret Accessed by an Impersonated User	Alerts when someone impersonates a user to access a secret in the cluster.	Enabled
Runtime	Remote File Copy Binary Execution	Alerts when a deployment runs a remote file copy tool.	Enabled

7.4. LOW SEVERITY SECURITY POLICIES

The following table lists the default security policies in Red Hat Advanced Cluster Security for Kubernetes that are of low severity. The policies are organized by life cycle stage.

Table 7.4. Low severity security policies

Life cycle stage	Name	Description	Status
Build or Deploy	90-Day Image Age	Alerts when a deployment has not been updated in 90 days.	Enabled
Build or Deploy	ADD Command used instead of COPY	Alerts when a deployment uses an ADD command.	Disabled
Build or Deploy	Alpine Linux Package Manager (apk) in Image	Alerts when a deployment includes the Alpine Linux package manager (apk).	Enabled
Build or Deploy	Curl in Image	Alerts when a deployment includes curl.	Disabled
Build or Deploy	Docker CIS 4.1: Ensure That a User for the Container Has Been Created	Ensures that containers are running as non-root users.	Enabled
Build or Deploy	Docker CIS 4.7: Alert on Update Instruction	Ensures that update instructions are not used alone in the Dockerfile.	Enabled
Build or Deploy	Insecure specified in CMD	Alerts when a deployment uses 'insecure' in the command.	Enabled
Build or Deploy	Latest tag	Alerts when a deployment includes images that use the 'latest' tag.	Enabled
Build or Deploy	Red Hat Package Manager in Image	Alerts when a deployment includes components of the Red Hat, Fedora, or CentOS package management system.	Enabled

Life cycle stage	Name	Description	Status
Build or Deploy	Required Image Label	Alerts when a deployment includes images that are missing the specified label.	Disabled
Build or Deploy	Ubuntu Package Manager in Image	Alerts when a deployment includes components of the Debian or Ubuntu package management system in the image.	Enabled
Build or Deploy	Wget in Image	Alerts when a deployment includes wget.	Disabled
Deploy	Drop All Capabilities	Alerts when a deployment does not drop all capabilities.	Disabled
Deploy	Improper Usage of Orchestrator Secrets Volume	Alerts when a deployment uses a Dockerfile with 'VOLUME /run/secrets'.	Enabled
Deploy	Kubernetes Dashboard Deployed	Alerts when a Kubernetes dashboard service is detected.	Enabled
Deploy	Required Annotation: Email	Alerts when a deployment is missing the 'email' annotation.	Disabled
Deploy	Required Annotation: Owner/Team	Alerts when a deployment is missing the 'owner' or 'team' annotation.	Disabled
Deploy	Required Label: Owner/Team	Alerts when a deployment is missing the 'owner' or 'team' label.	Disabled
Runtime	Alpine Linux Package Manager Execution	Alerts when the Alpine Linux package manager (apk) is run at run time.	Enabled

Life cycle stage	Name	Description	Status
Runtime	chkconfig Execution	Detects the usage of the ckconfig service manager, which is typically not used in a container.	Enabled
Runtime	Compiler Tool Execution	Alerts when binary files that compile software are run at run time.	Enabled
Runtime	Red Hat Package Manager Execution	Alerts when Red Hat, Fedora, or CentOS package manager programs are run at run time.	Enabled
Runtime	Shell Management	Alerts when commands are run to add or remove a shell.	Disabled
Runtime	systemctl Execution	Detects the usage of the systemctl service manager.	Enabled
Runtime	systemd Execution	Detects the usage of the systemd service manager.	Enabled

CHAPTER 8. MANAGING NETWORK POLICIES

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.

Red Hat Advanced Cluster Security for Kubernetes (RHACS) gathers all defined network policies from your orchestrator and provides tools to make these policies easier to use.

To support network policy enforcement, RHACS provides the following tools:

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

8.1. NETWORK GRAPH

8.1.1. About the network graph

The network graph provides high-level and detailed information about deployments, network flows, and network policies in your environment.

RHACS processes all network policies in each secured cluster to show you which deployments can contact each other and which can reach external networks. It also monitors running deployments and tracks traffic between them. You can view the following items in the network graph:

External entities

These represent entities that are connected outside of your cluster. For more information, see "External entities and connections in the network graph".

Internal entities

These represent entities that are connected inside of your cluster and are not considered external. Internal entities are displayed when Sensor is unable to match connections to existing deployments, but the involved IP address is part of the cluster’s internal networks. Based on the IP addresses, the system has determined that those flows do not leave the cluster. For more information about the kind of connections that are considered internal, see "Connections involving internal entities".

Network components

From the top menu, you can select namespaces (indicated by the **NS** label) and deployments (indicated by the **D** label) to display on the graph for a chosen cluster (indicated by the **CL** label). You can further filter deployments by using the drop-down list and selecting criteria on which to filter, such as common vulnerabilities and exposures (CVEs), labels, and images.

Network flows

You can select one of the following flows for the graph:

Active traffic

Selecting this default option shows observed traffic, focused on the namespace or specific deployment that you selected. You can select the time period for which to display information.

Inactive flows

Selecting this option shows potential flows allowed by your network policies, helping you identify missing network policies needed to achieve tighter isolation. You can select the time period for which to display information.

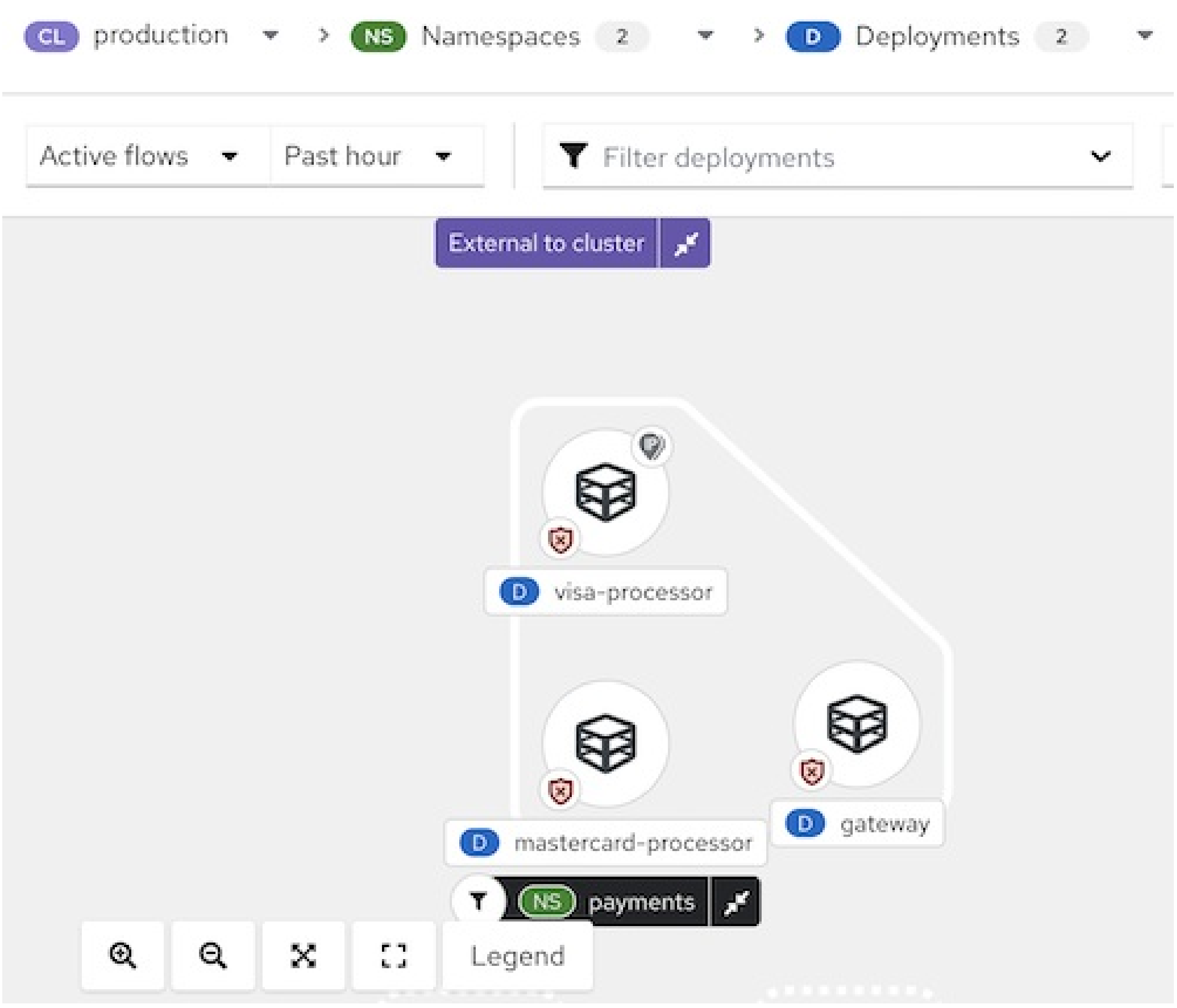
Network policies

You can view existing policies for a selected component or view components that have no policies. You can also simulate network policies from the network graph view. See "Simulating network policies from the network graph" for more information.

8.1.1.1. Displays, navigation, and the user interface in the network graph

You can use the network graph, shown in the following graphic, to click on items and view additional information about them. You can also perform actions within the graph such as adding a network flow to your baseline.

Figure 8.1. Network graph example



The following tips can help you use the network graph:

- Opening the legend provides information about the symbols in use and their meaning. The legend shows explanatory text for symbols representing namespaces, deployments, and connections on the network graph.

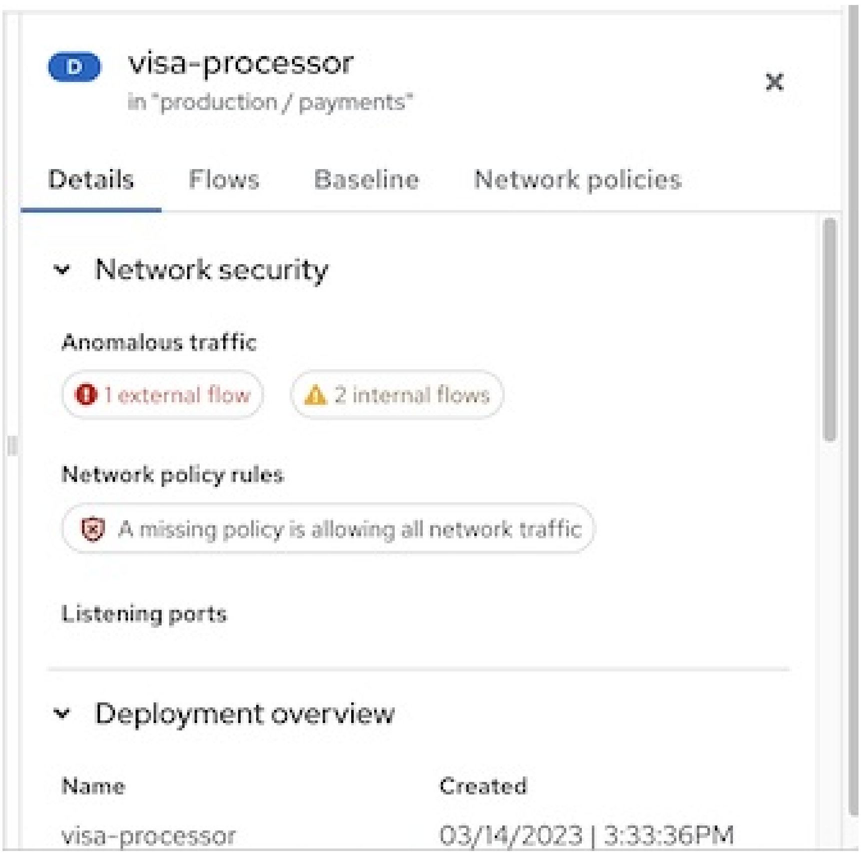
- Selecting additional display options from the drop-down list controls whether the graph displays icons such as the network policy status badge, active external traffic badge, and port and protocol labels for edge connections.
- RHACS detects changes in network traffic, such as nodes joining or leaving. If changes are detected, the network graph displays a notification showing the number of updates available. To avoid interrupting your focus, the graph is not updated automatically. Click the notification to update the graph.

When you click an item in the graph, the rearranged side panel with collapsible sections presents information about that item. You can click on the following items:

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

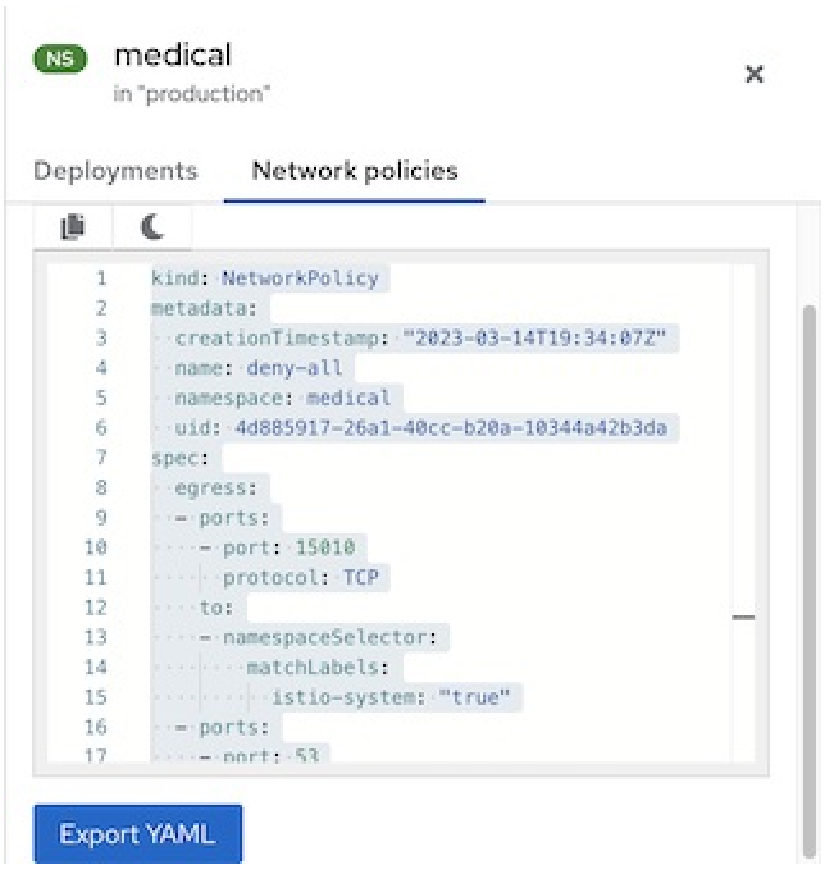
The side panel displays relevant information based on the item in the graph that you have selected. The **D** or **NS** label next to the item name in the header (in this example, "visa-processor") indicates if it is a deployment or a namespace. The following example illustrates the side panel for a deployment.

Figure 8.2. Side panel for a deployment example



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

Figure 8.3. Side panel for a namespace example



```
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
```

8.1.1.2. External entities and connections in the network graph

The network graph view shows network connections between managed clusters and external sources. In addition, RHACS automatically discovers and highlights public Classless Inter-Domain Routing (CIDR) address blocks, such as Google Cloud, AWS, Microsoft Azure, Oracle Cloud, and Cloudflare. Using this information, you can identify deployments with active external connections and decide if they are making or receiving unauthorized connections from outside your network.

By default, the external connections point to a common **External Entities** icon and different CIDR address blocks in the network graph. However, you can choose not to show auto-discovered CIDR blocks by clicking **Manage CIDR blocks** and deselecting **Auto-discovered CIDR blocks**.

RHACS includes IP ranges for the following cloud providers:

- Google Cloud
- AWS
- Microsoft Azure
- Oracle Cloud
- Cloudflare

RHACS fetches and updates the cloud providers' IP ranges every 7 days, and updates CIDR blocks daily. If you are using offline mode, you can update these ranges by installing new support packages.

The following image provides an example of the network graph. In this example, based on the options that the user has chosen, the graph depicts deployments in the selected namespace. Traffic flows are not displayed until you click on an item such as a deployment. The graph uses a red badge to indicate deployments that are missing policies and therefore allowing all network traffic.

8.1.1.3. Connections involving internal entities

The network graph is useful for identifying deployments with active connections to entities that do not belong to any known deployment or CIDR block. Some of these connections never reach outside of the cluster and are made within the cluster's private network. The network graph represents those as connections to or from *internal entities*.

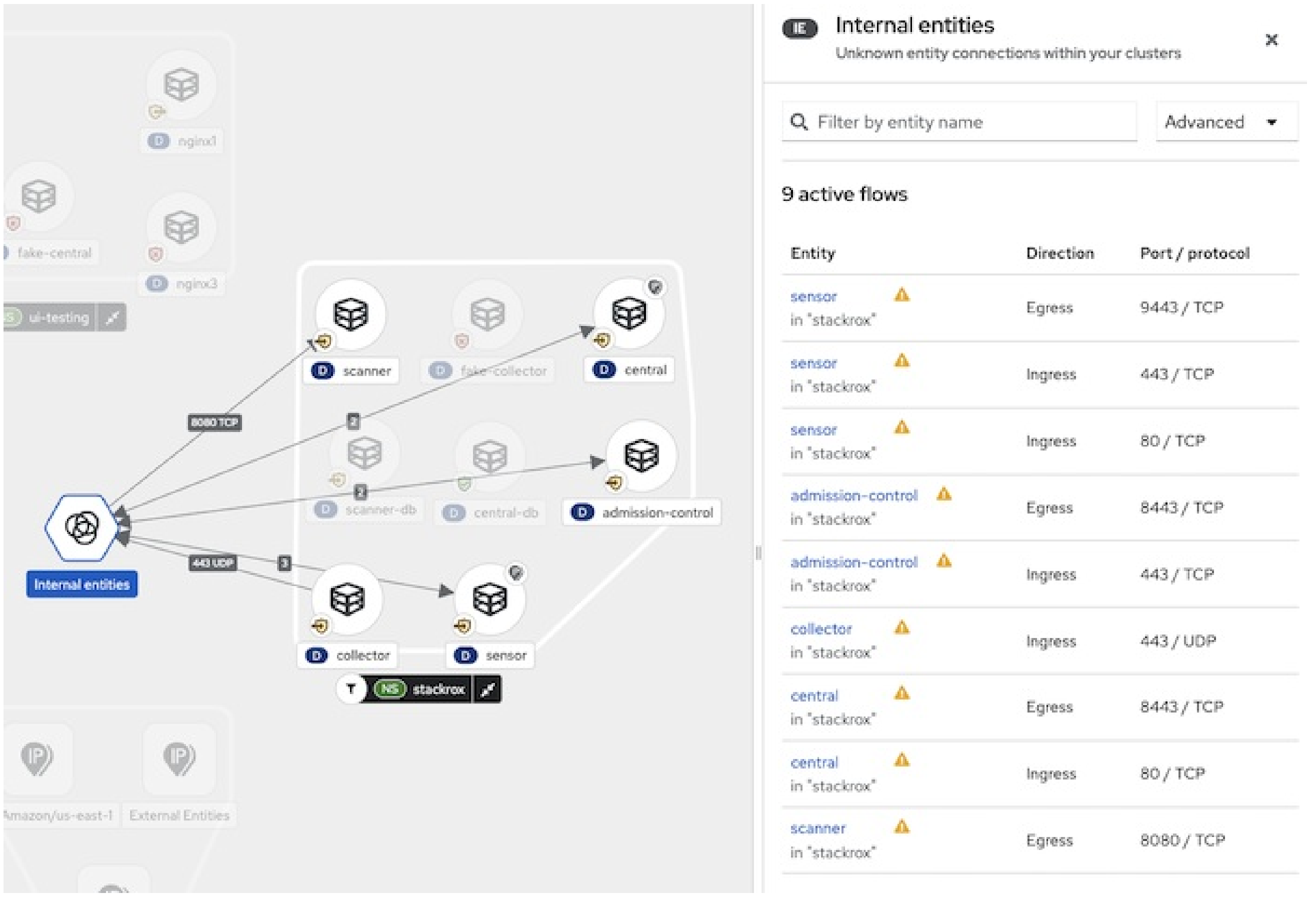
Connections with internal entities represent a connection between a deployment and an IP address that belongs to the private address space as defined in RFC 1918. In some cases, Sensor is unable to identify one or both deployments involved in a connection. In that case, the system analyzes the IP address and decides whether the connection is internal or external.

The following scenarios can lead to a connection being categorized as one involving internal entities:

- A change of IP address or the deletion of a deployment accepting connections (the server) while the party initiating the connection (the client) still attempts to reach it
- A deployment communicating with the orchestrator API
- A deployment communicating using a networking CNI plugin, for example, Calico
- A restart of Sensor, resulting in a reset of the mapping of IP addresses to past deployments, for example, when Sensor does not recognize the IP addresses of past entities or past IP addresses of existing entities

Internal entities are indicated with an icon as shown in the following graphic. Clicking on **Internal entities** shows the flows for these entities.

Figure 8.4. Internal entities example



Internal entities
Unknown entity connections within your clusters

9 active flows

Entity	Direction	Port / protocol
sensor in "stackrox"	Egress	9443 / TCP
sensor in "stackrox"	Ingress	443 / TCP
sensor in "stackrox"	Ingress	80 / TCP
admission-control in "stackrox"	Egress	8443 / TCP
admission-control in "stackrox"	Ingress	443 / TCP
collector in "stackrox"	Ingress	443 / UDP
central in "stackrox"	Egress	8443 / TCP
central in "stackrox"	Ingress	80 / TCP
scanner in "stackrox"	Egress	8080 / TCP

8.1.2. Access control and permissions

To view network graphs, the user must have at least the permissions granted to the **Network Graph Viewer** default permission set.

The following permissions are granted to the **Network Graph Viewer** permission set:

- Read **Deployment**
- Read **NetworkGraph**
- Read **NetworkPolicy**

For more information, see "System permission sets" in the "Additional resources" section.

Additional resources

- System permission sets

8.1.3. Viewing deployment information

The network graph provides a visual map of deployments, namespaces, and connections that RHACS has discovered. By clicking on a deployment in the graph, you can view information about the deployment, including the following details:

- Network security, such as the number of flows, existing or missing network policy rules, and listening ports
- Labels and annotations
- Port configurations
- Container information
- Anomalous and baseline flows for ingress and egress connections, including protocols and port numbers
- Network policies

Procedure

To view details for deployments in a namespace:

1. In the RHACS 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.

8.1.4. Viewing network policies in the network graph

Network policies specify how groups of pods are allowed to communicate with each other and with other network endpoints. Kubernetes **NetworkPolicy** resources use labels to select pods and define rules that specify what traffic is allowed to or from the selected pods. RHACS discovers and displays network policy information for all your Kubernetes clusters, namespaces, deployments, and pods, in the network graph.

Procedure

1. In the RHACS 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, in the **Network security** section, you can view summary messages about network policy rules that give the following information:
    - If policies exist in the network that regulate ingress or egress traffic
    - If your network is missing policies and is therefore allowing all ingress or egress traffic
6. To view the YAML file for the network policies, you can click on the policy rule, or click the **Network policies** tab.

8.1.5. Configuring CIDR blocks in the network graph

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

Procedure

1. In the RHACS portal, go to **Network Graph**, and then select **Manage CIDR Blocks**. You can perform the following actions:
    - Toggle **Auto-discovered CIDR blocks** to hide auto-discovered CIDR blocks in the network graph.

      NOTE

      When you hide the auto-discovered CIDR blocks, the auto-discovered CIDR blocks are hidden for all clusters, and not only for the selected cluster in the network graph.

    - Add a custom CIDR block to the graph by performing the following steps:
        a. Enter the CIDR name and CIDR address in the fields. To add additional CIDR blocks, click **Add CIDR block** and enter information for each block.
        b. Click **Update Configuration** to save the changes.

8.2. USING THE NETWORK GRAPH TO GENERATE AND SIMULATE NETWORK POLICIES

8.2.1. About generating policies from the network graph

A Kubernetes network policy controls which pods receive incoming network traffic, and which pods can send outgoing traffic. By using network policies to enable and disable traffic to or from pods, you can limit your network attack surface.

These network policies are YAML configuration files. It is often difficult to gain insights into the network flow and manually create these files. You can use RHACS to generate these files. When you automatically generate network policies, RHACS follows these guidelines:

- RHACS generates a single network policy for each deployment in the namespace. The pod selector for the policy is the pod selector of the deployment.
  - If a deployment already has a network policy, RHACS does not generate new policies or delete existing policies.
    Generated policies only restrict traffic to existing deployments.
  - Deployments that you create later will not have any restrictions unless you create or generate new network policies for them.
  - If a new deployment needs to contact a deployment with a network policy, you might need to edit the network policy to allow access.
- Each policy has the same name as the deployment name, prefixed with **stackrox-generated-**. For example, the policy name for the deployment **depABC** in the generated network policy is **stackrox-generated-depABC**. All generated policies also have an identifying label.
- RHACS generates a single rule allowing traffic from any IP address if one of the following conditions are met:
  - The deployment has an incoming connection from outside the cluster within the selected time
  - The deployment is exposed through a node port or load balancer service
- RHACS generates one **ingress** rule for every deployment from which there is an incoming connection.
  - For deployments in the same namespace, this rule uses the pod selector labels from the other deployment.
  - For deployments in different namespaces, this rule uses a namespace selector. To make this possible, RHACS automatically adds a label, **namespace.metadata.stackrox.io/name**, to each namespace.

IMPORTANT

In rare cases, if a standalone pod does not have any labels, the generated policy allows traffic from or to the pod’s entire namespace.

8.2.2. Generating network policies in the network graph

RHACS lets you automatically generate network policies based on the actual observed network communication flows in your environment.

You can generate policies based on the cluster, namespaces, and deployments that you have selected in the network graph. Policies are generated for any deployments that are included in the current Network Graph scope. For example, the current scope could include the entire cluster, a cluster and namespaces, or individually selected deployments in the selected namespaces. You can also further reduce the scope by applying one of the filters from the **Filter deployments** field with any combination of the cluster, namespace, and deployment selections. For example, you could narrow the scope to deployments in a specific cluster and namespace that are affected by a specific CVE. Policies are generated from the traffic observed during the baseline discovery period.

1. In the RHACS portal, go to **Network Graph**.
2. Select a cluster, and then select one or more namespaces.
3. Optional: Select individual deployments to restrict the policy generated to only those deployments. You can also use the **Filter deployments** feature to further narrow the scope.
4. In the network graph header, select **Network policy generator**.
5. Optional: In the information panel that opens, select **Exclude ports & protocols**to remove the port/protocol restrictions when generating network policies from a baseline.
   As an example, the **nginx3** deployment makes a port 80 connection to **nginx4**, and this is included as part of the baseline for **nginx4**. If policies are generated and this checkbox is not selected (the default behavior), the generated policy will restrict the allowed connections from **nginx3** to **nginx4** to only port 80. If policies are generated with this option selected, the generated policy will allow any port in the connection from **nginx3** to **nginx4**.
6. Click **Generate and simulate network policies**. RHACS generates policies for the scope that you have chosen. This scope is displayed at the top of the **Generate network policies** panel.

   NOTE

   Clicking on the deployment information in the scope displays a list of the deployments that are included.

7. Optional: Copy the generated network policy configuration YAML file to the clipboard or download it by clicking the download icon in the panel.
8. Optional: To compare the generated network policies to the existing network policies, click **Compare**. The YAML files for existing and generated network policies are shown in a side-by-side view.

   NOTE

   Some items do not have generated policies, such as namespaces with existing ingress policies or deployments in certain protected namespaces such as as **stackrox** or **acs**.

9. Optional: Click the **Actions** menu to perform the following activities:
   - Share the YAML file with notifiers: Sends the YAML file to one of the system notifiers you have configured, for example, Slack, ServiceNow, or an application that uses generic webhooks. These notifiers are configured by navigating to **Platform Configuration →**

Integrations. See the documentation in the "Additional resources" section for more information.

- Rebuild rules from active traffic: Refreshes the generated policies that are displayed.
- Revert rules to previously applied YAML: Removes the simulated policy and reverts to the last network policy.

8.2.3. Saving generated policies in the network graph

You can download and save the generated network policies from RHACS. Use this option to download policies so that you can commit the policies into a version control system such as Git.

Procedure

- After generating a network policy, click the **Download YAML** icon in the **Network Policy Simulator** panel.

8.2.4. Testing generated policies in the network graph

After you download the network policies that RHACS generates, you can test them by applying them to your cluster by using the CLI or your automated deployment procedures. You cannot apply generated network policies directly in the network graph.

Procedure

1. To create policies using the saved YAML file, run the following command:

   ```
   $ oc create -f "<generated_file>.yml" 1
   ```

   1 If you use Kubernetes, enter **kubectl** instead of **oc**.

2. If the generated policies cause problems, you can remove them by running the following command:

   ```
   $ oc delete -f "<generated_file>.yml" 1
   ```

   1 If you use Kubernetes, enter **kubectl** instead of **oc**.

WARNING

Directly applying network policies might cause problems for running applications. Always download and test the network policies in a development environment or test clusters before applying them to production workloads.

8.2.5. Reverting to a previously applied policy in the network graph

You can remove a policy and revert to a previously applied policy.

Procedure

1. In the RHACS portal, go to **Network Graph**.
2. Select a cluster name from the menu on the top bar.
3. Select one or more namespaces and deployments.
4. Select **Simulate network policy**.
5. Select **View active YAMLS**.
6. From the **Actions** menu, select **Revert rules to previously applied YAML**

WARNING

Directly applying network policies might cause problems for running applications. Always download and test the network policies in a development environment or test clusters before applying them to production workloads.

8.2.6. Deleting all policies autogenerated in the network graph

You can delete all automatically generated policies from your cluster that you have created by using RHACS.

Procedure

- Run the following command:

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

1 If you use Kubernetes, enter **kubectl** instead of **oc**.

8.2.7. Simulating network policies from the network graph

Your current network policies might allow unneeded network communications. You can use the network policy generator to create network policies that restrict ingress traffic to the computed baselines for a set of deployments.

NOTE

The Network Graph does not display the generated policies in the visualization. Generated policies are only for ingress traffic and policies that restrict egress traffic are not generated.

Procedure

1. In the RHACS portal, go to **Network Graph**.
2. Select a cluster, and then select one or more namespaces.
3. On the network graph header, select **Network policy generator**.
4. Optional: To generate a YAML file with network policies to use in the simulation, click **Generate and simulate network policies**. For more information, see "Generating network policies in the network graph".
5. Upload a YAML file of a network policy that you want to use in the simulation. The network graph view displays what your proposed network policies would achieve. Perform the following steps:
   a. Click **Upload YAML** and then select the file.
   b. Click **Open**. The system displays a message to indicate the processing status of the uploaded policy.
6. You can view active YAML files that correspond to the current network policies by clicking the **View active YAMLS** tab, and then selecting policies from the drop-down list. You can also perform the following actions:
   - Click the appropriate button to copy or download the displayed YAML file.
   - Use the **Actions** menu to rebuild rules from active traffic or revert rules to a previously applied YAML. For more information, see "Generating network policies in the network graph".

Additional resources

- Updating kernel support packages in offline mode
- Integrating using generic webhooks

8.3. ABOUT NETWORK BASELINING IN THE NETWORK GRAPH

In RHACS, you can minimize your risks by using network baselining. It is a proactive approach to keep your infrastructure secure. RHACS first discovers existing network flows and creates a baseline, and then it treats network flows outside of this baseline as anomalous.

When you install RHACS, there is no default network baseline. As RHACS discovers network flows, it creates a baseline and then it adds all discovered network flows to it, following these guidelines:

- When RHACS discovers new network activity, it adds that network flow to the network baseline.
- Network flows do not show up as anomalous flows and do not trigger any violations.

After the discovery phase, the following actions occur:

- RHACS stops adding network flows to the network baselines.
- New network flows that are not in the network baseline show up as anomalous flows but they do not trigger any violations.

8.3.1. Viewing network baselines from the network graph

You can view network baselines from the network graph view.

Procedure

1. Click the **Namespaces** list and use the search field to locate a namespace, or select individual namespaces.
2. Click the **Deployments** list and use the search field to locate a deployment, or select individual deployments to display in the network graph.
3. In the network graph, click on a deployment to view the information panel.
4. Select the **Baseline** tab. Use the **filter by entity name** field to further restrict the flows that are displayed.
5. Optional: You can mark baseline flows as anomalous by performing one of the following actions:
   - Select an individual entity. Click the overflow menu, ⋮, and then select **Mark as anomalous**.
   - Select multiple entities, and then click **Bulk actions** and select **Mark as anomalous**.
6. Optional: Check the box to exclude ports and protocols.
7. Optional: To save the baseline as a network policy YAML file, click **Download baseline as network policy**.

8.3.2. Downloading network baselines from the network graph

You can download network baselines as YAML files from the network graph view.

Procedure

1. In the RHACS portal, go to **Network Graph**.
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. The **Baseline** tab lists the baseline flows. Use the **filter by entity name** field to further restrict the list of flows.
6. Optional: Check the box to exclude ports and protocols.
7. Click **Download baseline as network policy**.

8.3.3. Configuring network baselining time frame

You can use the **ROX_NETWORK_BASELINE_OBSERVATION_PERIOD** and the **ROX_BASELINE_GENERATION_DURATION** environment variables to configure the observation period and the network baseline generation duration.

Procedure

1. Set the **ROX_NETWORK_BASELINE_OBSERVATION_PERIOD** environment variable by running the following command:

```
$ oc -n stackrox set env deploy/central \ 1
  ROX_NETWORK_BASELINE_OBSERVATION_PERIOD=<value> 2
```

1 If you use Kubernetes, enter **kubectl** instead of **oc**.

2 Value must be time units, for example: **300ms**, **-1.5h**, or **2h45m**. Valid time units are **ns**, **us** or **µs**, **ms**, **s**, **m**, **h**.

2. Set the **ROX_BASELINE_GENERATION_DURATION** environment variable by running the following command:

```
$ oc -n stackrox set env deploy/central \ 1
  ROX_BASELINE_GENERATION_DURATION=<value> 2
```

1 If you use Kubernetes, enter **kubectl** instead of **oc**.

2 Value must be time units, for example: **300ms**, **-1.5h**, or **2h45m**. Valid time units are **ns**, **us** or **µs**, **ms**, **s**, **m**, **h**.

8.3.4. Enabling alerts on baseline violations in the network graph

You can configure RHACS to detect anomalous network flows and trigger violations for traffic that is not in the baseline. This can help you determine if the network contains unwanted traffic before you block traffic with a network policy.

Procedure

1. Click the **Namespaces** list and use the search field to locate a namespace, or select individual namespaces.
2. Click the **Deployments** list and use the search field to locate a deployment, or select individual deployments to display in the network graph.
3. In the network graph, click on a deployment to view the information panel.
4. In the **Baseline** tab, you can view baseline flows. Use the **filter by entity name** field to further restrict the flows that are displayed.
5. Toggle the **Alert on baseline violations** option.
   - After you toggle the **Alert on baseline violations** option, anomalous network flows trigger violations.

- You can toggle the **Alert on baseline violations** option again to stop receiving violations for anomalous network flows.

CHAPTER 9. BUILD-TIME NETWORK POLICY TOOLS

Build-time network policy tools let you automate the creation and validation of Kubernetes network policies in your development and operations workflows using the **roxctl** CLI. These tools work with a specified file directory containing your project's workload and network policy manifests and do not require RHACS authentication.

Table 9.1. Network policy tools

Command	Description
roxctl netpol generate	Generates Kubernetes network policies by analyzing your project's YAML manifests in a specified directory. For more information, see Using the build-time network policy generator.
roxctl netpol connectivity map	Lists the allowed connections between workloads in your project directory by examining the workload and Kubernetes network policy manifests. You can generate the output in various text formats or in a graphical **.dot** format. For more information, see Connectivity mapping using the roxctl netpol connectivity map command.
roxctl netpol connectivity diff	Creates a list of variations in the allowed connections between two project versions. This is determined by the workload and Kubernetes network policy manifests in each version's directory. This feature shows the semantic differences which are not obvious when performing a source code (syntactic) **diff**. For more information, seeIdentifying the differences in allowed connections between project versions.

9.1. USING THE BUILD-TIME NETWORK POLICY GENERATOR

The build-time network policy generator can automatically generate Kubernetes network policies based on application YAML manifests. You can use it to develop network policies as part of the continuous integration/continuous deployment (CI/CD) pipeline before deploying applications on your cluster.

Red Hat developed this feature in partnership with the developers of the NP-Guard project. First, the build-time network policy generator analyzes Kubernetes manifests in a local folder, including service manifests, config maps, and workload manifests such as **Pod**, **Deployment**, **ReplicaSet**, **Job**, **DaemonSet**, and **StatefulSet**. Then, it discovers the required connectivity and creates the Kubernetes network policies to achieve pod isolation. These policies allow no more and no less than the needed ingress and egress traffic.

9.1.1. 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.

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 hard-coded. 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>)? (1)
   ```

   (1) 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**
   - **http://rating-service.**

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> [flags] (1)
   ```

1 Specify the path to the folder, which can include sub-folders that contain YAML resources for analysis. The command scans the entire sub-folder tree. Optionally, you can also specify parameters to modify the behavior of the command.

For more information about optional parameters, see roxctl netpol generate command options .

Next steps

- After generating the policies, you must inspect them for completeness and accuracy, in case any relevant network address was not specified as expected in the YAML files.
- Most importantly, verify that required connections are not blocked by the isolating policies. To help with this inspection you can use the **roxctl netpol connectivity map** tool.

NOTE

Applying network policies to the cluster as part of the workload deployment using automation saves time and ensures accuracy. You can follow a GitOps approach by submitting the generated policies using pull requests, providing the team an opportunity to review the policies before deploying them as part of the pipeline.

9.1.2. roxctl netpol generate command options

The **roxctl netpol generate** command supports the following options:

Option	Description
-h, --help	View the help text for the **netpol** command.
-d, --output-dir <dir>	Save the generated policies into a target folder. One file per policy.
-f, --output-file <filename>	Save and merge the generated policies into a single YAML file.
--fail	Fail on the first encountered error. The default value is **false**.
--remove	Remove the output path if it already exist.
--strict	Treat warnings as errors. The default value is **false**.

9.2. CONNECTIVITY MAPPING USING THE ROXCTL NETPOL CONNECTIVITY MAP COMMAND

Connectivity mapping provides details on the allowed connections between different workloads based on network policies defined in Kubernetes manifests. You can visualize and understand how different workloads in your Kubernetes environment are allowed to communicate with each other according to the network policies you set up.

To retrieve connectivity mapping information, the **roxctl netpol connectivity map** command requires a directory path that contains Kubernetes workloads and network policy manifests. The output provides details about connectivity details within the Kubernetes resources analyzed.

9.2.1. Retrieving connectivity mapping information from a Kubernetes manifest directory

Procedure

- Run the following command to retrieve the connectivity mapping information:

```
$ roxctl netpol connectivity map <folder_path> [flags] 1
```

1 Specify the path to the folder, which can include sub-folders that contain YAML resources and network policies for analysis, for example, **netpol-analysis-example-minimal/**. The command scans the entire sub-folder tree. Optionally, you can also specify parameters to modify the behavior of the command.

For more information about optional parameters, see roxctl netpol connectivity map command options.

Example 9.1. Example output

src	dst	conn
0.0.0.0-255.255.255.255	default/frontend[Deployment]	TCP 8080
default/frontend[Deployment]	0.0.0.0-255.255.255.255	UDP 53
default/frontend[Deployment]	default/backend[Deployment]	TCP 9090

The output shows you a table with a list of allowed connectivity lines. Each connectivity line consists of three parts: source (**src**), destination (**dst**), and allowed connectivity attributes (**conn**).

You can interpret **src** as the source endpoint, **dst** as the destination endpoint, and **conn** as the allowable connectivity attributes. An endpoint has the format **namespace/name[Kind]**, for example, **default/backend[Deployment]**.

9.2.2. Connectivity map output formats and visualizations

You can use various output formats, including **txt**, **md**, **csv**, **json**, and **dot**. The **dot** format is ideal for visualizing the output as a connectivity graph. It can be viewed using graph visualization software such as Graphviz tool, and extensions to VSCode. You can convert the **dot** output to formats such as **svg**, **jpeg**, or **png** using Graphviz, whether it is installed locally or through an online viewer.

9.2.3. Generating svg graphs from the dot output using Graphviz

Follow these steps to create a graph in **svg** format from the **dot** output.

Prerequisites

- Graphviz is installed on your local system.

Procedure

- Run the following command to create the graph in **svg** format:

  ```
  $ dot -Tsvg connlist_output.dot > connlist_output_graph.svg
  ```

  The following are examples of the dot output and the resulting graph generated by Graphviz:

  - Example 1: dot output
  - Example 2: Graph generated by Graphviz

9.2.4. roxctl netpol connectivity map command options

The **roxctl netpol connectivity map** command supports the following options:

Option	Description
--fail	Fail on the first encountered error. The default value is **false**.
--focus-workload string	Focus on connections of a specified workload name in the output.
-h, **--help**	View the help text for the **roxctl netpol connectivity map** command.
-f, **--output-file string**	Save the connections list output into a specific file.
-o, **--output-format string**	Configure the output format. The supported formats are **txt**, **json**, **md**, **dot**, and **csv**. The default value is **txt**.
--remove	Remove the output path if it already exists. The default value is **false**.
--save-to-file	Save the connections list output into a default file. The default value is **false**.
--strict	Treat warnings as errors. The default value is **false**.

9.3. IDENTIFYING THE DIFFERENCES IN ALLOWED CONNECTIONS BETWEEN PROJECT VERSIONS

This command helps you understand the differences in allowed connections between two project versions. It analyses the workload and Kubernetes network policy manifests located in each version’s directory and creates a representation of the differences in text format.

You can view connectivity difference reports in a variety of output formats, including **text**, **md**, and **csv**.

9.3.1. Generating connectivity difference reports with the roxctl netpol connectivity diff command

To produce a connectivity difference report, the **roxctl netpol connectivity diff** command requires two folders, **dir1** and **dir2**, each containing Kubernetes manifests, including network policies.

Procedure

- Run the following command to determine the connectivity differences between the Kubernetes manifests in the specified directories:

```
$ roxctl netpol connectivity diff --dir1=<folder_path_1> --dir2=<folder_path_2> [flags] (1)
```

(1) Specify the path to the folders, which can include sub-folders that contain YAML resources and network policies for analysis. The command scans the entire sub-folder trees for both the directories. For example, **<folder_path_1>** is **netpol-analysis-example-minimal/** and **<folder_path_2>** is **netpol-diff-example-minimal/**. Optionally, you can also specify parameters to modify the behavior of the command.

For more information about optional parameters, see roxctl netpol connectivity diff command options.

NOTE

The command considers all YAML files that you can accept using **kubectl apply -f**, and then these become valid inputs for your **roxctl netpol connectivity diff** command.

Example 9.2. Example output

diff-type	source	destination	dir 1	dir 2	workloads-diff-info
changed	default/frontend[Deployment]	default/backend[Deployment]	TCP 9090	TCP 9090,UDP 53	
added	0.0.0.0-255.255.255.255	default/backend[Deployment]	No Connections	TCP 9090	

The semantic difference report gives you an overview of the connections that were changed, added, or removed in **dir2** compared to the connections allowed in **dir1**. When you review the output, each line represents one allowed connection that was added, removed, or changed in **dir2** compared to **dir1**.

The following are example outputs generated by the **roxctl netpol connectivity diff** command in various formats:

- Example 1: text format
- Example 2: md format
- Example 3: csv format

If applicable, the **workloads-diff-info** provides additional details about added or removed workloads related to the added or removed connection.

For example, if a connection from workload **A** to workload **B** is removed because workload **B** was deleted, the **workloads-diff-info** indicates that workload **B** was removed. However, if such a connection was removed only because of network policy changes and neither workload **A** nor **B** was deleted, the **workloads-diff-info** is empty.

9.3.2. roxctl netpol connectivity diff command options

The **roxctl netpol connectivity diff** command supports the following options:

Option	Description
--dir1 string	First directory path of the input resources. This is a mandatory option.
--dir2 string	Second directory path of the input resources to be compared with the first directory path. This is a mandatory option.
--fail	Fail on the first encountered error. The default value is **false**.
-h, **--help**	View the help text for the **roxctl netpol connectivity diff** command.
-f, **--output-file string**	Save the connections difference output into a specific file.
-o, **--output-format string**	Configure the output format. The supported formats are **txt**, **md**, and **csv**. The default value is **txt**.
--remove	Remove the output path if it already exists. The default value is **false**.
--save-to-file	Save the connections difference output into default a file. The default value is **false**.
--strict	Treat warnings as errors. The default value is **false**.

9.3.3. Distinguishing between syntactic and semantic difference outputs

In the following example, **dir1** is netpol-analysis-example-minimal/, and **dir2** is netpol-diff-example-minimal/. The difference between the directories is a small change in the network policy **backend-netpol**.

Example policy from dir1:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  creationTimestamp: null
  name: backend-netpol
spec:
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: frontend
    ports:
    - port: 9090
      protocol: TCP
  podSelector:
    matchLabels:
      app: backendservice
  policyTypes:
  - Ingress
  - Egress
status: {}
```

The change in **dir2** is an added **-** before the ports attribute, which produces a difference output.

9.3.3.1. Syntactic difference output

Procedure

- Run the following command to compare the contents of the **netpols.yaml** files in the two specified directories:

  ```
  $ diff netpol-diff-example-minimal/netpols.yaml netpol-analysis-example-minimal/netpols.yaml
  ```

  ### Example output

  ```
  12c12
  <   - ports:
  ---
  >     ports:
  ```

9.3.3.2. Semantic difference output

Procedure

- Run the following command to analyze the connectivity differences between the Kubernetes manifests and network policies in the two specified directories:

```
$ roxctl netpol connectivity diff --dir1=roxctl/netpol/connectivity/diff/testdata/netpol-analysis-
example-minimal/ --dir2=roxctl/netpol/connectivity/diff/testdata/netpol-diff-example-minimal
```

Example output

```
Connectivity diff:
diff-type: changed, source: default/frontend[Deployment], destination:
default/backend[Deployment], dir1:  TCP 9090, dir2: TCP 9090,UDP 53
diff-type: added, source: 0.0.0.0-255.255.255.255, destination: default/backend[Deployment],
dir1:  No Connections, dir2: TCP 9090
```

CHAPTER 10. AUDITING LISTENING ENDPOINTS

Red Hat Advanced Cluster Security for Kubernetes (RHACS) provides the ability to audit the processes that are listening on ports in your secured clusters and filter this data by deployment, namespace, or cluster.

You can view information about processes and ports that they are listening on by using the following methods:

- In the RHACS web portal, go to **Network → Listening Endpoints**.
- Connect to the **ListeningEndpointsService** object in the API. For more information on the API, go to **Help → API reference** in the RHACS web portal.

The page provides a list of processes by deployment, with the following information displayed for each process on the list:

- Deployment name
- Cluster
- Namespace
- Count, or the number of processes listening on the ports in the deployment

You can further filter the information displayed on the page by using the filter field and entering individual deployments, namespaces, and clusters.

Click the expand icon at the top of the list to expand all sections for all deployments listed, or click the expand icon on a single deployment line to view additional information about that deployment. The following information is provided:

- Exec file path: Location of the process
- PID: System ID of the process
- Port: Port on which the process is listening
- Protocol: Protocol in use by the process
- Pod ID: Name of the pod where the process is contained

Clicking on a deployment name brings you to the **Risk** page in the RHACS web portal, where you can view information about the deployment, including risk indicators such as policy violations and additional deployment details.

CHAPTER 11. REVIEWING CLUSTER CONFIGURATION

Learn how to use the **Configuration Management** view and understand the correlation between various entities in your cluster to manage your cluster configuration efficiently.

Every OpenShift Container Platform cluster includes many different entities distributed throughout the cluster, which makes it more challenging to understand and act on the available information.

Red Hat Advanced Cluster Security for Kubernetes (RHACS) provides efficient configuration management that combines all these distributed entities on a single page. It brings together information about all your clusters, namespaces, nodes, deployments, images, secrets, users, groups, service accounts, and roles in a single **Configuration Management** view, helping you visualize different entities and the connections between them.

11.1. USING THE CONFIGURATION MANAGEMENT VIEW

To open the **Configuration Management** view, select **Configuration Management** from the navigation menu. Similar to the **Dashboard**, it displays some useful widgets.

These widgets are interactive and show the following information:

- Security policy violations by severity
- The state of CIS (Center for Information Security) Docker and Kubernetes benchmark controls
- Users with administrator rights in the most clusters
- Secrets used most widely in your clusters

The header in the **Configuration Management** view shows you the number of policies and CIS controls in your cluster.

NOTE

Only policies in the Deploy life cycle phase are included in the policy count and policy list view.

The header includes drop-down menus that allow you to switch between entities. For example, you can:

- Click **Policies** to view all policies and their severity, or select **CIS Controls** to view detailed information about all controls.
- Click **Application and Infrastructure** and select clusters, namespaces, nodes, deployments, images, and secrets to view detailed information.
- Click **RBAC Visibility and Configuration** and select users and groups, service accounts, and roles to view detailed information.

11.2. IDENTIFYING MISCONFIGURATIONS IN KUBERNETES ROLES

You can use the **Configuration Management** view to identify potential misconfigurations, such as users, groups, or service accounts granted the **cluster-admin** role, or roles that are not granted to anyone.

11.2.1. Finding Kubernetes roles and their assignment

Use the **Configuration Management** view to get information about the Kubernetes roles that are assigned to specific users and groups.

Procedure

1. Go to the RHACS portal and click **Configuration Management**.
2. Select **Role-Based Access Control → Users and Groups** from the header in the **Configuration Management** view. The **Users and Groups** view displays a list of Kubernetes users and groups, their assigned roles, and whether the **cluster-admin** role is enabled for each of them.
3. Select a user or group to view more details about the associated cluster and namespace permissions.

11.2.2. Finding service accounts and their permissions

Use the **Configuration Management** view to find out where service accounts are in use and their permissions.

Procedure

1. In the RHACS portal, go to **Configuration Management**.
2. Select **RBAC Visibility and Configuration → Service Accounts** from the header in the **Configuration Management** view. The **Service Accounts** view displays a list of Kubernetes service accounts across your clusters, their assigned roles, whether the **cluster-admin** role is enabled, and which deployments use them.
3. Select a row or an underlined link to view more details, including which cluster and namespace permissions are granted to the selected service account.

11.2.3. Finding unused Kubernetes roles

Use the **Configuration Management** view to get more information about your Kubernetes roles and find unused roles.

Procedure

1. In the RHACS portal, go to **Configuration Management**.
2. Select **RBAC Visibility and Configuration → Roles** from the header in the **Configuration Management** view. The **Roles** view displays a list of Kubernetes roles across your clusters, the permissions they grant, and where they are used.
3. Select a row or an underlined link to view more details about the role.
4. To find roles not granted to any users, groups, or service accounts, select the **Users & Groups** column header. Then select the **Service Account** column header while holding the **Shift** key. The list shows the roles that are not granted to any users, groups, or service accounts.

11.3. VIEWING KUBERNETES SECRETS

View Kubernetes secrets in use in your environment and identify deployments using those secrets.

Procedure

1. In the RHACS portal, go to **Configuration Management**.
2. On the **Secrets Most Used Across Deployments** widget, select **View All**. The **Secrets** view displays a list of Kubernetes secrets.
3. Select a row to view more details.

Use the available information to identify if the secrets are in use in deployments where they are not needed.

11.4. FINDING POLICY VIOLATIONS

The **Policy Violations by Severity** widget in the **Configuration Management** view displays policy violations in a sunburst chart. Each level of the chart is represented by one ring or circle.

- The innermost circle represents the total number of violations.
- The next ring represents the **Low**, **Medium**, **High**, and **Critical** policy categories.
- The outermost ring represents individual policies in a particular category.

The **Configuration Management** view only shows the information about policies that have the **Lifecycle Stage** set to **Deploy**. It does not include policies that address runtime behavior or those configured for assessment in the **Build** stage.

Procedure

1. In the RHACS portal, go to **Configuration Management**.
2. On the **Policy Violations by Severity** widget, move your mouse over the sunburst chart to view details about policy violations.
3. Select ***n* rated as high**, where *n* is a number, to view detailed information about high-priority policy violations. The **Policies** view displays a list of policy violations filtered on the selected category.
4. Select a row to view more details, including policy description, remediation, deployments with violations, and more. The details are visible in a panel.
5. The **Policy Findings** section in the information panel lists deployments where these violations occurred.
6. Select a deployment under the **Policy Findings** section to view related details including Kubernetes labels, annotations, and service account.

You can use the detailed information to plan a remediation for violations.

11.5. FINDING FAILING CIS CONTROLS

Similar to the **Policy Violations** sunburst chart in the **Configuration Management** view, the **CIS controls** widget provides information about failing Center for Information Security (CIS) controls.

Each level of the chart is represented by one ring or circle.

- The innermost circle represents the percentage of failing controls.
- The next ring represents the control categories.
- The outermost ring represents individual controls in a particular category.

Procedure

1. Select **CIS Docker v1.2.0** from the header of the **CIS controls** widget. Use this to switch between CIS Docker and Kubernetes controls.
2. Hover over the sunburst chart to view details about failing controls.
3. Select ***n* controls failing**, where *n* is a number, to view detailed information about failing controls. The **Controls** view displays a list of failing controls filtered based on the compliance state.
4. Select a row to view more details, including control descriptions and nodes where the controls are failing.
5. The **Control Findings** section in the information panel lists nodes where the controls are failing. Select a row to view more details, including Kubernetes labels, annotations, and other metadata.

You can use the detailed information to focus on a subset of nodes, industry standards, or failing controls. You can also assess, check, and report on the compliance status of your containerized infrastructure.

CHAPTER 12. EXAMINING IMAGES FOR VULNERABILITIES

With Red Hat Advanced Cluster Security for Kubernetes, you can analyze images for vulnerabilities using the RHACS scanners, or you can configure an integration to use another supported scanner.

The scanners in RHACS analyze each image layer to find packages and match them against known vulnerabilities by comparing them with a vulnerability database populated from different sources. Depending on the scanner used, sources include the National Vulnerability Database (NVD), the Open Source Vulnerabilities (OSV) database, and operating system vulnerability feeds.

NOTE

The RHACS Scanner V4 uses the OSV database available at OSV.dev under this license.

RHACS contains two scanners: the StackRox Scanner and Scanner V4.

IMPORTANT

Scanner V4 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.

The StackRox Scanner originates from a fork of the Clair v2 open source scanner and is the default scanner. In version 4.4, RHACS introduced Scanner V4, built on ClairCore, which provides additional image scanning features.

NOTE

This documentation uses the term "RHACS scanner" or "Scanner" to refer to the combined scanning capabilities provided by the two scanners: the StackRox Scanner and Scanner V4. When referring to the capabilities of a specific scanner, the name of the specific scanner is used.

When the RHACS scanner finds any vulnerabilities, it performs the following actions:

- Shows them in the **Vulnerability Management** view for detailed analysis
- Ranks vulnerabilities according to risk and highlights them in the RHACS portal for risk assessment
- Checks them against enabled security policies

The RHACS scanner inspects the images and identifies the installed components based on the files in the images. It might fail to identify installed components or vulnerabilities if the final images are modified to remove the following files:

Components	Files
Package managers	• **/etc/alpine-release** • **/etc/apt/sources.list** • **/etc/lsb-release** • **/etc/os-release** or **/usr/lib/os-release** • **/etc/oracle-release**, **/etc/centos-release**, **/etc/redhat-release**, or **/etc/system-release** • Other similar system files.
Language-level dependencies	• **package.json** for JavaScript. • **dist-info** or **egg-info** for Python. • **MANIFEST.MF** in Java Archive (JAR) for Java.
Application-level dependencies	• **dotnet/shared/Microsoft.AspNetCore.App/** • **dotnet/shared/Microsoft.NETCore.App/**

12.1. ABOUT RHACS SCANNER V4 (TECHNOLOGY PREVIEW)

RHACS provides its own scanner, or you can configure an integration to use RHACS with another vulnerability scanner.

Beginning with version 4.4, Scanner V4, built on ClairCore, provides scanning for language and operating system-specific image components. For version 4.4, RHACS also uses the StackRox Scanner to provide some scanning functionality until that functionality is implemented in a future release.

IMPORTANT

Scanner V4 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 .

Additional resources

- Scanner V4 settings for installing RHACS for OpenShift Container Platform by using the Operator
- Scanner V4 settings for installing RHACS for OpenShift Container Platform by using Helm
- Scanner V4 settings for installing RHACS for Kubernetes by using Helm

12.2. SCANNING IMAGES

For version 4.4, RHACS provides two scanners: the StackRox Scanner and Scanner V4. Both scanners can examine images in secured clusters connected in your network. Secured cluster scanning is enabled by default in Red Hat OpenShift environments deployed by using the Operator or when delegated scanning is used. See "Accessing delegated image scanning" for more information.

IMPORTANT

Scanner V4 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 .

When using the StackRox Scanner, RHACS performs the following actions:

- Central submits image scanning requests to the StackRox Scanner.
- Upon receiving these requests, the StackRox Scanner pulls the image layers from the relevant registry, checks the images, and identifies installed packages in each layer. Then it compares the identified packages and programming language-specific dependencies with the vulnerability lists and sends information back to Central
- The StackRox Scanner identifies the vulnerabilities in the following areas:
  - Base image operating system
  - Packages that are installed by the package managers
  - Programming language specific dependencies
  - Programming runtimes and frameworks

When using Scanner V4, RHACS performs the following actions:

- Central requests the Scanner V4 Indexer to download and index (analyze) given images.
- Scanner V4 Indexer pulls image metadata from registries to determine the layers of the image, and downloads each previously unindexed layer.
- Scanner V4 Indexer requests mapping files from Central that assist the indexing process. Scanner V4 Indexer produces in an index report.

- Central requests that Scanner V4 Matcher match given images to known vulnerabilities. This process results in the final scan result: a vulnerability report. Scanner V4 Matcher requests the latest vulnerabilities from Central.
- Scanner V4 Matcher requests the results of the image indexing, the index report, from Scanner V4 Indexer. It then uses the report to determine relevant vulnerabilities. This interaction occurs only when the image is indexed in the Central cluster. This interaction does not occur when Scanner V4 is matching vulnerabilities for images indexed in secured clusters.
- The Indexer stores data in the Scanner V4 DB that is related to the indexing results to ensure that image layers are only downloaded and indexed once. This prevents unnecessary network traffic and other resource utilization.
- When secured cluster scanning is enabled, Sensor requests Scanner V4 to index images. Scanner V4 Indexer requests mapping files from Sensor that assist the indexing process unless Central exists in the same namespace. In that case, Central is contacted instead.

12.2.1. Understanding and addressing common Scanner warning messages

When scanning images with Red Hat Advanced Cluster Security for Kubernetes (RHACS), you might see the **CVE DATA MAY BE INACCURATE** warning message. Scanner displays this message when it cannot retrieve complete information about the operating system or other packages in the image.

The following table shows some common Scanner warning messages:

Table 12.1. Warning messages

Message	Description
Unable to retrieve the OS CVE data, only Language CVE data is available	Indicates that Scanner does not officially support the base operating system of the image; therefore, it cannot retrieve CVE data for the operating system-level packages.
Stale OS CVE data	Indicates that the base operating system of the image has reached end-of-life, which means the vulnerability data is outdated. For example, Debian 8 and 9. For more information about the files needed to identify the components in the images, see Examining images for vulnerabilities.
Failed to get the base OS information	Indicates that Scanner scanned the image, but was unable to determine the base operating system used for the image.

Message	Description
Failed to retrieve metadata from the registry	Indicates that the target registry is unreachable on the network. The cause could be a firewall blocking **docker.io**, or an authentication issue preventing access. To analyze the root cause, create a special registry integration for private registries or repositories to get the pod logs for RHACS Central. For instructions on how to do this, see Integrating with image registries
Image out of scope for Red Hat Vulnerability Scanner Certification	Indicates that Scanner scanned the image, but the image is old and does not fall within the scope of Red Hat Scanner Certification. For more information, see Partner Guide for Red Hat Vulnerability Scanner Certification. IMPORTANT If you are using a Red Hat container image, consider using a base image newer than June 2020.

12.2.2. Supported package formats

Scanner can check for vulnerabilities in images that use the following package formats:

- apt
- apk
- dpkg
- rpm

12.2.3. Supported programming languages

Scanner can check for vulnerabilities in dependencies for the following programming languages:

- Go (Scanner V4 only)
  - Binaries: The standard library version used to build the binary is analyzed. If the binaries are built with module support (go.mod), then the dependencies are also analyzed.
- Java
  - JAR
  - WAR

- EAR

- JavaScript
  - Node.js
  - npm package.json
- Python
  - egg and wheel formats
- Ruby
  - gem

12.2.4. Supported runtimes and frameworks

Beginning from Red Hat Advanced Cluster Security for Kubernetes 3.0.50 (Scanner version 2.5.0), the StackRox Scanner identifies vulnerabilities in the following developer platforms:

- .NET Core
- ASP.NET Core

These are not supported by Scanner V4.

12.2.5. Supported operating systems

The supported platforms listed in this section are the distributions in which Scanner identifies vulnerabilities, and it is different from the supported platforms on which you can install Red Hat Advanced Cluster Security for Kubernetes.

Scanner identifies vulnerabilities in images that contain the following Linux distributions. For more information about the vulnerability databases used, see "Vulnerability sources" in "RHACS Architecture".

Distribution	Version
Alpine Linux	**alpine:3.2**[1],**alpine:3.3**, **alpine:3.4**, **alpine:3.5**, **alpine:3.6**, **alpine:3.7**, **alpine:3.8**, **alpine:3.9**, **alpine:3.10**, **alpine:3.11**, **alpine:3.12**, **alpine:3.13**, **alpine:3.14**, **alpine:3.15**, **alpine:3.16**, **alpine:3.17**, **alpine:3.18**, **alpine:3.19**[2], **alpine:edge**
Amazon Linux	**amzn:2018.03**, **amzn:2**, **amzn:2023**[2]
CentOS	**centos:6**[1], **centos:7**[1], **centos:8**[1]
Debian	**debian:10**, **debian:11**, **debian:12**, **debian:unstable**, **distroless**
Oracle Linux	versions 5-9[2]

Distribution	Version
Photon OS	1.0[2], 2.0[2], 3.0 [2]
Red Hat Enterprise Linux (RHEL)	**rhel:6**[3], **rhel:7**[3], **rhel:8**[3], **rhel:9**[3]
SUSE	SLES 11, 12, 15[2]; openSUSE Leap 42.3, 15.0, 15.1[2]; SUSE Linux[2]
Ubuntu	**ubuntu:14.04**, **ubuntu:16.04**, **ubuntu:18.04**, **ubuntu:20.04**, **ubuntu:21.04**, **ubuntu:21.10**, **ubuntu:22.04**, **ubuntu:22.10**, **ubuntu:23.04**, **ubuntu:23.10** The following vulnerability sources are not updated by the vendor: **ubuntu:12.04**, **ubuntu:12.10**, **ubuntu:13.04**, **ubuntu:14.10**, **ubuntu:15.04**, **ubuntu::15.10**, **ubuntu::16.10**, **ubuntu:17.04**, **ubuntu:17.10**, **ubuntu:18.10**, **ubuntu:19.04**, **ubuntu:19.10**, **ubuntu:20.10**,

1. Only supported in the StackRox Scanner.
2. Only supported in Scanner V4.
3. Images older than June 2020 are not supported in Scanner V4.

NOTE

- Scanner does not support the Fedora operating system because Fedora does not maintain a vulnerability database. However, Scanner still detects language-specific vulnerabilities in Fedora-based images.

Additional resources

- Vulnerability sources
- Integrating with image vulnerability scanners

12.3. ACCESSING DELEGATED IMAGE SCANNING

You can have isolated container image registries that are only accessible from your secured clusters. The delegated image scanning feature enables you to scan images from any registry in your secured clusters.

12.3.1. Enhancing image scanning by accessing delegated image scanning

Currently, by default, Central Services Scanner performs both indexing (identification of components) and vulnerability matching (enrichment of components with vulnerability data) for images observed in your secured clusters, with the exception of images from the OpenShift Container Platform integrated registry.

For images from the OpenShift Container Platform integrated registry, Scanner-slim installed in your secured cluster performs the indexing, and the Central Services Scanner performs the vulnerability matching.

The delegated image scanning feature extends scanning functionality by allowing Scanner-slim to index images from any registry and then send them to Central for vulnerability matching. To use this feature, ensure that Scanner-slim is installed in your secured clusters. If Scanner-slim is not present, scan requests are sent directly to Central.

12.3.2. Configuring delegated image scanning

A new delegated registry configuration specifies the registries from which image scans are to be delegated. For images observed by Sensor, this configuration allows you to delegate scans from no registries, all registries, or specific registries. To enable delegation of scans using the **roxctl** CLI, the Jenkins plugin, or the API, you must also specify a destination cluster and source registry.

Prerequisites

- Scanner-slim must be installed in the secured cluster to scan images.

  NOTE

  Enabling Scanner-slim is supported on OpenShift Container Platform and Kubernetes secured clusters.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Clusters**.
2. In the **Clusters** view header, click **Manage delegated scanning**.
3. In the **Delegated Image Scanning** page, provide the following information:
   - **Delegate scanning for**: Choose the scope of the image delegation by selecting one of the following options:
     - None: The default option. This option specifies that no images are scanned by the secured clusters, except for images from the OpenShift Container Platform integrated registry.
     - All registries: This option indicates that all images are scanned by secured clusters.
     - Specified registries: This option specifies which images should be scanned by secured clusters based on the registries list.
   - **Select default cluster to delegate to**: From the drop-down list, select the name of the default cluster that will process the scan requests coming from the command-line interface (CLI) and API. This is optional and you can select **None** if required.

- Optional: Click **Add registry** and specify the source registry and destination cluster details. You can select the destination cluster as **None** if the scan requests are not coming from the CLI and API. You can add more than one source registry and destination cluster if required.

4. Click **Save**.

Image integrations are now synchronized between Central and Sensor, and Sensor captures pull-secrets from each namespace. Sensor then uses these credentials to authenticate to the image registries.

12.3.3. Installing and configuring Scanner-slim on secured clusters

12.3.3.1. Using the Operator

RHACS Operator installs a Scanner-slim version on each secured cluster to scan images in the OpenShift Container Platform integrated registry and optionally other registries.

For more information, see Installing RHACS on secured clusters by using the Operator .

12.3.3.2. Using Helm

Secured Cluster Services Helm chart (**secured-cluster-services**) installs a Scanner-slim version on each secured cluster. In Kubernetes, the secured cluster services include Scanner-slim as an optional component. On OpenShift Container Platform, however, RHACS installs a Scanner-slim version on each secured cluster to scan images in the OpenShift Container Platform integrated registry and optionally other registries.

- For OpenShift Container Platform installations, see Installing the secured-cluster-services Helm chart without customization.
- For non-OpenShift Container Platform installations, such as Amazon Elastic Kubernetes Service (Amazon EKS), Google Kubernetes Engine (Google GKE), and Microsoft Azure Kubernetes Service (Microsoft AKS), see Installing the secured-cluster-services Helm chart without customization.

12.3.3.3. Verifying after installation

Procedure

- Verify that the status of the secured cluster indicates that Scanner is present and healthy:
  a. In the RHACS portal, go to **Platform Configuration → Clusters**.
  b. In the **Clusters** view, select a cluster to view its details.
  c. In the **Health Status** card, ensure that **Scanner** is present and is marked as **Healthy.**

12.3.3.4. Using image scanning

You can scan images stored in a cluster specific OpenShift Container Platform integrated image registry by using **roxctl** CLI, Jenkins, and API. You can specify the appropriate cluster in the delegated scanning configuration or use the cluster parameter available in **roxctl** CLI, Jenkins, and API.

For more information about how to scan images by using the **roxctl** CLI, see Image scanning by using the roxctl CLI.

12.4. SETTING UP SCANNING

You can configure settings for scanning, such as automatic scanning of active and inactive images.

12.4.1. Automatic scanning of active images

Red Hat Advanced Cluster Security for Kubernetes periodically scans all active images and updates the image scan results to reflect the latest vulnerability definitions. Active images are the images you have deployed in your environment.

NOTE

From Red Hat Advanced Cluster Security for Kubernetes 3.0.57, you can enable automatic scanning of inactive images by configuring the **Watch** setting for images.

Central fetches the image scan results for all active images from Scanner or other integrated image scanners that you use and updates the results every 4 hours.

You can also use the **roxctl** CLI to check the image scan results on demand.

12.4.2. Scanning inactive images

Red Hat Advanced Cluster Security for Kubernetes (RHACS) scans all active (deployed) images every 4 hours and updates the image scan results to reflect the latest vulnerability definitions.

You can also configure RHACS to scan inactive (not deployed) images automatically.

Procedure

1. In the RHACS portal, go to **Vulnerability Management (2.0) → Workload CVEs (Tech preview)**.
2. Click **<number> Images** to display a list of images and locate the image you want to watch.
3. Click the overflow menu, ⋮, and then select **Watch image**. RHACS then scans the image and shows an error or success message.
4. (Optional) To remove a watched image, click the overflow menu, ⋮, and then select **Unwatch image**.
5. (Optional) You can view the list of all watched images and add additional images to watch by clicking **Manage watched images** in the page header.

IMPORTANT

In the RHACS portal, click **Platform Configuration → System Configuration** to view the data retention configuration.

All the data related to the image removed from the watched image list continues to appear in the RHACS portal for the number of days mentioned on the **System Configuration** page and is only removed after that period is over.

6. Click **Close** to return to the **Workload CVEs** page.

Additional resources

- Scanning inactive images
- Installing the roxctl CLI

12.5. ABOUT VULNERABILITIES

RHACS fetches vulnerability definitions and updates from multiple vulnerability feeds. These feeds are both general in nature, such as NVD, or distribution-specific, such as Alpine, Debian, and Ubuntu. For more information on viewing and addressing vulnerabilities that are found, see Vulnerability management.

12.5.1. Fetching vulnerability definitions

In online mode, Central fetches the vulnerability definitions every 5 minutes from a single feed. This feed combines vulnerability definitions from upstream sources, and it refreshes every 3 hours.

- The address of the feed is **https://definitions.stackrox.io**.
- You can change the default query frequency for Central and the StackRox Scanner by setting the **ROX_SCANNER_VULN_UPDATE_INTERVAL** environment variable:

```
$ oc -n stackrox set env deploy/central ROX_SCANNER_VULN_UPDATE_INTERVAL=
<value> 1
```

1 If you use Kubernetes, enter **kubectl** instead of **oc**.

Note the following guidance:

- The StackRox Scanner’s configuration map still has an **updater.interval** parameter for configuring the scanner’s updating frequency, but it no longer includes the **fetchFromCentral** parameter.
- Setting this environment variable is not supported for Scanner V4.

For more information about the vulnerability sources that RHACS uses, see "Vulnerability sources" in "Red Hat Advanced Cluster Security for Kubernetes architecture".

Additional resources

- Vulnerability sources

12.5.2. Understanding vulnerability scores

The Red Hat Advanced Cluster Security for Kubernetes portal shows a single Common Vulnerability Scoring System (CVSS) base score for each vulnerability. RHACS shows the CVSS score based on the following criteria:

- If a CVSS v3 score is available, RHACS shows the score and lists **v3** along with it. For example, **6.5 (v3)**.

NOTE

CVSS v3 scores are only available if you are using the StackRox Scanner version 1.3.5 and later or Scanner V4.

- If a CVSS v3 score is not available, RHACS might show only the CVSS v2 score. For example, **6.5**.

You can use the API to get the CVSS scores. If CVSS v3 information is available for a vulnerability, the response might include both CVSS v3 and CVSS v2 information.

For a Red Hat Security Advisory (RHSA), the CVSS score is set to the highest CVSS score among all the related CVEs. One RHSA can contain multiple CVEs, and Red Hat sometimes assigns a different score based on how a vulnerability affects other Red Hat products.

12.6. DISABLING LANGUAGE-SPECIFIC VULNERABILITY SCANNING

Scanner identifies the vulnerabilities in the programming language-specific dependencies by default. You can disable the language-specific dependency scanning.

Procedure

- To disable language-specific vulnerability scanning, run the following command:

```
$ oc -n stackrox set env deploy/scanner \ (1)
  ROX_LANGUAGE_VULNS=false (2)
```

(1) If you use Kubernetes, enter **kubectl** instead of **oc**.

(2) If you are using Red Hat Advanced Cluster Security for Kubernetes version 3.0.47 or older, replace the environment variable name **ROX_LANGUAGE_VULNS** with **LANGUAGE_VULNS**.

12.7. ADDITIONAL RESOURCES

- Red Hat CVE Database

CHAPTER 13. VERIFYING IMAGE SIGNATURES

You can use Red Hat Advanced Cluster Security for Kubernetes (RHACS) to ensure the integrity of the container images in your clusters by verifying image signatures against pre-configured keys.

You can create policies to block unsigned images and images that do not have a verified signature. You can also enforce the policy by using the RHACS admission controller to stop unauthorized deployment creation.

NOTE

- RHACS 3.70 only supports Cosign signatures and Cosign public key signature verification. For more information about Cosign, see Cosign overview.
- You must configure signature integration with at least 1 Cosign public key for signature verification.
- For all deployed and watched images:
  - RHACS fetches and verifies the signatures every 4 hours.
  - RHACS verifies the signatures whenever you change or update your signature integration public keys.

13.1. CONFIGURING SIGNATURE INTEGRATION

Before performing image signature verification, you must first add your Cosign public keys in RHACS.

Prerequisites

- You must already have a PEM-encoded Cosign public key. For more information about Cosign, see Cosign overview.

Procedure

1. In the RHACS portal, select **Platform Configuration → Integrations**.
2. Scroll down to the **Signature Integrations** section and click **Signature**.
3. Click **New integration**.
4. Enter a name for the **Integration name**.
5. Click **Cosign → Add a new public key**.
6. Enter the **Public key** name.
7. For the **Public key value** field, enter the PEM-encoded public key.
8. (Optional) You can add more than one key by clicking **Add a new public key** and entering the details.
9. Click **Save**.

13.2. USING SIGNATURE VERIFICATION IN A POLICY

When creating custom security policies, you can use the **Trusted image signers** policy criteria to verify image signatures.

Prerequisites

- You must have already configured a signature integration with at least 1 Cosign public key.

Procedure

1. When creating or editing a policy, drag the **Not verified by trusted image signers** policy criteria in the policy field drop area for the **Policy criteria** section.
2. Click **Select**.
3. Select the trusted image signers from the list and click **Save**.

Additional resources

- Creating a security policy from the system policies view
- Policy criteria

13.3. ENFORCING SIGNATURE VERIFICATION

To prevent the users from using unsigned images, you can enforce signature verification by using the RHACS admission controller. You must first enable the **Contact Image Scanners** feature in your cluster configuration settings. Then, while creating a security policy to enforce signature verification, you can use the **Inform and enforce** option.

For more information, see Enabling admission controller enforcement.

Additional resources

- Creating a security policy from the system policies view

CHAPTER 14. MANAGING VULNERABILITIES

14.1. VULNERABILITY MANAGEMENT

Security vulnerabilities in your environment might be exploited by an attacker to perform unauthorized actions such as denial of service, remote code execution, or unauthorized access to sensitive data. Therefore, the management of vulnerabilities is a foundational step towards a successful Kubernetes security program.

14.1.1. Vulnerability management process

Vulnerability management is a continuous process to identify and remediate vulnerabilities. Red Hat Advanced Cluster Security for Kubernetes helps you to facilitate a vulnerability management process.

A successful vulnerability management program often includes the following critical tasks:

- Performing asset assessment
- Prioritizing the vulnerabilities
- Assessing the exposure
- Taking action
- Continuously reassessing assets

Red Hat Advanced Cluster Security for Kubernetes helps organizations to perform continuous assessments on their OpenShift Container Platform and Kubernetes clusters. It provides organizations with the contextual information they need to prioritize and act on vulnerabilities in their environment more effectively.

14.1.1.1. Performing asset assessment

Performing an assessment of an organization’s assets involve the following actions:

- Identifying the assets in your environment
- Scanning these assets to identify known vulnerabilities
- Reporting on the vulnerabilities in your environment to impacted stakeholders

When you install Red Hat Advanced Cluster Security for Kubernetes on your Kubernetes or OpenShift Container Platform cluster, it first aggregates the assets running inside of your cluster to help you identify those assets. RHACS allows organizations to perform continuous assessments on their OpenShift Container Platform and Kubernetes clusters. RHACS provides organizations with the contextual information to prioritize and act on vulnerabilities in their environment more effectively.

Important assets that should be monitored by the organization’s vulnerability management process using RHACS include:

- **Components**: Components are software packages that may be used as part of an image or run on a node. Components are the lowest level where vulnerabilities are present. Therefore, organizations must upgrade, modify or remove software components in some way to remediate vulnerabilities.

- **Image**: A collection of software components and code that create an environment to run an executable portion of code. Images are where you upgrade components to fix vulnerabilities.
- **Nodes**: A server used to manage and run applications using OpenShift or Kubernetes and the components that make up the OpenShift Container Platform or Kubernetes service.

Red Hat Advanced Cluster Security for Kubernetes groups these assets into the following structures:

- **Deployment**: A definition of an application in Kubernetes that may run pods with containers based on one or many 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 OpenShift or Kubernetes.

Red Hat Advanced Cluster Security for Kubernetes scans the assets for known vulnerabilities and uses the Common Vulnerabilities and Exposures (CVE) data to assess the impact of a known vulnerability.

14.1.2. Viewing vulnerabilities

RHACS provides the following methods to view vulnerabilities discovered in your system:

- To view application vulnerabilities by namespace or deployment, or to view vulnerabilities in an image, in the RHACS web portal, go to **Vulnerability Management (1.0)** → **Dashboard**.
- To view vulnerabilities in applications running on clusters in your system, go to **Vulnerability Management (2.0)** → **Workload CVEs**. You can filter vulnerabilities by image, deployment, namespace, and cluster.

14.1.2.1. Viewing application vulnerabilities

You can view application vulnerabilities in Red Hat Advanced Cluster Security for Kubernetes.

Procedure

1. In the RHACS portal, go to **Vulnerability Management 1.0** → **Dashboard**.
2. On the **Dashboard** view header, select **Application & Infrastructure** → **Namespaces** or **Deployments**.
3. From the list, search for and select the **Namespace** or **Deployment** you want to review.
4. To get more information about the application, select an entity from **Related entities** on the right.

14.1.2.2. Viewing image vulnerabilities

You can view image vulnerabilities in Red Hat Advanced Cluster Security for Kubernetes.

Procedure

1. In the RHACS portal, go to **Vulnerability Management 1.0** → **Dashboard**.
2. On the **Dashboard** view header, select **Images**.

3. From the list of images, select the image you want to investigate. You can also filter the list by performing one of the following steps:
   a. Enter **Image** in the search bar and then select the **Image** attribute.
   b. Enter the image name in the search bar.
4. In the image details view, review the listed CVEs and prioritize taking action to address the impacted components.
5. Select **Components** from **Related entities** on the right to get more information about all the components that are impacted by the selected image. Or select **Components** from the **Affected components** column under the **Image findings** section for a list of components affected by specific CVEs.

Additional resources

- Using local page filtering

14.1.2.3. Viewing workload CVEs in Vulnerability Management (2.0)

You can view a comprehensive list of vulnerabilities, or CVEs, in RHACS across images and deployments. You can use the search filter bar to select specific CVEs, images, deployments, namespaces, or clusters.

Procedure

1. In the RHACS portal, go to **Vulnerability Management (2.0)** → **Workload CVEs**.
2. From the drop-down list, select the search criteria you want to use. You can select an item type, such as a cluster, from the list, and then select the specific name of the item. You can add additional items to the filter by selecting another item from the list and selecting the specific name of the new item. For example, you can select a specific image and a specific cluster to limit results to those selections. You can filter on the following items:
   - CVE
   - Image
   - Deployment
   - Namespace
   - Cluster
   - Component
   - Component source
3. Optional: Use the **CVE severity** list to select the severities of the CVEs that you want to display.
4. Click the relevant button to view a list of vulnerabilities, images, or deployments in the system.

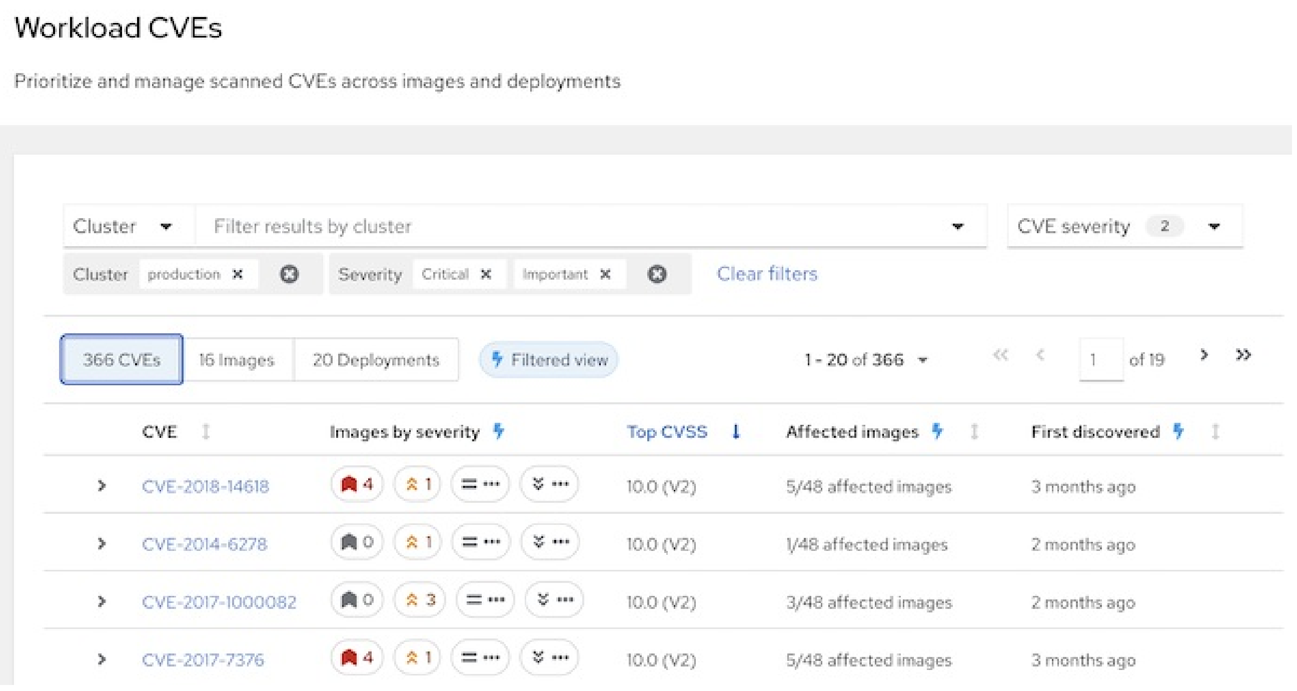
NOTE

The **Filtered view** icon indicates that the displayed results were filtered based on the criteria that you selected. You can click **Clear filters** to remove all filters, or remove individual filters by clicking on them.

5. In the list of results, click a CVE, image name, or deployment name to view more information about the item. For example, depending on the item type, you can view the following information:
   - Whether a CVE is fixable
   - Whether an image is active
   - The Dockerfile line in the image that contains the CVE
   - External links to information about the CVE in Red Hat and other CVE databases

Search example

The following graphic shows an example of search criteria for a cluster called "production" to view CVEs of critical and important severity in that cluster.



Workload CVEs

Prioritize and manage scanned CVEs across images and deployments

Cluster ▾ Filter results by cluster ▾ CVE severity 2 ▾

Cluster production ✕ ⊗ Severity Critical ✕ Important ✕ ⊗ Clear filters

366 CVEs | 16 Images | 20 Deployments | Filtered view | 1 - 20 of 366 ▾ | 1 of 19

	CVE	Images by severity	Top CVSS	Affected images	First discovered
›	CVE-2018-14618	4 · 1 · ... · ...	10.0 (V2)	5/48 affected images	3 months ago
›	CVE-2014-6278	0 · 1 · ... · ...	10.0 (V2)	1/48 affected images	2 months ago
›	CVE-2017-1000082	0 · 3 · ... · ...	10.0 (V2)	3/48 affected images	2 months ago
›	CVE-2017-7376	4 · 1 · ... · ...	10.0 (V2)	5/48 affected images	3 months ago

14.1.2.3.1. Viewing infrastructure vulnerabilities

You can view vulnerabilities in nodes by using Red Hat Advanced Cluster Security for Kubernetes.

Procedure

1. In the RHACS portal, go to **Vulnerability Management 1.0 → Dashboard**.
2. On the **Dashboard** view header, select **Application & Infrastructure → Cluster**.
3. From the list of clusters, select the cluster you want to investigate.

4. Review the clusters vulnerabilities and prioritize taking action on the impacted nodes on the cluster.

14.1.2.3.2. Viewing node vulnerabilities

You can view vulnerabilities in specific nodes by using Red Hat Advanced Cluster Security for Kubernetes.

Procedure

1. In the RHACS portal, go to **Vulnerability Management 1.0 → Dashboard**.
2. On the **Dashboard** view header, select **Nodes**.
3. From the list of nodes, select the node you want to investigate.
4. Review vulnerabilities for the selected node and prioritize taking action.
5. To get more information about the affected components in a node, select **Components** from **Related entities** on the right.

14.1.2.4. Prioritizing the vulnerabilities

Answer the following questions to prioritize the vulnerabilities in your environment for action and investigation:

- How important is an affected asset for your organization?
- How severe does a vulnerability need to be for investigation?
- Can the vulnerability be fixed by a patch for the affected software component?
- Does the existence of the vulnerability violate any of your organization’s security policies?

The answers to these questions help security and development teams decide if they want to gauge the exposure of a vulnerability.

Red Hat Advanced Cluster Security for Kubernetes provides you the means to facilitate the prioritization of the vulnerabilities in your applications and components.

14.1.2.5. Assessing the exposure

To assess your exposure to a vulnerability, answer the following questions:

- Is your application impacted by a vulnerability?
- Is the vulnerability mitigated by some other factor?
- Are there any known threats that could lead to the exploitation of this vulnerability?
- Are you using the software package which has the vulnerability?
- Is spending time on a specific vulnerability and the software package worth it?

Take some of the following actions based on your assessment:

- Consider marking the vulnerability as a false positive if you determine that there is no exposure or that the vulnerability does not apply in your environment.
- Consider if you would prefer to remediate, mitigate or accept the risk if you are exposed.
- Consider if you want to remove or change the software package to reduce your attack surface.

14.1.2.6. Taking action

Once you have decided to take action on a vulnerability, you can take one of the following actions:

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

You can remediate vulnerabilities by performing one of the following actions:

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

Additional resources

- Reviewing a false positive or deferred CVE

14.1.2.6.1. Finding a new component version

The following procedure finds a new component version to upgrade to.

Procedure

1. In the RHACS portal, go to **Vulnerability Management 1.0 → Dashboard**.
2. On the **Dashboard** view header, select **Images**.
3. From the list of images, select the image you already assessed.
4. Under the **Image findings** section, select the CVE.
5. Select the affected components of the CVE you want to take action on.
6. Review the version of the component that the CVE is fixed in and update your image.

14.1.2.7. Accepting risks

Follow the instructions in this section to accept the risks in Red Hat Advanced Cluster Security for Kubernetes.

Prerequisites

- You must have **write** permission for the **VulnerabilityManagementRequests** resource.

To accept risk with or without mitigation:

Procedure

1. In the RHACS portal, go to **Vulnerability Management 1.0 → Dashboard**.
2. On the **Dashboard** view header, select **Images**.
3. From the list of images, select the image you already assessed.
4. Find the row which lists the CVE you would like to take action on.
5. Click the overflow menu, ⋮, for the CVE you identified.
6. Click **Defer CVE**.
7. Select the date and time till you want to defer the CVE.
8. Select if you want to defer the CVE for the selected image tag or all tags for this image.
9. Enter the reason for the deferral.
10. Click **Request approval**. Select the blue information icon on the right of the CVE and copy the approval link to share with your organization’s deferral approver.

14.1.2.7.1. Marking vulnerabilities as false positive

The following procedure marks a vulnerability as a false positive.

Prerequisites

- You must have the **write** permission for the **VulnerabilityManagementRequests** resource.

Procedure

1. In the RHACS portal, go to **Vulnerability Management 1.0 → Dashboard**.
2. On the **Dashboard** view header, select **Images**.
3. From the list of images, select the image you already assessed.
4. Find the row which lists the CVE you would like to take action on.
5. Click the ⋮ on the right for the CVE you identified and click **Defer CVE**.
6. Select the date and time you want to defer the CVE.
7. Select if you want to defer the CVE for the selected image tag or all tags for this image.
8. Enter the reason for the deferral.
9. Click **Request approval**.

10. Select the blue information icon on the right of the CVE and copy the approval link to share with your organization's deferral approver.

14.1.2.7.2. Reviewing a false positive or deferred CVE

Use the following procedure to review a false positive or deferred CVE.

Prerequisites

- You must have the **write** permission for the **VulnerabilityManagementApprovals** resource.

You can review a false positive or deferred CVE:

Procedure

1. Open the approval link in your browser or in the RHACS portal.
2. Go to **Vulnerability Management → Risk Acceptance** and search for the CVE.
3. Review the vulnerabilities scope and action to decide if you would like to approve it.
4. Click on the ⋮ at the far right of the CVE and approve or deny the request for approval.

14.1.2.8. Reporting vulnerabilities to teams

As organizations must constantly reassess and report on their vulnerabilities, some organizations find it helpful to have scheduled communications to key stakeholders to help in the vulnerability management process.

You can use Red Hat Advanced Cluster Security for Kubernetes to schedule these reoccurring communications through e-mail. These communications should be scoped to the most relevant information that the key stakeholders need.

For sending these communications, you must consider the following questions:

- What schedule would have the most impact when communicating with the stakeholders?
- Who is the audience?
- Should you only send specific severity vulnerabilities in your report?
- Should you only send fixable vulnerabilities in your report?

14.1.3. Vulnerability reporting

You can create and download an on-demand image vulnerability report from the **Vulnerability Management (2.0)** menu in the RHACS web portal. This report includes a comprehensive list of common vulnerabilities and exposures across images and deployments, called workload CVEs in RHACS. You can share this report with auditors or internal stakeholders by scheduling emails in RHACS or by downloading the report and sharing it by using other methods.

14.1.3.1. Creating vulnerability management report configurations

RHACS guides you through the process of creating a vulnerability management report configuration. This configuration determines the information that will be included in a report job that runs at a scheduled time or that you run on demand.

Procedure

1. In the RHACS portal, go to **Vulnerability Management (2.0)→ Vulnerability Reporting**.
2. Click **Create report**.
3. Enter a name for your report configuration in the **Report name** field.
4. Optional: Enter text describing the report configuration in the **Description** field.
5. In the **CVE severity** field, select the severity of common vulnerabilities and exposures (CVEs) that you want to include in the report configuration.
6. Select the **CVE status**. You can select **Fixable**, **Unfixable**, or both.
7. In the **Image type** field, select whether you want to include CVEs from deployed images, watched images, or both.
8. In the **CVEs discovered since** field, select the time period for which you want CVEs to be included in the report configuration.
9. In the **Configure report scope** field, you can perform the following actions:
    - Select an existing collection and click **View** to view the collection information, edit the collection, and get a preview of collection results. When viewing the collection, entering text in the field searches for collections matching that text string.
    - Click **Create collection** to create a new collection.

    > **NOTE**
    >
    > For more information about collections, see "Creating and using deployment collections" in the "Additional resources" section.

10. Click **Next** to configure the delivery destinations and optionally set up a schedule for delivery.

14.1.3.1.1. Configuring delivery destinations and scheduling

Configuring destinations and delivery schedules for vulnerability reports is optional, unless on the previous page, you selected the option to include CVEs that were discovered since the last scheduled report. If you selected that option, configuring destinations and delivery schedules for vulnerability reports is required.

Procedure

1. To configure destinations for delivery, in the **Configure delivery destinations** section, you can add a delivery destination and set up a schedule for reporting.
2. To email reports, you must configure at least one email notifier. Select an existing notifier or create a new email notifier to send your report by email. For more information about creating an email notifier, see "Configuring the email plugin" in the "Additional resources" section.

    [illegible]

When you select a notifier, the email addresses configured in the notifier as **Default recipients** appear in the **Distribution list** field. You can add additional email addresses that are separated by a comma.

3. A default email template is automatically applied. To edit this default template, perform the following steps:

   a. Click the edit icon and enter a customized subject and email body in the **Edit** tab.

   b. Click the **Preview** tab to see your proposed template.

   c. Click **Apply** to save your changes to the template.

   NOTE

   When reviewing the report jobs for a specific report, you can see whether the default template or a customized template was used when creating the report.

4. In the **Configure schedule** section, select the frequency and day of the week for the report.

5. Click **Next** to review your vulnerability report configuration and finish creating it.

14.1.3.1.2. Reviewing and creating the report configuration

You can review the details of your vulnerability report configuration before creating it.

Procedure

1. In the **Review and create** section, you can review the report configuration parameters, delivery destination, email template that is used if you selected email delivery, delivery schedule, and report format. To make any changes, click **Back** to go to the previous section and edit the fields that you want to change.

2. Click **Create** to create the report configuration and save it.

14.1.3.2. Vulnerability report permissions

The ability to create, view, and download reports depends on the access control settings, or roles and permission sets, for your user account.

For example, you can only view, create, and download reports for data that your user account has permission to access. In addition, the following restrictions apply:

- You can only download reports that you have generated; you cannot download reports generated by other users.

- Report permissions are restricted depending on the access settings for user accounts. If the access settings for your account change, old reports do not reflect the change. For example, if you are given new permissions and want to view vulnerability data that is now allowed by those permissions, you must create a new vulnerability report.

14.1.3.3. Editing vulnerability report configurations

You can edit existing vulnerability report configurations from the list of report configurations, or by selecting an individual report configuration first.

Procedure

1. To edit an existing vulnerability report configuration, in the RHACS web portal, go to **Vulnerability Management (2.0)→ Vulnerability Reporting** and choose one of the following methods:
   - Locate the report configuration that you want to edit in the list of report configurations.

     Click the overflow menu, ⋮ , and then select **Edit report**.
   - Click the report configuration name in the list of report configurations. Then, click **Actions** and select **Edit report**.
2. Make changes to the report configuration and save.

14.1.3.4. Downloading vulnerability reports

You can generate an on-demand vulnerability report and then download it.

> **NOTE**
>
> You can only download reports that you have generated; you cannot download reports generated by other users.

Procedure

1. In the RHACS web portal, go to **Vulnerability Management (2.0)→ Vulnerability Reporting** and, in the list of report configurations, locate the report configuration that you want to use to create the downloadable report.
2. Generate the vulnerability report by using one of the following methods:
   - To generate the report from the list:
     a. Click the overflow menu, ⋮ , and then select **Generate download**. The **My active job status** column displays the status of your report creation. After the **Processing** status goes away, you can download the report.
   - To generate the report from the report window:
     a. Click the report configuration name to open the configuration detail window.
     b. Click **Actions** and select **Generate download**.
3. To download the report, if you are viewing the list of report configurations, click the report configuration name to open it.
4. Click **All report jobs**.
5. If the report is completed, click the **Ready for download** link in the **Status** column. The report is in **.csv** format and is compressed into a **.zip** file for download.

14.1.3.5. Sending vulnerability reports on-demand

You can send vulnerability reports immediately, rather than waiting for the scheduled send time.

Procedure

1. In the RHACS web portal, go to **Vulnerability Management (2.0)→ Vulnerability Reporting** and, in the list of report configurations, locate the report configuration for the report that you want to send.
2. Click the overflow menu, ⋮, and then select **Send report now**.

14.1.3.6. Cloning vulnerability report configurations

You can make copies of vulnerability report configurations by cloning them. This is useful when you want to reuse report configurations with minor changes, such as reporting vulnerabilities in different deployments or namespaces.

Procedure

1. In the RHACS web portal, go to **Vulnerability Management (2.0)→ Vulnerability Reporting** and locate the report configuration that you want to clone in the list of report configurations.
2. Click **Clone report**.
3. Make any changes that you want to the report parameters and delivery destinations.
4. Click **Create**.

14.1.3.7. Deleting vulnerability report configurations

Deleting a report configuration deletes the configuration and any reports that were previously run using this configuration.

Procedure

1. In the RHACS web portal, go to **Vulnerability Management (2.0)→ Vulnerability Reporting** and locate the report configuration that you want to delete in the list of reports.
2. Click the overflow menu, ⋮, and then select **Delete report**.

14.1.3.8. Configuring vulnerability management report job retention settings

You can configure settings that determine when vulnerability report job requests expire and other retention settings for report jobs.

NOTE

These settings do not affect the following vulnerability report jobs:

- Jobs in the **WAITING** or **PREPARING** state (unfinished jobs)
- The last successful scheduled report job
- The last successful on-demand emailed report job
- The last successful downloadable report job
- Downloadable report jobs for which the report file has not been deleted by either manual deletion or by configuring the downloadable report pruning settings

Procedure

1. In the RHACS web portal, go to **Platform Configuration** → **System Configuration**. You can configure the following settings for vulnerability report jobs:
   - **Vulnerability report run history retention**: The number of days that a record is kept of vulnerability report jobs that have been run. This setting controls how many days that report jobs are listed in the **All report jobs** tab under **Vulnerability Management (2.0)** → **Vulnerability Reporting** when a report configuration is selected. All report history beyond the cutoff date is pruned except for the following jobs:
     - Unfinished jobs.
     - Jobs for which prepared downloadable reports still exist in the system.
     - The last successful report job for each job type (scheduled email, on-demand email, or download). This ensures users have information about the last run job for each type.
   - **Prepared downloadable vulnerability reports retention days**: The number of days that created on-demand downloadable vulnerability report jobs are available for download in the **All report jobs** tab under **Vulnerability Management (2.0)** → **Vulnerability Reporting** when a report configuration is selected.
   - **Prepared downloadable vulnerability reports limit**: The limit, in MB, of space allocated to prepared downloadable vulnerability report jobs. After the limit is reached, the oldest report job in the download queue is removed.
2. To change these values, click **Edit**, make your changes, and then click **Save**.

14.1.3.9. Migration of vulnerability reports when upgrading to RHACS version 4.3 and later

Red Hat Advanced Cluster Security for Kubernetes (RHACS) version 4.3 includes an automatic migration of vulnerability report configurations that were created in previous versions of RHACS in the **Vulnerability Management 1.0** → **Reporting** page. You can access migrated report configurations by clicking **Vulnerability Management (2.0)** → **Vulnerability Reporting**. The previous versions of the report configurations are no longer available in the RHACS web portal or by using the API.

RHACS performs the following actions during the migration:

- Report configurations are copied to create a new version of the report that you can access by clicking **Vulnerability Management (2.0)** → **Vulnerability Reporting**.

- The original name for the report is used when migrating reports to the new location.
- Report configurations created in the **Vulnerability Management 2.0 (Tech preview)→ Reporting** page are not affected by upgrading to RHACS version 4.3 or later. The menu item to access these report configurations was renamed **Vulnerability Management (2.0)** and the page was renamed **Vulnerability Reporting**.
- If a report configuration previously created by using the **Vulnerability Management 1.0** page is not migrated because the notifier attached to it no longer exists, then the details of that configuration are added to the logs generated by the Central pod. You can use details from the log to re-create the report configuration by clicking **Vulnerability Management (2.0)→ Vulnerability Reporting** and adding a new report.
- For each report configuration that was previously created by using the **Vulnerability Management 1.0** page, the most recent successful scheduled report job is migrated to the **All Report jobs** section of the report configuration. To view the report configuration, click **Vulnerability Management (2.0)→ Vulnerability Reporting**, and then click the report configuration.

If you need to roll back to RHACS 4.2 from a later version, the following actions occur:

- The report configurations that became defunct with migration now become functional again and are available by clicking **Vulnerability Management 1.0 → Reporting**.
- The report configurations created by the migration remain functional and are available by clicking **Vulnerability Reporting 2.0 (Tech Preview)**. You can manually delete unwanted report configurations created in either the 1.0 or 2.0 reporting version.
- If a report configuration in the **Vulnerability Management 1.0 → Reporting** page is updated after rolling back to RHACS 4.2 or earlier, those updates might not be applied to the migrated report configuration when the system is upgraded again. If this happens, the details of the report configuration are added to the logs generated by the Central pod. You can manually update the report configuration by clicking **Vulnerability Management (2.0)→ Vulnerability Reporting** and using the details from the log.
- Any new report configurations created in the **Vulnerability Management 1.0 → Reporting** page are migrated when you upgrade again to RHACS version 4.3 or later.

14.1.4. Additional resources

- Creating and using deployment collections
- Migration of access scopes to collections
- Configuring the email plugin

14.2. COMMON VULNERABILITY MANAGEMENT TASKS

Common vulnerability management tasks involve identifying and prioritizing vulnerabilities, remedying them, and monitoring for new threats. Following are some common tasks you can perform from the **Vulnerability Management → Dashboard** view.

14.2.1. Finding critical CVEs impacting your infrastructure

Use the **Vulnerability Management** view for identifying CVEs that are impacting your platform the most.

Procedure

1. Go to the RHACS portal and click **Vulnerability Management** from the navigation menu.
2. Select CVEs on the **Vulnerability Management** view header.
3. In the **CVEs** view, select the **Env Impact** column header to arrange the CVEs in descending order (highest first) based on the environment impact.

14.2.2. Finding the most vulnerable image components

Use the **Vulnerability Management** view for identifying highly vulnerable image components.

Procedure

1. Go to the RHACS portal and click **Vulnerability Management** from the navigation menu.
2. From the **Vulnerability Management** view header, select **Application & Infrastructure** → **Components**.
3. In the **Components** view, select the **CVEs** column header to arrange the components in descending order (highest first) based on the CVEs count.

14.2.3. Identifying the container image layer that introduces vulnerabilities

Use the **Vulnerability Management** view to identify vulnerable components and the image layer they appear in.

Procedure

1. Go to the RHACS portal and click **Vulnerability Management** from the navigation menu.
2. Select an image from either the **Top Riskiest Images** widget or click the **Images** button at the top of the Dashboard and select an image.
3. In the **Image** details view, next to **Dockerfile**, select the expand icon to see a summary of image components.
4. Select the expand icon for specific components to get more details about the CVEs affecting the selected component.

You can also view this information by navigating to **Vulnerability Management (2.0)** → **Workload CVEs**. See "Viewing workload CVEs in Vulnerability Management (2.0)" in the "Additional Resources" section for more information.

14.2.4. Identifying Dockerfile lines in images that introduced components with CVEs

You can identify specific Dockerfile lines in an image that introduced components with CVEs.

Procedure

To view a problematic line:

1. Go to the RHACS portal and click **Vulnerability Management** from the navigation menu.

2. Select an image from either the **Top Riskiest Images** widget or click the **Images** button at the top of the Dashboard and select an image.

3. In the **Image** details view, under **Image Findings**, CVEs are listed in the **Observed CVEs**, **Deferred CVEs**, and **False positive CVEs** tabs.

4. Locate the CVE you want to examine further. In the **Affected Components** column, click on the **<number> Components** link to view a list of components affected by the CVE. You can perform the following actions in this window:

   - Click the expand icon next to a specific component to view the Dockerfile line in the image that introduced the CVE. To address the CVE, you need to change this line in the Dockerfile; for example, you can upgrade the component.

   - Click the name of the component to go to the **Component Summary** page and view more information about the component.

You can also view this information by navigating to **Vulnerability Management (2.0)→ Workload CVEs**. See "Viewing workload CVEs in Vulnerability Management (2.0)" in the "Additional Resources" section for more information.

14.2.5. Viewing details only for fixable CVEs

Use the **Vulnerability Management** view to filter and show only the fixable CVEs.

Procedure

```
 In the {product-title-short} portal, go to *Vulnerability Management*.
. From the *Vulnerability Management* view header, select *Filter CVEs* -> *Fixable*.
```

14.2.6. Identifying the operating system of the base image

Use the **Vulnerability Management** view to identify the operating system of the base image.

Procedure

1. Go to the RHACS portal and click **Vulnerability Management** from the navigation menu.

2. From the **Vulnerability Management** view header, select **Images**.

3. View the base operating system (OS) and OS version for all images under the **Image OS** column.

4. Select an image to view its details. The base operating system is also available under the **Image Summary → Details and Metadata** section.

NOTE

Red Hat Advanced Cluster Security for Kubernetes lists the **Image OS** as **unknown** when either:

- The operating system information is not available, or
- If the image scanner in use does not provide this information.

Docker Trusted Registry, Google Container Registry, and Anchore do not provide this information.

You can also view this information by navigating to **Vulnerability Management (2.0)** → **Workload CVEs**. See "Viewing workload CVEs in Vulnerability Management (2.0)" in the "Additional Resources" section for more information.

14.2.7. Identifying top risky objects

Use the **Vulnerability Management** view for identifying the top risky objects in your environment. The **Top Risky** widget displays information about the top risky images, deployments, clusters, and namespaces in your environment. The risk is determined based on the number of vulnerabilities and their CVSS scores.

Procedure

1. Go to the RHACS portal and click **Vulnerability Management** from the navigation menu.
2. Select the **Top Risky** widget header to choose between riskiest images, deployments, clusters, and namespaces.
   The small circles on the chart represent the chosen object (image, deployment, cluster, namespace). Hover over the circles to see an overview of the object they represent. And select a circle to view detailed information about the selected object, its related entities, and the connections between them.

   For example, if you are viewing **Top Risky Deployments by CVE Count and CVSS score**, each circle on the chart represents a deployment.

   - When you hover over a deployment, you see an overview of the deployment, which includes deployment name, name of the cluster and namespace, severity, risk priority, CVSS, and CVE count (including fixable).
   - When you select a deployment, the **Deployment** view opens for the selected deployment. The **Deployment** view shows in-depth details of the deployment and includes information about policy violations, common vulnerabilities, CVEs, and riskiest images for that deployment.
3. Select **View All** on the widget header to view all objects of the chosen type. For example, if you chose **Top Risky Deployments by CVE Count and CVSS score**, you can select **View All** to view detailed information about all deployments in your infrastructure.

14.2.8. Identifying top riskiest images and components

Similar to the **Top Risky**, the **Top Riskiest** widget lists the names of the top riskiest images and components. This widget also includes the total number of CVEs and the number of fixable CVEs in the listed images.

Procedure

1. Go to the RHACS portal and click **Vulnerability Management** from the navigation menu.
2. Select the **Top Riskiest Images** widget header to choose between the riskiest images and components. If you are viewing **Top Riskiest Images**:
   - When you hover over an image in the list, you see an overview of the image, which includes image name, scan time, and the number of CVEs along with severity (critical, high, medium, and low).
   - When you select an image, the **Image** view opens for the selected image. The **Image** view shows in-depth details of the image and includes information about CVEs by CVSS score, top riskiest components, fixable CVEs, and Dockerfile for the image.
3. Select **View All** on the widget header to view all objects of the chosen type. For example, if you chose **Top Riskiest Components**, you can select **View All** to view detailed information about all components in your infrastructure.

14.2.9. Viewing the Dockerfile for an image

Use the **Vulnerability Management** view to find the root cause of vulnerabilities in an image. You can view the Dockerfile and find exactly which command in the Dockerfile introduced the vulnerabilities and all components that are associated with that single command.

The Dockerfile section shows information about:

- All the layers in the Dockerfile
- The instructions and their value for each layer
- The components included in each layer
- The number of CVEs in components for each layer

When there are components introduced by a specific layer, you can select the expand icon to see a summary of its components. If there are any CVEs in those components, you can select the expand icon for an individual component to get more details about the CVEs affecting that component.

Procedure

1. In the RHACS portal, go to **Vulnerability Management**.
2. Select an image from either the **Top Riskiest Images** widget or click the **Images** button at the top of the Dashboard and select an image.
3. In the **Image** details view, next to **Dockerfile**, select the expand icon to see a summary of instructions, values, creation date, and components.
4. Select the expand icon for an individual component to view more information.

You can also view this information by navigating to **Vulnerability Management (2.0)** → **Workload CVEs**. See "Viewing workload CVEs in Vulnerability Management (2.0)" in the "Additional Resources" section for more information.

14.2.10. Disabling identifying vulnerabilities in nodes

Identifying vulnerabilities in nodes is enabled by default. You can disable it from the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Integrations**.
2. Under **Image Integrations**, select **StackRox Scanner**.
3. From the list of scanners, select **StackRox Scanner** to view its details.
4. Remove the **Node Scanner** option from **Types**.
5. Select **Save**.

14.2.11. Scanning inactive images

Red Hat Advanced Cluster Security for Kubernetes (RHACS) scans all active (deployed) images every 4 hours and updates the image scan results to reflect the latest vulnerability definitions.

You can also configure RHACS to scan inactive (not deployed) images automatically.

Procedure

1. In the RHACS portal, go to **Vulnerability Management (2.0)** → **Workload CVEs (Tech preview)**.
2. Click **<number> Images** to display a list of images and locate the image you want to watch.
3. Click the overflow menu, ⋮, and then select **Watch image**. RHACS then scans the image and shows an error or success message.
4. (Optional) To remove a watched image, click the overflow menu, ⋮, and then select **Unwatch image**.
5. (Optional) You can view the list of all watched images and add additional images to watch by clicking **Manage watched images** in the page header.

   IMPORTANT

   In the RHACS portal, click **Platform Configuration** → **System Configuration** to view the data retention configuration.

   All the data related to the image removed from the watched image list continues to appear in the RHACS portal for the number of days mentioned on the **System Configuration** page and is only removed after that period is over.

6. Click **Close** to return to the **Workload CVEs** page.

14.2.12. Creating policies to block specific CVEs

You can create new policies or add specific CVEs to an existing policy from the **Vulnerability Management** view.

Procedure

1. Click **CVEs** from the **Vulnerability Management** view header.
2. You can select the checkboxes for one or more CVEs, and then click **Add selected CVEs to Policy** (**add** icon) or move the mouse over a CVE in the list, and select the **Add** icon.
3. For **Policy Name**:
   - To add the CVE to an existing policy, select an existing policy from the drop-down list box.
   - To create a new policy, enter the name for the new policy, and select **Create <policy_name>**.
4. Select a value for **Severity**, either **Critical**, **High**, **Medium**, or **Low**.
5. Choose the **Lifecycle Stage** to which your policy is applicable, from **Build**, or **Deploy**. You can also select both life-cycle stages.
6. Enter details about the policy in the **Description** box.
7. Turn off the **Enable Policy** toggle if you want to create the policy but enable it later. The **Enable Policy** toggle is on by default.
8. Verify the listed CVEs which are included in this policy.
9. Click **Save Policy**.

14.2.13. Viewing recently detected vulnerabilities

The **Recently Detected Vulnerabilities** widget on the **Vulnerability Management** view shows a list of recently discovered vulnerabilities in your scanned images, based on the scan time and CVSS score. It also includes information about the number of images affected by the CVE and its impact (percentage) on your environment.

- When you hover over a CVE in the list, you see an overview of the CVE, which includes scan time, CVSS score, description, impact, and whether it's scored by using CVSS v2 or v3.
- When you select a CVE, the **CVE** details view opens for the selected CVE. The **CVE** details view shows in-depth details of the CVE and the components, images, and deployments and deployments in which it appears.
- Select **View All** on the **Recently Detected Vulnerabilities** widget header to view a list of all the CVEs in your infrastructure. You can also filter the list of CVEs.

14.2.14. Viewing the most common vulnerabilities

The **Most Common Vulnerabilities** widget on the **Vulnerability Management** view shows a list of vulnerabilities that affect the largest number of deployments and images arranged by their CVSS score.

- When you hover over a CVE in the list, you see an overview of the CVE which includes, scan time, CVSS score, description, impact, and whether it is scored by using CVSS v2 or v3.

- When you select a CVE, the **CVE** details view opens for the selected CVE. The **CVE** details view shows in-depth details of the CVE and the components, images, and deployments and deployments in which it appears.
- Select **View All** on the **Most Common Vulnerabilities** widget header to view a list of all the CVEs in your infrastructure. You can also filter the list of CVEs. To export the CVEs as a CSV file, select **Export → Download CVES as CSV**.

14.2.15. Identifying deployments with most severe policy violations

The **Deployments with most severe policy violations**widget on the **Vulnerability Management** view shows a list of deployments and severity of vulnerabilities affecting that deployment.

- When you hover over a deployment in the list, you see an overview of the deployment, which includes the deployment name, the name of the cluster and the namespace in which the deployment exists, and the number of failing policies and their severity.
- When you select a deployment, the **Deployment** view opens for the selected deployment. The **Deployment** view shows in-depth details of the deployment and includes information about policy violations, common vulnerabilities, CVEs, and riskiest images for that deployment.
- Select **View All** on the **Most Common Vulnerabilities** widget header to view a list of all the CVEs in your infrastructure. You can also filter the list of CVEs. To export the CVEs as a CSV file, select **Export → Download CVES as CSV**.

14.2.16. Finding clusters with most Kubernetes and Istio vulnerabilities

Use the **Vulnerability Management (1.0)** view for identifying the clusters with most Kubernetes, Red Hat OpenShift, and Istio vulnerabilities (deprecated) in your environment.

The **Clusters with most orchestrator and Istio vulnerabilities**widget shows a list of clusters, ranked by the number of Kubernetes, Red Hat OpenShift, and Istio vulnerabilities (deprecated) in each cluster. The cluster on top of the list is the cluster with the highest number of vulnerabilities.

Procedure

1. Click on one of the clusters from the list to view details about the cluster. The **Cluster** view includes:
   - **Cluster Summary** section, which shows cluster details and metadata, top risky objects (deployments, namespaces, and images), recently detected vulnerabilities, riskiest images, and deployments with the most severe policy violations.
   - **Cluster Findings** section, which includes a list of failing policies and list of fixable CVEs.
   - **Related Entities** section, which shows the number of namespaces, deployments, policies, images, components, and CVEs the cluster contains. You can select these entities to view more details.
2. Click **View All** on the widget header to view the list of all clusters.

14.2.17. Identifying vulnerabilities in nodes

You can use the **Vulnerability Management** view to identify vulnerabilities in your nodes. The identified vulnerabilities include vulnerabilities in:

- Core Kubernetes components.
- Container runtimes (Docker, CRI-O, runC, and containerd).

NOTE

- Red Hat Advanced Cluster Security for Kubernetes can identify vulnerabilities in the following operating systems:
  - Amazon Linux 2
  - CentOS
  - Debian
  - Garden Linux (Debian 11)
  - Red Hat Enterprise Linux CoreOS (RHCOS)
  - Red Hat Enterprise Linux (RHEL)
  - Ubuntu (AWS, Microsoft Azure, GCP, and GKE specific versions)

Procedure

1. In the RHACS portal, go to **Vulnerability Management → Dashboard**.
2. Select **Nodes** on the **Dashboard** view header to view a list of all the CVEs affecting your nodes.
3. Select a node from the list to view details of all CVEs affecting that node.
   a. When you select a node, the **Node** details panel opens for the selected node. The **Node** view shows in-depth details of the node and includes information about CVEs by CVSS score and fixable CVEs for that node.
   b. Select **View All** on the **CVEs by CVSS score** widget header to view a list of all the CVEs in the selected node. You can also filter the list of CVEs.
   c. To export the fixable CVEs as a CSV file, select **Export as CSV** under the **Node Findings** section.

14.3. SCANNING RHCOS NODE HOSTS

For OpenShift Container Platform, Red Hat Enterprise Linux CoreOS (RHCOS) is the only supported operating system for control plane. Whereas, for node hosts, OpenShift Container Platform supports both RHCOS and Red Hat Enterprise Linux (RHEL). With Red Hat Advanced Cluster Security for Kubernetes (RHACS), you can scan RHCOS nodes for vulnerabilities and detect potential security threats.

RHACS scans RHCOS RPMs installed on the node host, as part of the RHCOS installation, for any known vulnerabilities.

First, RHACS analyzes and detects RHCOS components. Then it matches vulnerabilities for identified components by using RHEL and OpenShift 4.X Open Vulnerability and Assessment Language (OVAL) v2 security data streams.

NOTE

- If you installed RHACS by using the **roxctl** CLI, you must manually enable the RHCOS node scanning features. When you use Helm or Operator installation methods on OpenShift Container Platform, this feature is enabled by default.

Additional resources

- RHEL Versions Utilized by RHEL CoreOS and OCP

14.3.1. Enabling RHCOS node scanning

If you use OpenShift Container Platform, you can enable scanning of Red Hat Enterprise Linux CoreOS (RHCOS) nodes for vulnerabilities by using Red Hat Advanced Cluster Security for Kubernetes (RHACS).

Prerequisites

- For scanning RHCOS node hosts of the Secured cluster, you must have installed Secured cluster on OpenShift Container Platform 4.11 or later. For more information on supported managed and self-managed OpenShift Container Platform versions, see Red Hat Advanced Cluster Security for Kubernetes Support Policy.

Procedure

1. Run one of the following commands to update the compliance container.
   - For a default compliance container with metrics disabled, run the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"containers":
[{"name":"compliance","env":[{"name":"ROX_METRICS_PORT","value":"disabled"},
{"name":"ROX_NODE_SCANNING_ENDPOINT","value":"127.0.0.1:8444"},
{"name":"ROX_NODE_SCANNING_INTERVAL","value":"4h"},
{"name":"ROX_NODE_SCANNING_INTERVAL_DEVIATION","value":"24m"},
{"name":"ROX_NODE_SCANNING_MAX_INITIAL_WAIT","value":"5m"},
{"name":"ROX_RHCOS_NODE_SCANNING","value":"true"},
{"name":"ROX_CALL_NODE_INVENTORY_ENABLED","value":"true"}]}]}}}}'
```

   - For a compliance container with Prometheus metrics enabled, run the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"containers":
[{"name":"compliance","env":[{"name":"ROX_METRICS_PORT","value":":9091"},
{"name":"ROX_NODE_SCANNING_ENDPOINT","value":"127.0.0.1:8444"},
{"name":"ROX_NODE_SCANNING_INTERVAL","value":"4h"},
{"name":"ROX_NODE_SCANNING_INTERVAL_DEVIATION","value":"24m"},
{"name":"ROX_NODE_SCANNING_MAX_INITIAL_WAIT","value":"5m"},
{"name":"ROX_RHCOS_NODE_SCANNING","value":"true"},
{"name":"ROX_CALL_NODE_INVENTORY_ENABLED","value":"true"}]}]}}}}'
```

2. Update the Collector DaemonSet (DS) by taking the following steps:
   a. Add new volume mounts to Collector DS by running the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"volumes":
[{"name":"tmp-volume","emptyDir":{}},{"name":"cache-volume","emptyDir":
{"sizeLimit":"200Mi"}}]}}}}'
```

b. Add the new **NodeScanner** container by running the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"containers":
[{"command":["/scanner","--nodeinventory","--config=",""],"env":
[{"name":"ROX_NODE_NAME","valueFrom":{"fieldRef":
{"apiVersion":"v1","fieldPath":"spec.nodeName"}}},
{"name":"ROX_CLAIR_V4_SCANNING","value":"true"},
{"name":"ROX_COMPLIANCE_OPERATOR_INTEGRATION","value":"true"},
{"name":"ROX_CSV_EXPORT","value":"false"},
{"name":"ROX_DECLARATIVE_CONFIGURATION","value":"false"},
{"name":"ROX_INTEGRATIONS_AS_CONFIG","value":"false"},
{"name":"ROX_NETPOL_FIELDS","value":"true"},
{"name":"ROX_NETWORK_DETECTION_BASELINE_SIMULATION","value":"true"},
{"name":"ROX_NETWORK_GRAPH_PATTERNFLY","value":"true"},
{"name":"ROX_NODE_SCANNING_CACHE_TIME","value":"3h36m"},
{"name":"ROX_NODE_SCANNING_INITIAL_BACKOFF","value":"30s"},
{"name":"ROX_NODE_SCANNING_MAX_BACKOFF","value":"5m"},
{"name":"ROX_PROCESSES_LISTENING_ON_PORT","value":"false"},
{"name":"ROX_QUAY_ROBOT_ACCOUNTS","value":"true"},
{"name":"ROX_ROXCTL_NETPOL_GENERATE","value":"true"},
{"name":"ROX_SOURCED_AUTOGENERATED_INTEGRATIONS","value":"false"},
{"name":"ROX_SYSLOG_EXTRA_FIELDS","value":"true"},
{"name":"ROX_SYSTEM_HEALTH_PF","value":"false"},
{"name":"ROX_VULN_MGMT_WORKLOAD_CVES","value":"false"}],"image":"registry.red
hat.io/advanced-cluster-security/rhacs-scanner-slim-
rhel8:4.4.0","imagePullPolicy":"IfNotPresent","name":"node-inventory","ports":
[{"containerPort":8444,"name":"grpc","protocol":"TCP"}],"volumeMounts":
[{"mountPath":"/host","name":"host-root-ro","readOnly":true},
{"mountPath":"/tmp/","name":"tmp-volume"},{"mountPath":"/cache","name":"cache-
volume"}]}]}}}}'
```

14.3.2. Analysis and detection

When you use RHACS with OpenShift Container Platform, RHACS creates two coordinating containers for analysis and detection, the Compliance container and the Node-inventory container. The Compliance container was already a part of earlier RHACS versions. However, the Node-inventory container is new with RHACS 4.0 and works only with OpenShift Container Platform cluster nodes.

Upon start-up, the Compliance and Node-inventory containers begin the first inventory scan of Red Hat Enterprise Linux CoreOS (RHCOS) software components within five minutes. Next, the Node-inventory container scans the node’s file system to identify installed RPM packages and report on RHCOS software components. Afterward, inventory scanning occurs at periodic intervals, typically every four hours. You can customize the default interval by configuring the ROX_NODE_SCANNING_INTERVAL environment variable for the Compliance container.

14.3.3. Vulnerability matching

Central services, which include Central and Scanner, perform vulnerability matching. Scanner uses Red Hat’s Open Vulnerability and Assessment Language (OVAL) v2 security data streams to match vulnerabilities on Red Hat Enterprise Linux CoreOS (RHCOS) software components.

Unlike the earlier versions, RHACS 4.0 no longer uses the Kubernetes node metadata to find the kernel and container runtime versions. Instead, it uses the installed RHCOS RPMs to assess that information.

14.3.4. Related environment variables

You can use the following environment variables to configure RHCOS node scanning on RHACS.

Table 14.1. Node-inventory configuration

Environment Variable	Description
ROX_NODE_SCANNING_CACHE_TIME	The time after which a cached inventory is considered outdated. Defaults to 90% of **ROX_NODE_SCANNING_INTERVAL** that is **3h36m**.
ROX_NODE_SCANNING_INITIAL_BACKOFF	The initial time in seconds a node scan will be delayed if a backoff file is found. The default value is **30s**.
ROX_NODE_SCANNING_MAX_BACKOFF	The upper limit of backoff. The default value is 5m, being 50% of Kubernetes restart policy stability timer.

Table 14.2. Compliance configuration

Environment Variable	Description
ROX_NODE_SCANNING_INTERVAL	The base value of the interval duration between node scans. The deafult value is **4h**.
ROX_NODE_SCANNING_INTERVAL_DEVIATION	The duration of node scans may differ from the base interval time. However, the maximum value is limited by the **ROX_NODE_SCANNING_INTERVAL**.
ROX_NODE_SCANNING_MAX_INITIAL_WAIT	The maximum wait time before the first node scan, which is randomly generated. You can set this value to **0** to disable the initial node scanning wait time. The default value is **5m**.

14.3.5. Identifying vulnerabilities in nodes

You can use the **Vulnerability Management** view to identify vulnerabilities in your nodes. The identified vulnerabilities include vulnerabilities in:

- Core Kubernetes components.
- Container runtimes (Docker, CRI-O, runC, and containerd).

NOTE

- Red Hat Advanced Cluster Security for Kubernetes can identify vulnerabilities in the following operating systems:
  - Amazon Linux 2
  - CentOS
  - Debian
  - Garden Linux (Debian 11)
  - Red Hat Enterprise Linux CoreOS (RHCOS)
  - Red Hat Enterprise Linux (RHEL)
  - Ubuntu (AWS, Microsoft Azure, GCP, and GKE specific versions)

Procedure

1. In the RHACS portal, go to **Vulnerability Management → Dashboard**.
2. Select **Nodes** on the **Dashboard** view header to view a list of all the CVEs affecting your nodes.
3. Select a node from the list to view details of all CVEs affecting that node.
   a. When you select a node, the **Node** details panel opens for the selected node. The **Node** view shows in-depth details of the node and includes information about CVEs by CVSS score and fixable CVEs for that node.
   b. Select **View All** on the **CVEs by CVSS score** widget header to view a list of all the CVEs in the selected node. You can also filter the list of CVEs.
   c. To export the fixable CVEs as a CSV file, select **Export as CSV** under the **Node Findings** section.

CHAPTER 15. RESPONDING TO VIOLATIONS

Using Red Hat Advanced Cluster Security for Kubernetes (RHACS) you can view policy violations, drill down to the actual cause of the violation, and take corrective actions.

RHACS's built-in policies identify a variety of security findings, including vulnerabilities (CVEs), violations of DevOps best practices, high-risk build and deployment practices, and suspicious runtime behaviors. Whether you use the default out-of-box security policies or use your own custom policies, RHACS reports a violation when an enabled policy fails.

15.1. VIOLATIONS VIEW

You can analyze all violations in the **Violations** view and take corrective action.

In the RHACS portal, go to **Violations** to see the discovered violations.

The **Violations** view shows a list of violations with the following attributes for each row:

- **Policy**: The name of the violated policy.
- **Entity**: The entity where the violation occurred.
- **Type**: The type of entity, such as a deployment, namespace, or cluster.
- **Enforced**: Indicates if the policy was enforced when the violation occurred.
- **Severity**: Indicates the severity as **Low**, **Medium**, **High**, or **Critical**.
- **Categories**: The policy categories. Policy categories are listed in **Platform Configuration → Policy Management** in the **Policy categories** tab.
- **Lifecycle**: The lifecycle stages to which the policy applies, **Build**, **Deploy**, or **Runtime**.
- **Time**: The date and time when the violation occurred.

Similar to other views, you can perform the following actions:

- Select a column heading to sort the violations in ascending or descending order.
- Use the filter bar to filter violations. See the Searching and filtering section for more information.
- Select a violation in the **Violations** view to see more details about the violation.

15.1.1. Marking violations as resolved

If a policy that has runtime violations is deleted, attempted violations are not deleted from the **Violations** page. You can manually remove the violation by marking it as resolved.

Procedure

1. Select **Violations** and locate the violation in the list of violations.
2. Click the overflow menu, ⋮ , and then select one of the following options:

- **Resolve and add to process baseline** Resolves the violation and adds the associated process to the process baseline. If the process is executed again, a new violation will be displayed.
- **Mark as resolved**: Resolves the violation.

15.2. VIEWING VIOLATION DETAILS

When you select a violation in the **Violations** view, a window opens with more information about the violation. It provides detailed information grouped by multiple tabs.

15.2.1. Violation tab

The **Violation** tab of the **Violation Details** panel explains how the policy was violated. If the policy targets deploy-phase attributes, you can view the specific values that violated the policies, such as violation names. If the policy targets runtime activity, you can view detailed information about the process that violated the policy, including its arguments and the ancestor processes that created it.

15.2.2. Deployment tab

The **Deployment** tab of the **Details** panel displays details of the deployment to which the violation applies.

Overview section
The **Deployment overview** section lists the following information:

- **Deployment ID**: The alphanumeric identifier for the deployment.
- **Deployment name**: The name of the deployment.
- **Deployment type**: The type of the deployment.
- **Cluster**: The name of the cluster where the container is deployed.
- **Namespace**: The unique identifier for the deployed cluster.
- **Replicas**: The number of the replicated deployments.
- **Created**: The time and date when the deployment was created.
- **Updated**: The time and date when the deployment was updated.
- **Labels**: The labels that apply to the selected deployment.
- **Annotations**: The annotations that apply to the selected deployment.
- **Service Account**: The name of the service account for the selected deployment.

Container configuration section
The **Container configuration** section lists the following information:

- **containers**: For each container, provides the following information:
  - **Image name**: The name of the image for the selected deployment. Click the name to view more information about the image.

- **Resources**: This section provides information for the following fields:
  - **CPU request (cores)**: The number of cores requested by the container.
  - **CPU limit (cores)**: The maximum number of cores that can be requested by the container.
  - **Memory request (MB)**: The memory size requested by the container.
  - **Memory limit (MB)**: The maximum memory that can be requested by the container.
- **volumes**: Volumes mounted in the container, if any.
- **secrets**: Secrets associated with the selected deployment. For each secret, provides information for the following fields:
  - **Name**: Name of the secret.
  - **Container path**: Location where the secret is stored.
- **Name**: The name of the location where the service will be mounted.
- **Source**: The data source path.
- **Destination**: The path where the data is stored.
- **Type**: The type of the volume.

Port configuration section
The **Port configuration** section provides information about the ports in the deployment, including the following fields:

- **ports**: All ports exposed by the deployment and any Kubernetes services associated with this deployment and port if they exist. For each port, the following fields are listed:
  - **containerPort**: The port number exposed by the deployment.
  - **protocol**: Protocol, such as, TCP or UDP, that is used by the port.
  - **exposure**: Exposure method of the service, for example, load balancer or node port.
  - **exposureInfo**: This section provides information for the following fields:
    - **level**: Indicates if the service exposing the port internally or externally.
    - **serviceName**: Name of the Kubernetes service.
    - **serviceID**: ID of the Kubernetes service as stored in RHACS.
    - **serviceClusterIp**: The IP address that another deployment or service *within the cluster* can use to reach the service. This is not the external IP address.
    - **servicePort**: The port used by the service.
    - **nodePort**: The port on the node where external traffic comes into the node.
    - **externalIps**: The IP addresses that can be used to access the service externally, from outside the cluster, if any exist. This field is not available for an internal service.

Security context section

The **Security context** section lists whether the container is running as a privileged container.

- **Privileged**:
  - **true** if it is **privileged**.
  - **false** if it is **not privileged**.

Network policy section

The **Network policy** section lists the namespace and all network policies in the namespace containing the violation. Click on a network policy name to view the full YAML file of the network policy.

15.2.3. Policy tab

The **Policy** tab of the **Details** panel displays details of the policy that caused the violation.

Policy overview section

The **Policy overview** section lists the following information:

- **Severity**: A ranking of the policy (critical, high, medium, or low) for the amount of attention required.
- **Categories**: The policy category of the policy. Policy categories are listed in **Platform Configuration** → **Policy Management** in the **Policy categories** tab.
- **Type**: Whether the policy is user generated (policies created by a user) or a system policy (policies built into RHACS by default).
- **Description**: A detailed explanation of what the policy alert is about.
- **Rationale**: Information about the reasoning behind the establishment of the policy and why it matters.
- **Guidance**: Suggestions on how to address the violation.
- **MITRE ATT&CK**: Indicates if there are MITRE tactics and techniques that apply to this policy.

Policy behavior

The **Policy behavior** section provides the following information:

- **Lifecycle Stage**: Lifecycle stages that the policy belongs to, **Build**, **Deploy**, or **Runtime**.
- **Event source**: This field is only applicable if the lifecycle stage is **Runtime**. It can be one of the following:
  - **Deployment**: RHACS triggers policy violations when event sources include process and network activity, pod exec, and pod port forwarding.
  - **Audit logs**: RHACS triggers policy violations when event sources match Kubernetes audit log records.
- **Response**: The response can be one of the following:
  - **Inform**: Policy violations generate a violation in the violations list.
  - **Inform and enforce**: The violation is enforced.

- **Enforcement**: If the response is set to **Inform and enforce**, lists the type of enforcement that is set for the following stages:
  - **Build**: RHACS fails your continuous integration (CI) builds when images match the criteria of the policy.
  - **Deploy**: For the **Deploy** stage, RHACS blocks the creation and update of deployments that match the conditions of the policy if the RHACS admission controller is configured and running.
    - In clusters with admission controller enforcement, the Kubernetes or OpenShift Container Platform API server blocks all noncompliant deployments. In other clusters, RHACS edits noncompliant deployments to prevent pods from being scheduled.
    - For existing deployments, policy changes only result in enforcement at the next detection of the criteria, when a Kubernetes event occurs. For more information about enforcement, see "Security policy enforcement for the deploy stage".
  - **Runtime**: RHACS deletes all pods when an event in the pods matches the criteria of the policy.

Policy criteria section
The **Policy criteria** section lists the policy criteria for the policy.

15.2.3.1. Security policy enforcement for the deploy stage

Red Hat Advanced Cluster Security for Kubernetes supports two forms of security policy enforcement for deploy-time policies: hard enforcement through the admission controller and soft enforcement by RHACS Sensor. The admission controller blocks creation or updating of deployments that violate policy. If the admission controller is disabled or unavailable, Sensor can perform enforcement by scaling down replicas for deployments that violate policy to **0**.

WARNING

Policy enforcement can impact running applications or development processes. Before you enable enforcement options, inform all stakeholders and plan how to respond to the automated enforcement actions.

15.2.3.1.1. Hard enforcement

Hard enforcement is performed by the RHACS admission controller. In clusters with admission controller enforcement, the Kubernetes or OpenShift Container Platform API server blocks all noncompliant deployments. The admission controller blocks **CREATE** and **UPDATE** operations. Any pod create or update request that satisfies a policy configured with deploy-time enforcement enabled will fail.

NOTE

Kubernetes admission webhooks support only **CREATE**, **UPDATE**, **DELETE**, or **CONNECT** operations. The RHACS admission controller supports only **CREATE** and **UPDATE** operations. Operations such as **kubectl patch**, **kubectl set**, and **kubectl scale** are PATCH operations, not UPDATE operations. Because PATCH operations are not supported in Kubernetes, RHACS cannot perform enforcement on PATCH operations.

For blocking enforcement, you must enable the following settings for the cluster in RHACS:

- **Enforce on Object Creates**: This toggle in the **Dynamic Configuration** section controls the behavior of the admission control service. You must have the **Configure Admission Controller Webhook to listen on Object Creates** toggle in the **Static Configuration** section turned on for this to work.
- **Enforce on Object Updates**: This toggle in the **Dynamic Configuration** section controls the behavior of the admission control service. You must have the **Configure Admission Controller Webhook to listen on Object Updates** toggle in the **Static Configuration** section turned on for this to work.

If you make changes to settings in the **Static Configuration** setting, you must redeploy the secured cluster for those changes to take effect.

15.2.3.1.2. Soft enforcement

Soft enforcement is performed by RHACS Sensor. This enforcement prevents an operation from being initiated. With soft enforcement, Sensor scales the replicas to 0, and prevents pods from being scheduled. In this enforcement, a non-ready deployment is available in the cluster.

If soft enforcement is configured, and Sensor is down, then RHACS cannot perform enforcement.

15.2.3.1.3. Namespace exclusions

By default, RHACS excludes certain administrative namespaces, such as the **stackrox**, **kube-system**, and **istio-system** namespaces, from enforcement blocking. The reason for this is that some items in these namespaces must be deployed for RHACS to work correctly.

15.2.3.1.4. Enforcement on existing deployments

For existing deployments, policy changes only result in enforcement at the next detection of the criteria, when a Kubernetes event occurs. If you make changes to a policy, you must reassess policies by selecting **Policy Management** and clicking **Reassess All**. This action applies deploy policies on all existing deployments regardless of whether there are any new incoming Kubernetes events. If a policy is violated, then RHACS performs enforcement.

Additional resources

- Using admission controller enforcement

CHAPTER 16. CREATING AND USING DEPLOYMENT COLLECTIONS

You can use collections in RHACS to define and name a group of resources by using matching patterns. You can then configure system processes to use these collections.

Currently, collections are available only under the following conditions:

- Collections are available only for deployments.
- You can only use collections with vulnerability reporting. See "Vulnerability reporting" in the Additional resources section for more information.
- Deployment collections are only available to RHACS customers if they are using the PostgreSQL database.

NOTE

By default, RHACS Cloud Service uses the PostgreSQL database, and it is also used by default when installing RHACS release 4.0 and later. RHACS customers using an earlier release than 3.74 can migrate to the PostgreSQL database with help from Red Hat.

16.1. PREREQUISITES

A user account must have the following permissions to use the Collections feature:

- **WorkflowAdministration**: You must have **Read** access to view collections and **Write** access to add, change, or delete collections.
- **Deployment**: You need **Read Access** or **Read and Write Access** to understand how configured rules will match with deployments.

These permissions are included in the **Admin** system role. For more information about roles and permissions, see "Managing RBAC in RHACS" in "Additional resources".

16.2. UNDERSTANDING DEPLOYMENT COLLECTIONS

Deployment collections are only available to RHACS customers using the PostgreSQL database. By default, RHACS Cloud Service uses the PostgreSQL database, and it is also used by default when installing RHACS release 4.0 and later. RHACS customers using an earlier release than 3.74 can migrate to the PostgreSQL database with help from Red Hat.

An RHACS collection is a user-defined, named reference. It defines a logical grouping by using selection rules. Those rules can match a deployment, namespace, or cluster name or label. You can specify rules by using exact matches or regular expressions. Collections are resolved at run time and can refer to objects that do not exist at the time of the collection definition. Collections can be constructed by using other collections to describe complex hierarchies.

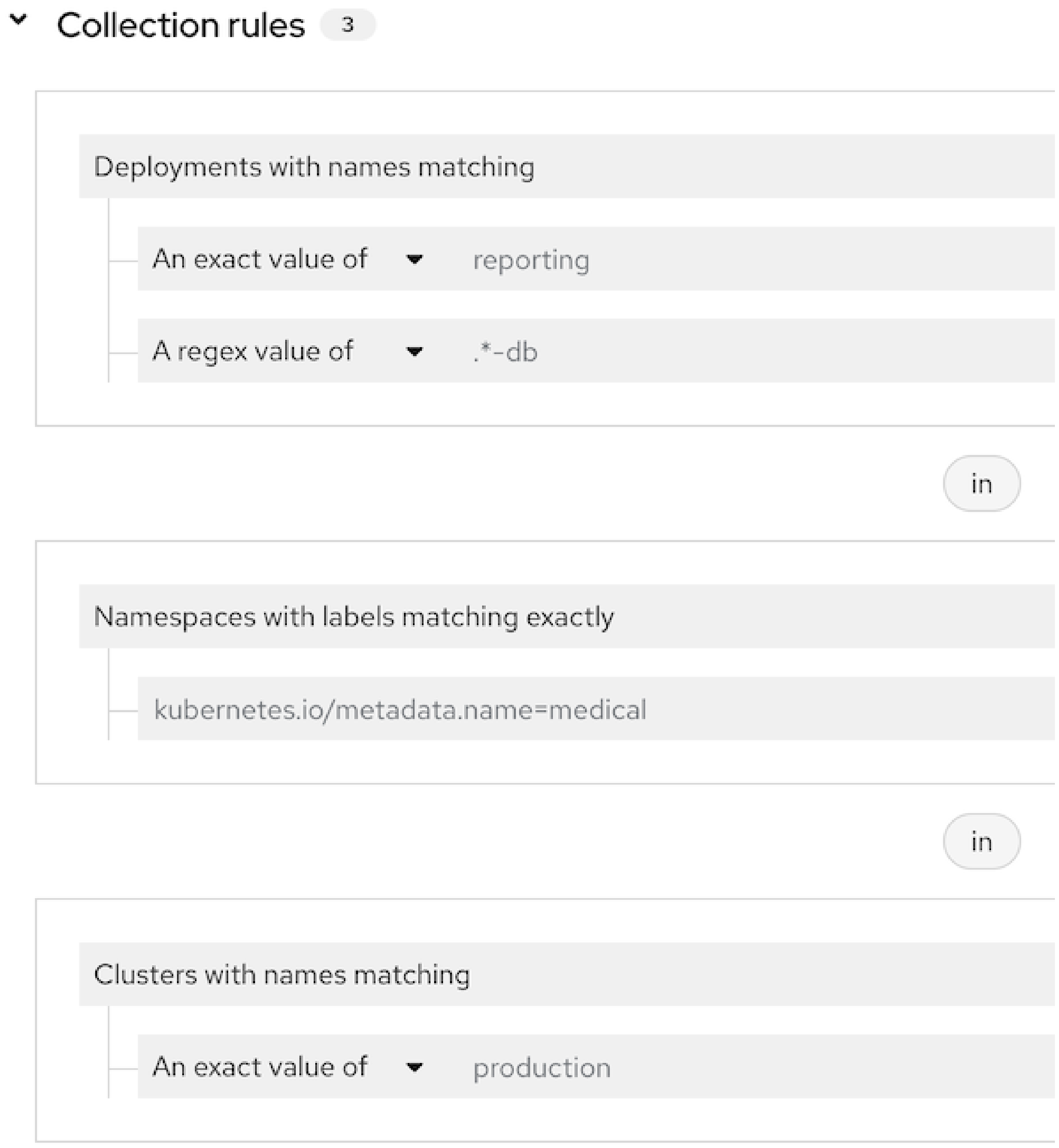
Collections provide you with a language to describe how your dynamic infrastructure is organized, eliminating the need for cloning and repetitive editing of RHACS properties such as inclusion and exclusion scopes.

You can use collections to identify any group of deployments in your system, such as:

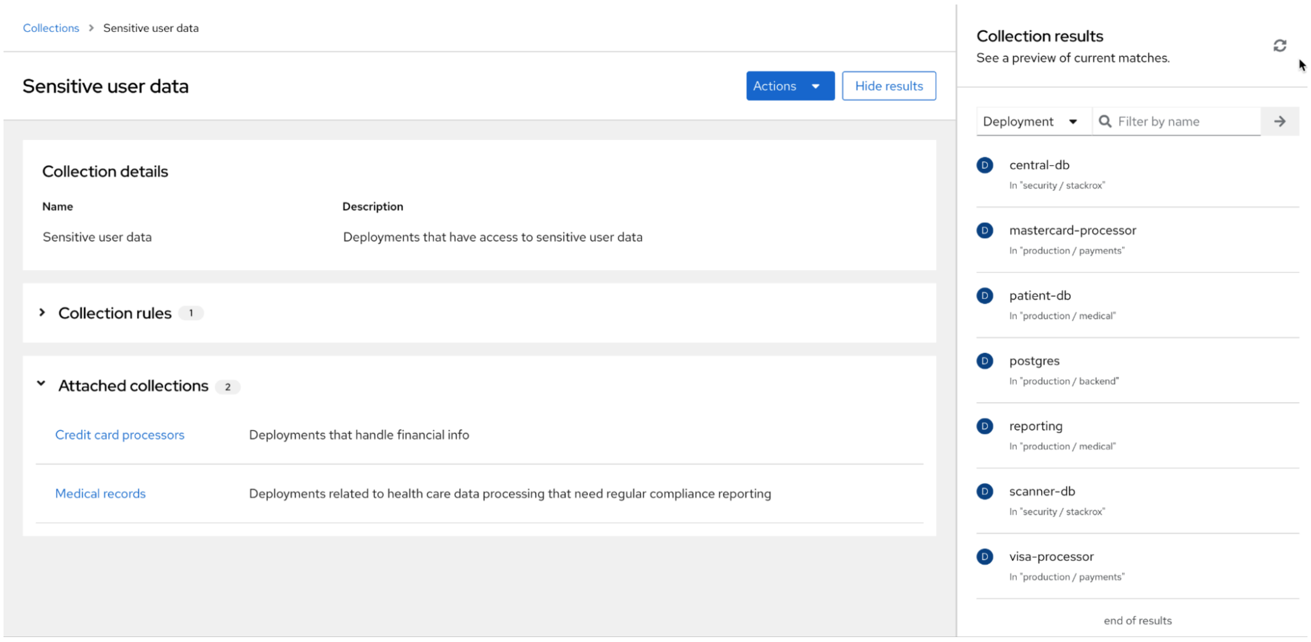
- An infrastructure area that is owned by a specific development team
- An application that requires different policy exceptions when running in a development or in a production cluster
- A distributed application that spans multiple namespaces, defined with a common deployment label
- An entire production or test environment

Collections can be created and managed by using the RHACS portal. The collection editor helps you apply selection rules at the deployment, namespace, and cluster level. You can use simple and complex rules, including regular expression.

You can define a collection by selecting one or more deployments, namespaces, or clusters, as shown in the following image. This image shows a collection that contains deployments with the name reporting or that contain **db** in the name. The collection includes deployments matching those names in the namespace with a specific label of **kubernetes.io/metadata.name=medical**, and in clusters named **production**.


Collection rules 3
Deployments with names matching
An exact value of
reporting
A regex value of
.*-db
in
Namespaces with labels matching exactly
kubernetes.io/metadata.name=medical
in
Clusters with names matching
An exact value of
production

The collection editor also helps you to describe complex hierarchies by attaching, or nesting, other collections. The editor provides a real-time preview side panel that helps you understand the rules you are applying by showing the resulting matches to the rules that you have configured. The following image provides an example of results from a collection named "Sensitive User Data" with a set of collection rules (not shown). The "Sensitive User Data" collection has two attached collections, "Credit card processors" and "Medical records" and each of those collections have their own collection rules. The results shown in the side panel include items that match the rules configured for all three collections.



Collection details

Name	Description
Sensitive user data	Deployments that have access to sensitive user data

Collection rules 1

Attached collections 2

Credit card processors	Deployments that handle financial info
Medical records	Deployments related to health care data processing that need regular compliance reporting

- central-db In "security / stackrox"
- mastercard-processor In "production / payments"
- patient-db In "production / medical"
- postgres In "production / backend"
- reporting In "production / medical"
- scanner-db In "security / stackrox"
- visa-processor In "production / payments"

end of results

16.3. ACCESSING DEPLOYMENT COLLECTIONS

To use collections, click **Platform Configuration** → **Collections**. The page displays a list of currently-configured collections. You can perform the following actions:

- Search for collections by entering text in the **Search by name** field, and then press **→**.
- Click on a collection in the collection list to view the collection in read-only mode.
- Click on ⋮ for an existing collection to edit, clone, or delete it.

  NOTE

  You cannot delete a collection that is actively used in RHACS.

- Click **Create collection** to create a new deployment collection.

16.4. CREATING DEPLOYMENT COLLECTIONS

When creating a collection, you must name it and define the rules for the collection.

Procedure

1. In the Collections page, click **Create collection**.
2. Enter the name and description for the collection.
3. In the **Collection rules** section, you must perform at least one of the following actions:
   - Define the rules for the collection: See the "Creating collection rules" section for more information.
   - Attach existing collections to the collection: See the "Adding attached collections" section for more information.

4. The results of your rule configuration or choosing attached collections are available in the **Collection results** live preview panel. Click **Hide results** to remove this panel from display.

5. Click **Save**.

16.4.1. Creating collection rules

When creating collections, you must configure at least one rule or attach another collection to the new collection that you are creating.

NOTE

Currently, collections are available only for deployments.

Configure rules to select the resources to include in the collection. Use the preview panel to see the results of the collection rules as you configure them. You can configure rules in any order.

Procedure

1. In the **Deployments** section, select one of the following options from the drop-down list:
   - **All deployments**: Includes all deployments in the collection. If you select this option, you must filter the collection by using namespaces or clusters or by attaching another collection.
   - **Deployments with names matching** Click this option to select by name and then click one of the following options:
     - Select **An exact value of** and enter the exact name of the deployment.
     - Select **A regex value of** to use regular expression to search for a deployment. This option is useful if you do not know the exact name of the deployment. A regular expression is a string of letters, numbers, and symbols that defines a pattern. RHACS uses this pattern to match characters or groups of characters and return results. For more information about regular expression, see "Regular-Expressions.info" in the "Additional resources" section.
   - **Deployments with labels matching exactly**. Click this option to select deployments with labels that match the exact text that you enter. The label must be a valid Kubernetes label in the format of **key=value**.
2. Optional: To add more deployments with names or labels that match additional criteria for inclusion, click **OR** and configure another exact or regular expression value.

The following example provides the steps for configuring a collection for a medical application. In this example, you want your collection to include the **reporting** deployment, a database called **patient-db**, and you want to select namespaces with labels where **key = kubernetes.io/metadata.name** and **value = medical**. For this example, perform the following steps:

1. In **Collection rules**, select **Deployments with names matching**.

2. Click **An exact value of** and enter **reporting**.

3. Click **OR**.

4. Click **A regex value of** and enter **.*-db** to select all deployments with a name ending in **db** in your environment. The **regex value** option uses regular expression for pattern matching; for

more information about regular expression, see "Regular-Expressions.info" in the Additional resources section. The panel on the right might display databases that you do not want to include. You can exclude those databases by using additional filters. For example:

a. Filter by namespace labels by clicking **Namespaces with labels matching exactly** and entering **kubernetes.io/metadata.name=medical** to include only deployments in the namespace that is labeled **medical**.

b. If you know the name of the namespace, click **Namespaces with names matching** and enter the name.

16.4.2. Adding attached collections

Grouping collections and adding them to other collections can be useful if you want to create small collections based on deployments. You can reuse and combine those smaller collections into larger, hierarchical collections. To add additional collections to a collection that you are creating:

1. Perform one of the following actions:
   - Enter text in the **Filter by name** field and press **→** to view matching results.
   - Click the name of a collection from the **Available collections** list to view information about the collection, such as the name and rules for the collection and the deployments that match that collection.
2. After viewing collection information, close the window to return to the **Attached collections** page.
3. Click **+Attach**. The **Attached collections** section lists the collections that you attached.

   NOTE

   When you add an attached collection, the attached collection contains results based on the configured selection rules. For example, if an attached collection includes resources that would be filtered out by the rules used in the parent collection, then those items are still added to the parent collection because of the rules in the attached collection. Attached collections extend the original collection using an **OR** operator.

4. Click **Save**.

16.5. MIGRATION OF ACCESS SCOPES TO COLLECTIONS

Database changes in RHACS from **rocksdb** to PostgreSQL are provided as a Technology Preview beginning with release 3.74 and are generally available in release 4.0. When the database is migrated from **rocksdb** to PostgreSQL, existing access scopes used in vulnerability reporting are migrated to collections. You can verify that the migration resulted in the correct configuration for your existing reports by navigating to **Vulnerability Management → Reporting** and viewing the report information.

The migration process creates collection objects for access scopes that were used in report configurations. RHACS generates two or more collections for a single access scope, depending on the complexity of the access scope. The generated collections for a given access scope include the following types:

- Embedded collections: To mimic the exact selection logic of the original access scope, RHACS generates one or more collections where matched deployments result in the same selection of clusters and namespaces as the original access scope. The collection name is in the format of **System-generated embedded collection *number* for the scope** where *number* is a number starting from 0.

> NOTE
>
> These embedded collections will not have any attached collections. They have cluster and namespace selection rules, but no deployment rules because the original access scopes did not filter on deployments.

- Root collection for the access scope: This collection is added to the report configurations. The collection name is in the format of **System-generated root collection for the scope**. This collection does not define any rules, but attaches one or more embedded collections. The combination of these embedded collections results in the same selection of clusters and namespaces as the original access scope.

For access scopes that define cluster or namespace label selectors, RHACS can only migrate those scopes that have the 'IN' operator between the key and values. Access scopes with label selectors that were created by using the RHACS portal used the 'IN' operator by default. Migration of scopes that used the 'NOT_IN', 'EXISTS' and 'NOT_EXISTS' operators is not supported. If a collection cannot be created for an access scope, log messages are created during the migration. Log messages have the following format:

> Failed to create collections for scope _scope-name_: Unsupported operator NOT_IN in scope's label selectors. Only operator 'IN' is supported.
> The scope is attached to the following report configurations: [list of report configs]; Please manually create an equivalent collection and edit the listed report configurations to use this collection. Note that reports will not function correctly until a collection is attached.

You can also click the report in **Vulnerability Management** → **Reporting** to view the report information page. This page contains a message if a report needs a collection attached to it.

> NOTE
>
> The original access scopes are not removed during the migration. If you created an access scope only for use in filtering vulnerability management reports, you can manually remove the access scope.

16.6. MANAGING COLLECTIONS BY USING THE API

You can configure collections by using the **CollectionService** API object. For example, you can use **CollectionService_DryRunCollection** to return a list of results equivalent to the live preview panel in the RHACS portal. For more information, go to **Help** → **API reference** in the RHACS portal.

Additional resources

- Managing RBAC in RHACS
- Vulnerability reporting
- Using regular expression: Regular-Expressions.info

CHAPTER 17. SEARCHING AND FILTERING

The ability to instantly find resources is important to safeguard your cluster. Use Red Hat Advanced Cluster Security for Kubernetes search feature to find relevant resources faster. For example, you can use it to find deployments that are exposed to a newly published CVE or find all deployments that have external network exposure.

17.1. SEARCH SYNTAX

A search query is made up of two parts:

- An attribute that identifies the resource type you want to search for.
- A search term that finds the matching resource.

For example, to find all violations in the **visa-processor** deployment, the search query is **Deployment:visa-processor**. In this search query, **Deployment** is the attribute and **visa-processor** is the search term.

NOTE

You must select an attribute before you can use search terms. However, in some views, such as the **Risk** view and the **Violations** view, Red Hat Advanced Cluster Security for Kubernetes automatically applies the relevant attribute based on the search term you enter.

- You can use multiple attributes in your query. When you use more than one attribute, the results only include the items that match all attributes.

  Example

  When you search for **Namespace:frontend CVE:CVE-2018-11776**, it returns only those resources which violate CVE-2018-11776 in the **frontend** namespace.

- You can use more than one search term with each attribute. When you use more than one search term, the results include all items that match any of the search terms.

  Example

  If you use the search query **Namespace: frontend backend**, it returns matching results from the namespace **frontend** or **backend**.

- You can combine multiple attribute and search term pairs.

  Example

  The search query **Cluster:production Namespace:frontend CVE:CVE-2018-11776** returns all resources which violate CVE-2018-11776 in the **frontend** namespace in the **production** cluster.

- Search terms can be part of a word, in which case Red Hat Advanced Cluster Security for Kubernetes returns all matching results.

  Example

  If you search for **Deployment:def**, the results include all deployments starting with **def**.

- To explicitly search for a specific term, use the search terms inside quotes.

Example

When you search for **Deployment:"def"**, the results only include the deployment **def**.

- You can also use regular expressions by using **r/** before your search term.

  Example

  When you search for **Namespace:r/st.*x**, the results include matches from namespace **stackrox** and **stix**.

- Use **!** to indicate the search terms that you do not want in results.

  Example

  If you search for **Namespace:!stackrox**, the results include matches from all namespaces except the **stackrox** namespace.

- Use the comparison operators **>**, **<**, **=**, **>=**, or **<=** to match a specific value or range of values.

  Example

  If you search for **CVSS:>=6**, the results include all vulnerabilities with Common Vulnerability Scoring System (CVSS) score 6 or higher.

17.2. SEARCH AUTOCOMPLETE

As you enter your query, Red Hat Advanced Cluster Security for Kubernetes automatically displays relevant suggestions for the attributes and the search terms.

17.3. USING GLOBAL SEARCH

By using global search you can search across all resources in your environment. Based on the resource type you use in your search query, the results are grouped in the following categories:

- All results (Lists matching results across all categories)
- Clusters
- Deployments
- Images
- Namespaces
- Nodes
- Policies
- Policy categories [1]
- Roles
- Role bindings
- Secrets

- Service accounts
- Users and groups
- Violations

1. The **Policy categories** option is only available if you use the following:
   - PostgreSQL as a backend database in Red Hat Advanced Cluster Security for Kubernetes (RHACS).
   - Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service).

These categories are listed as a table on the RHACS portal global search page and you can click on the category name to identify results belonging to the selected category.

To do a global search, in the RHACS portal, select **Search**.

17.4. USING LOCAL PAGE FILTERING

You can use local page filtering from within all views in the RHACS portal. Local page filtering works similar to the global search, but only relevant attributes are available. You can select the search bar to show all available attributes for a specific view.

17.5. COMMON SEARCH QUERIES

Here are some common search queries you can run with Red Hat Advanced Cluster Security for Kubernetes.

Finding deployments that are affected by a specific CVE

Query	Example
CVE:<CVE_number>	**CVE:CVE-2018-11776**

Finding privileged running deployments

Query	Example
Privileged:<true_or_false>	**Privileged:true**

Finding deployments that have external network exposure

Query	Example
Exposure Level:<level>	**Exposure Level:External**

Finding deployments that are running specific processes

Query	Example
Process Name:<process_name>	**Process Name:bash**

Finding deployments that have serious but fixable vulnerabilities

Query	Example
CVSS:<expression_and_score>	**CVSS:>=6 Fixable:.***

Finding deployments that use passwords exposed through environment variables

Query	Example
Environment Key:<query>	**Environment Key:r/.*pass.***

Finding running deployments that have particular software components in them

Query	Example
Component:<component_name>	**Component:libgpg-error** or **Component:sudo**

Finding users or groups

Use Kubernetes Labels and Selectors, and Annotations to attach metadata to your deployments. You can then query based on the applied annotations and labels to identify individuals or groups.

Finding who owns a particular deployment

Query	Example
Deployment:<deployment_name> Label:<key_value> or **Deployment:<deployment_name> Annotation:<key_value>**	**Deployment:app-server Label:team=backend**

Finding who is deploying images from public registries

Query	Example
Image Registry:<registry_name> Label:<key_value> or **Image Registry:<registry_name> Annotation:<key_value>**	**Image Registry:docker.io Label:team=backend**

Finding who is deploying into the default namespace

Query	Example
Namespace:default Label:<key_value> or **Namespace:default Annotation:<key_value>**	**Namespace:default Label:team=backend**

17.6. SEARCH ATTRIBUTES

Following is the list of search attributes that you can use while searching and filtering in Red Hat Advanced Cluster Security for Kubernetes.

Attribute	Description
Add Capabilities	Provides the container with additional Linux capabilities, for instance the ability to modify files or perform network operations.
Annotation	Arbitrary non-identifying metadata attached to an orchestrator object.
CPU Cores Limit	Maximum number of cores that a resource is allowed to use.
CPU Cores Request	Minimum number of cores to be reserved for a given resource.
CVE	Common Vulnerabilities and Exposures, use it with specific CVE numbers.
CVSS	Common Vulnerability Scoring System, use it with the CVSS score and greater than (>), less than (<), or equal to (=) symbols.
Category	Policy categories include DevOps Best Practices, Security Best Practices, Privileges, Vulnerability Management, Multiple, and any custom policy categories that you create.
Cert Expiration	Certificate expiration date.
Cluster	Name of a Kubernetes or OpenShift Container Platform cluster.
Cluster ID	Unique ID for a Kubernetes or OpenShift Container Platform cluster.
Cluster Role	Use **true** to search for cluster-wide roles and **false** for namespace-scoped roles.
Component	Software (daemond, docker), objects (images, containers, services), registries (repository for Docker images).
Component Count	Number of components in the image.
Component version	The version of software, objects, or registries.
Created Time	Time and date when the secret object was created.
Deployment	Name of the deployment.
Deployment Type	The type of Kubernetes controller on which the deployment is based.
Description	Description of the deployment.
Dockerfile Instruction Keyword	Keyword in the Dockerfile instructions in an image.
Dockerfile Instruction Value	Value in the Dockerfile instructions in an image.

Attribute	Description
Drop Capabilities	Linux capabilities that have been dropped from the container. For example **CAP_SETUID** or **CAP_NET_RAW**.
Enforcement	Type of enforcement assigned to the deployment. For example, **None**, **Scale to Zero Replicas**, or **Add an Unsatisfiable Node Constraint**.
Environment Key	Key portion of a label key-value string that is metadata for further identifying and organizing the environment of a container.
Environment Value	Value portion of a label key-value string that is metadata for further identifying and organizing the environment of a container.
Exposed Node Port	Port number of the exposed node port.
Exposing Service	Name of the exposed service.
Exposing Service Port	Port number of the exposed service.
Exposure Level	The type of exposure for a deployment port, for example **external** or **node**.
External Hostname	The hostname for an external port exposure for a deployment.
External IP	The IP address for an external port exposure for a deployment.
Fixable CVE Count	Number of fixable CVEs on an image.
Fixed By	The version string of a package that fixes a flagged vulnerability in an image.
Image	The name of the image.
Image Command	The command specified in the image.
Image Created Time	The time and date when the image was created.
Image Entrypoint	The entrypoint command specified in the image.
Image Pull Secret	The name of the secret to use when pulling the image, as specified in the deployment.
Image Pull Secret Registry	The name of the registry for an image pull secret.
Image Registry	The name of the image registry.
Image Remote	Indication of an image that is remotely accessible.

Attribute	Description
Image Scan Time	The time and date when the image was last scanned.
Image Tag	Identifier for an image.
Image Users	Name of the user or group that a container image is configured to use when it runs.
Image Volumes	Names of the configured volumes in the container image.
Inactive Deployment	Use **true** to search for inactive deployments and**false** for active deployments.
Label	The key portion of a label key-value string that is metadata for further identifying and organizing images, containers, daemons, volumes, networks, and other resources.
Lifecycle Stage	The type of lifecycle stage where this policy is configured or alert was triggered.
Max Exposure Level	For a deployment, the maximum level of network exposure for all given ports/services.
Memory Limit (MB)	Maximum amount of memory that a resource is allowed to use.
Memory Request (MB)	Minimum amount of memory to be reserved for a given resource.
Namespace	The name of the namespace.
Namespace ID	Unique ID for the containing namespace object on a deployment.
Node	Name of a node.
Node ID	Unique ID for a node.
Pod Label	Single piece of identifying metadata attached to an individual pod.
Policy	The name of the security policy.
Port	Port numbers exposed by a deployment.
Port Protocol	IP protocol such as TCP or UDP used by exposed port.
Priority	Risk priority for a deployment. (Only available in **Risks** view.)
Privileged	Use **true** to search for privileged running deployments, or**false** otherwise.
Process Ancestor	Name of any parent process for a process indicator in a deployment.

Attribute	Description
Process Arguments	Command arguments for a process indicator in a deployment.
Process Name	Name of the process for a process indicator in a deployment.
Process Path	Path to the binary in the container for a process indicator in a deployment.
Process UID	Unix user ID for the process indicator in a deployment.
Read Only Root Filesystem	Use **true** to search for containers running with the root file system configured as read only.
Role	Name of a Kubernetes RBAC role.
Role Binding	Name of a Kubernetes RBAC role binding.
Role ID	Role ID to which a Kubernetes RBAC role binding is bound.
Secret	Name of the secret object that holds the sensitive information.
Secret Path	Path to the secret object in the file system.
Secret Type	Type of the secret, for example, certificate or RSA public key.
Service Account	Service account name for a service account or deployment.
Severity	Indication of level of importance of a violation: Critical, High, Medium, Low.
Subject	Name for a subject in Kubernetes RBAC.
Subject Kind	Type of subject in Kubernetes RBAC, such as **SERVICE_ACCOUNT**, **USER** or **GROUP**.
Taint Effect	Type of taint currently applied to a node.
Taint Key	Key for a taint currently applied to a node.
Taint Value	Allowed value for a taint currently applied to a node.
Toleration Key	Key for a toleration applied to a deployment.
Toleration Value	Value for a toleration applied to a deployment.
Violation	A notification displayed in the **Violations** page when the conditions specified by a policy have not been met.

Attribute	Description
Violation State	Use it to search for resolved violations.
Violation Time	Time and date that a violation first occurred.
Volume Destination	Mount path of the data volume.
Volume Name	Name of the storage.
Volume ReadOnly	Use **true** to search for volumes that are mounted as read only.
Volume Source	Indicates the form in which the volume is provisioned (for example, **persistentVolumeClaim** or **hostPath**).
Volume Type	The type of volume.

CHAPTER 18. MANAGING USER ACCESS

18.1. MANAGING RBAC IN RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES

Red Hat Advanced Cluster Security for Kubernetes (RHACS) comes with role-based access control (RBAC) that you can use to configure roles and grant various levels of access to Red Hat Advanced Cluster Security for Kubernetes for different users.

Beginning with version 3.63, RHACS includes a scoped access control feature that enables you to configure fine-grained and specific sets of permissions that define how a given RHACS user or a group of users can interact with RHACS, which resources they can access, and which actions they can perform.

- *Roles* are a collection of permission sets and access scopes. You can assign roles to users and groups by specifying rules. You can configure these rules when you configure an authentication provider. There are two types of roles in Red Hat Advanced Cluster Security for Kubernetes:
  - System roles that are created by Red Hat and cannot be changed.
  - Custom roles, which Red Hat Advanced Cluster Security for Kubernetes administrators can create and change at any time.

  NOTE

  - If you assign multiple roles for a user, they get access to the combined permissions of the assigned roles.
  - If you have users assigned to a custom role, and you delete that role, all associated users transfer to the minimum access role that you have configured.

- *Permission sets* are a set of permissions that define what actions a role can perform on a given resource. *Resources* are the functionalities of Red Hat Advanced Cluster Security for Kubernetes for which you can set view (**read**) and modify (**write**) permissions. There are two types of permission sets in Red Hat Advanced Cluster Security for Kubernetes:
  - System permission sets, which are created by Red Hat and cannot be changed.
  - Custom permission sets, which Red Hat Advanced Cluster Security for Kubernetes administrators can create and change at any time.
- *Access scopes* are a set of Kubernetes and OpenShift Container Platform resources that users can access. For example, you can define an access scope that only allows users to access information about pods in a given project. There are two types of access scopes in Red Hat Advanced Cluster Security for Kubernetes:
  - System access scopes, which are created by Red Hat and cannot be changed.
  - Custom access scopes, which Red Hat Advanced Cluster Security for Kubernetes administrators can create and change at any time.

18.1.1. System roles

Red Hat Advanced Cluster Security for Kubernetes (RHACS) includes some default system roles that you can apply to users when you create rules. You can also create custom roles as required.

System role	Description
Admin	This role is targeted for administrators. Use it to provide read and write access to all resources.
Analyst	This role is targeted for a user who cannot make any changes, but can view everything. Use it to provide read-only access for all resources.
Continuous Integration	This role is targeted for CI (continuous integration) systems and includes the permission set required to enforce deployment policies.
Network Graph Viewer	This role is targeted for users who need to view the network graph.
None	This role has no read and write access to any resource. You can set this role as the minimum access role for all users.
Sensor Creator	RHACS uses this role to automate new cluster setups. It includes the permission set to create Sensors in secured clusters.
Vulnerability Management Approver	This role allows you to provide access to approve vulnerability deferrals or false positive requests.
Vulnerability Management Requester	This role allows you to provide access to request vulnerability deferrals or false positives.
Vulnerability Report Creator	This role allows you to create and manage vulnerability reporting configurations for scheduled vulnerability reports.

18.1.1.1. Viewing the permission set and access scope for a system role

You can view the permission set and access scope for the default system roles.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Access control**.
2. Select **Roles**.
3. Click on one of the roles to view its details. The details page shows the permission set and access scope for the slected role.

NOTE

You cannot modify permission set and access scope for the default system roles.

18.1.1.2. Creating a custom role

You can create new roles from the **Access Control** view.

Prerequisites

Prerequisites

- You must have the **Admin** role, or a role with the permission set with read and write permissions for the **AuthProvider** and **Role** resources to create, modify, and delete custom roles.
- You must create a permissions set and an access scope for the custom role before creating the role.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Access Control**.
2. Select **Roles**.
3. Click **Create role**.
4. Enter a **Name** and **Description** for the new role.
5. Select a **Permission set** for the role.
6. Select an **Access scope** for the role.
7. Click **Save**.

Additional resources

- Creating a custom permission set
- Creating a custom access scope

18.1.1.3. Assigning a role to a user or a group

You can use the RHACS portal to assign roles to a user or a group.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Access Control**.
2. From the list of authentication providers, select the authentication provider.
3. Click **Edit minimum role and rules**
4. Under the **Rules** section, click **Add new rule**.
5. For **Key**, select one of the values from **userid**, **name**, **email** or **group**.
6. For **Value**, enter the value of the user ID, name, email address or group based on the key you selected.
7. Click the **Role** drop-down menu and select the role you want to assign.
8. Click **Save**.

You can repeat these instructions for each user or group and assign different roles.

18.1.2. System permission sets

Red Hat Advanced Cluster Security for Kubernetes includes some default system permission sets that you can apply to roles. You can also create custom permission sets as required.

Permission set	Description
Admin	Provides read and write access to all resources.
Analyst	Provides read-only access for all resources.
Continuous Integration	This permission set is targeted for CI (continuous integration) systems and includes the permissions required to enforce deployment policies.
Network Graph Viewer	Provides the minimum permissions to view network graphs.
None	No read and write permissions are allowed for any resource.
Sensor Creator	Provides permissions for resources that are required to create Sensors in secured clusters.

18.1.2.1. Viewing the permissions for a system permission set

You can view the permissions for a system permission set in the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Access control**.
2. Select **Permission sets**.
3. Click on one of the permission sets to view its details. The details page shows a list of resources and their permissions for the selected permission set.

NOTE

You cannot modify permissions for a system permission set.

18.1.2.2. Creating a custom permission set

You can create new permission sets from the **Access Control** view.

Prerequisites

- You must have the **Admin** role, or a role with the permission set with read and write permissions for the **AuthProvider** and **Role** resources to create, modify, and delete permission sets.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Access Control**.
2. Select **Permission sets**.

3. Click **Create permission set**
4. Enter a **Name** and **Description** for the new permission set.
5. For each resource, under the **Access level** column, select one of the permissions from **No access**, **Read access**, or **Read and Write access**.

> **WARNING**
>
> - If you are configuring a permission set for users, you must grant read-only permissions for the following resources:
>   - **Alert**
>   - **Cluster**
>   - **Deployment**
>   - **Image**
>   - **NetworkPolicy**
>   - **NetworkGraph**
>   - **WorkflowAdministration**
>   - **Secret**
> - These permissions are preselected when you create a new permission set.
> - If you do not grant these permissions, users will experience issues with viewing pages in the RHACS portal.

6. Click **Save**.

18.1.3. System access scopes

Red Hat Advanced Cluster Security for Kubernetes includes some default system access scopes that you can apply on roles. You can also create custom access scopes as required.

Acces scope	Description
Unrestricted	Provides access to all clusters and namespaces that Red Hat Advanced Cluster Security for Kubernetes monitors.
Deny All	Provides no access to any Kubernetes and OpenShift Container Platform resources.

18.1.3.1. Viewing the details for a system access scope

You can view the Kubernetes and OpenShift Container Platform resources that are allowed and not allowed for an access scope in the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Access control**.
2. Select **Access scopes**.
3. Click on one of the access scopes to view its details. The details page shows a list of clusters and namespaces, and which ones are allowed for the selected access scope.

NOTE

You cannot modify allowed resources for a system access scope.

18.1.3.2. Creating a custom access scope

You can create new access scopes from the **Access Control** view.

Prerequisites

- You must have the **Admin** role, or a role with the permission set with read and write permissions for the **AuthProvider** and **Role** resources to create, modify, and delete permission sets.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Access control**.
2. Select **Access scopes**.
3. Click **Create access scope**.
4. Enter a **Name** and **Description** for the new access scope.
5. Under the **Allowed resources** section:
   - Use the **Cluster filter** and **Namespace filter** fields to filter the list of clusters and namespaces visible in the list.
   - Expand the **Cluster name** to see the list of namespaces in that cluster.
   - To allow access to all namespaces in a cluster, toggle the switch in the **Manual selection** column.

NOTE

Access to a specific cluster provides users with access to the following resources within the scope of the cluster:

- OpenShift Container Platform or Kubernetes cluster metadata and security information
- Compliance information for authorized clusters
- Node metadata and security information
- Access to all namespaces in that cluster and their associated security information

- To allow access to a namespace, toggle the switch in the **Manual selection** column for a namespace.

NOTE

Access to a specific namespace gives access to the following information within the scope of the namespace:

- Alerts and violations for deployments
- Vulnerability data for images
- Deployment metadata and security information
- Role and user information
- Network graph, policy, and baseline information for deployments
- Process information and process baseline configuration
- Prioritized risk information for each deployment

6. If you want to allow access to clusters and namespaces based on labels, click **Add label selector** under the **Label selection rules** section. Then click **Add rule** to specify **Key** and **Value** pairs for the label selector. You can specify labels for clusters and namespaces.

7. Click **Save**.

18.1.4. Resource definitions

Red Hat Advanced Cluster Security for Kubernetes includes many resources. The following table lists the Red Hat Advanced Cluster Security for Kubernetes resources and describes the actions that users can perform with the **read** or **write** permission.

NOTE

- To prevent privilege escalation, when you create a new token, your role's permissions limit the permission 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 to use, 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 other automation. Also, use the **roxctl central login** command for human-to-machine communication, such as **roxctl** CLI or API access.

Resource	Read permission	Write permission
Access	View configurations for single sign-on (SSO) and role-based access control (RBAC) rules that match user metadata to Red Hat Advanced Cluster Security for Kubernetes roles and users that have accessed your Red Hat Advanced Cluster Security for Kubernetes instance, including the metadata that the authentication providers give about them.	Create, modify, or delete SSO configurations and configured RBAC rules.
Administration	View the following items: • Options for data retention, security notices and other related configurations • The current logging verbosity level in Red Hat Advanced Cluster Security for Kubernetes components • Manifest content for the uploaded probe files • Existing image scanner integrations • The status of automatic upgrades • Metadata about Red Hat Advanced Cluster Security for Kubernetes service-to-service authentication • The content of the scanner bundle (download)	Edit the following items: • Data retention, security notices, and related configurations • The logging level • Support packages in Central (upload) • Image scanner integrations (create/modify/delete) • Automatic upgrades for secured clusters (enable/disable) • Service-to-service authentication credentials (revoke/re-issue)
Alert	View existing policy violations.	Resolve or edit policy violations.

Resource	Read permission	Write permission
CVE	*Internal use only*	*Internal use only*
Cluster	View existing secured clusters.	Add new secured clusters and modify or delete existing clusters.
Compliance	View compliance standards and results, recent compliance runs, and the associated completion status.	Trigger compliance runs.
Deployment	View deployments (workloads) in secured clusters.	N/A
DeploymentExtension	View the following items: ● Process baselines ● Process activity in deployments ● Risk results	Modify the following items: ● Process baselines (add or remove processes)
Detection	Check build-time policies against images or deployment YAML.	N/A
Image	View images, their components, and their vulnerabilities.	N/A
Integration	View integrations and their configuration, including backup, registry, image signature, notification systems, and API tokens.	Add, modify, and delete integrations and their configurations, and API tokens.
K8sRole	View roles for Kubernetes RBAC in secured clusters.	N/A
K8sRoleBinding	View role bindings for Kubernetes RBAC in secured clusters.	N/A
K8sSubject	View users and groups for Kubernetes RBAC in secured clusters.	N/A
Namespace	View existing Kubernetes namespaces in secured clusters.	N/A
NetworkGraph	View active and allowed network connections in secured clusters.	N/A
NetworkPolicy	View existing network policies in secured clusters and simulate changes.	Apply network policy changes in secured clusters.

Resource	Read permission	Write permission
Node	View existing Kubernetes nodes in secured clusters.	N/A
WorkflowAdministration	View all resource collections.	Add, modify, or delete resource collections.
Role	View existing Red Hat Advanced Cluster Security for Kubernetes RBAC roles and their permissions.	Add, modify, or delete roles and their permissions.
Secret	View metadata about secrets in secured clusters.	N/A
ServiceAccount	List Kubernetes service accounts in secured clusters.	N/A
VulnerabilityManagementApprovals	View all pending deferral or false positive requests for vulnerabilities.	Approve or deny any pending deferral or false positive requests and move any previously approved requests back to observed.
VulnerabilityManagementRequests	View all pending deferral or false positive requests for vulnerabilities.	Request a deferral on a vulnerability, mark it as a false positive, or move a pending or previously approved request made by the same user back to observed.
WatchedImage	View undeployed and monitored watched images.	Configure watched images.
WorkflowAdministration	View all resource collections.	Create, modify, or delete resource collections.

18.1.5. Declarative configuration for authentication and authorization resources

You can use declarative configuration for authentication and authorization resources such as authentication providers, roles, permission sets, and access scopes. For instructions on how to use declarative configuration, see "Using declarative configuration" in the "Additional resources" section.

Additional resources

- Using declarative configuration

18.2. ENABLING PKI AUTHENTICATION

If you use an enterprise certificate authority (CA) for authentication, you can configure Red Hat Advanced Cluster Security for Kubernetes (RHACS) to authenticate users by using their personal certificates.

After you configure PKI authentication, users and API clients can log in using their personal certificates. Users without certificates can still use other authentication options, including API tokens, the local administrator password, or other authentication providers. PKI authentication is available on the same port number as the Web UI, gRPC, and REST APIs.

When you configure PKI authentication, by default, Red Hat Advanced Cluster Security for Kubernetes uses the same port for PKI, web UI, gRPC, other single sign-on (SSO) providers, and REST APIs. You can also configure a separate port for PKI authentication by using a YAML configuration file to configure and expose endpoints.

18.2.1. Configuring PKI authentication by using the RHACS portal

You can configure Public Key Infrastructure (PKI) authentication by using the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Access Control**.
2. Click **Create Auth Provider** and select **User Certificates** from the drop-down list.
3. In the **Name** field, specify a name for this authentication provider.
4. In the **CA certificate(s) (PEM)** field, paste your root CA certificate in PEM format.
5. Assign a **Minimum access role** for users who access RHACS using PKI authentication. A user must have the permissions granted to this role or a role with higher permissions to log in to RHACS.

   TIP

   For security, Red Hat recommends first setting the **Minimum access role** to **None** while you complete setup. Later, you can return to the **Access Control** page to set up more tailored access rules based on user metadata from your identity provider.

6. To add access rules for users and groups accessing RHACS, click **Add new rule** in the **Rules** section. For example, to give the **Admin** role to a user called **administrator**, you can use the following key-value pairs to create access rules:

Key	Value
Name	administrator
Role	Admin

7. Click **Save**.

18.2.2. Configuring PKI authentication by using the roxctl CLI

You can configure PKI authentication by using the **roxctl** CLI.

Procedure

- Run the following command:

```
$ roxctl -e <hostname>:<port_number> central userpki create -c <ca_certificate_file> -r
<default_role_name> <provider_name>
```

18.2.3. Updating authentication keys and certificates

You can update your authentication keys and certificates by using the RHACS portal.

Procedure

1. Create a new authentication provider.
2. Copy the role mappings from your old authentication provider to the new authentication provider.
3. Rename or delete the old authentication provider with the old root CA key.

18.2.4. Logging in by using a client certificate

After you configure PKI authentication, users see a certificate prompt in the RHACS portal login page. The prompt only shows up if a client certificate trusted by the configured root CA is installed on the user's system.

Use the procedure described in this section to log in by using a client certificate.

Procedure

1. Open the RHACS portal.
2. Select a certificate in the browser prompt.
3. On the login page, select the authentication provider name option to log in with a certificate. If you do not want to log in by using the certificate, you can also log in by using the administrator password or another login method.

NOTE

Once you use a client certificate to log into the RHACS portal, you cannot log in with a different certificate unless you restart your browser.

18.3. UNDERSTANDING AUTHENTICATION PROVIDERS

An authentication provider connects to a third-party source of user identity (for example, an identity provider or IDP), gets the user identity, issues a token based on that identity, and returns the token to Red Hat Advanced Cluster Security for Kubernetes (RHACS). This token allows RHACS to authorize the user. RHACS uses the token within the user interface and API calls.

After installing RHACS, you must set up your IDP to authorize users.

NOTE

If you are using OpenID Connect (OIDC) as your IDP, RHACS relies on mapping rules that examine the values of specific claims like **groups**, **email**, **userid** and **name** from either the user ID token or the **UserInfo** endpoint response to authorize the users. If these details are absent, the mapping cannot succeed and the user does not get access to the required resources. Therefore, you need to ensure that the claims required to authorize users from your IDP, for example, **groups**, are included in the authentication response of your IDP to enable successful mapping.

Additional resources

- Configuring Okta Identity Cloud as a SAML 2.0 identity provider
- Configuring Google Workspace as an OIDC identity provider
- Configuring OpenShift Container Platform OAuth server as an identity provider
- Connecting Azure AD to RHACS using SSO configuration

18.3.1. Claim mappings

A claim is the data an identity provider includes about a user inside the token they issue.

Using claim mappings, you can specify if RHACS should customize the claim attribute it receives from an IDP to another attribute in the RHACS-issued token. If you do not use the claim mapping, RHACS does not include the claim attribute in the RHACS-issued token.

For example, you can map from **roles** in the user identity to **groups** in the RHACS-issued token using claim mapping.

RHACS uses different default claim mappings for every authentication provider.

18.3.1.1. OIDC default claim mappings

The following list provides the default OIDC claim mappings:

- **sub** to **userid**
- **name** to **name**
- **email** to **email**
- **groups** to **groups**

18.3.1.2. Auth0 default claim mappings

The **Auth0** default claim mappings are the same as the OIDC default claim mappings.

18.3.1.3. SAML 2.0 default claim mappings

The following list applies to SAML 2.0 default claim mappings:

- **Subject.NameID** is mapped to **userid**
- every SAML **AttributeStatement.Attribute** from the response gets mapped to its name

18.3.1.4. Google IAP default claim mappings

The following list provides the Google IAP default claim mappings:

- **sub** to **userid**
- **email** to **email**
- **hd** to **hd**
- **google.access_levels** to **access_levels**

18.3.1.5. User certificates default claim mappings

User certificates differ from all other authentication providers because instead of communicating with a third-party IDP, they get user information from certificates used by the user.

The default claim mappings for user certificates include:

- **CertFingerprint** to **userid**
- **Subject → Common Name** to **name**
- **EmailAddresses** to **email**
- **Subject → Organizational Unit** to **groups**

18.3.1.6. OpenShift Auth default claim mappings

The following list provides the OpenShift Auth default claim mappings:

- **groups** to **groups**
- **uid** to **userid**
- **name** to **name**

18.3.2. Rules

To authorize users, RHACS relies on mapping rules that examine the values of specific claims such as **groups**, **email**, **userid**, and **name** from the user identity. Rules allow mapping of users who have attributes with a specific value to a specific role. As an example, a rule could include the following:`key` is **email**, **value** is **john@redhat.com**, **role** is **Admin**.

If the claim is missing, the mapping cannot succeed, and the user does not get access to the required resources. Therefore, to enable successful mapping, you must ensure that the authentication response from your IDP includes the required claims to authorize users, for example, **groups**.

18.3.3. Minimum access role

RHACS assigns a minimum access role to every caller with a RHACS token issued by a particular authentication provider. The minimum access role is set to **None** by default.

For example, suppose there is an authentication provider with the minimum access role of **Analyst**. In that case, all users who log in using this provider will have the **Analyst** role assigned to them.

18.3.4. Required attributes

Required attributes can restrict issuing of the RHACS token based on whether a user identity has an attribute with a specific value.

For example, you can configure RHACS only to issue a token when the attribute with key **is_internal** has the attribute value **true**. Users with the attribute **is_internal** set to **false** or not set do not get a token.

18.4. CONFIGURING IDENTITY PROVIDERS

18.4.1. Configuring Okta Identity Cloud as a SAML 2.0 identity provider

You can use Okta as a single sign-on (SSO) provider for Red Hat Advanced Cluster Security for Kubernetes (RHACS).

18.4.1.1. Creating an Okta app

Before you can use Okta as a SAML 2.0 identity provider for Red Hat Advanced Cluster Security for Kubernetes, you must create an Okta app.

WARNING

Okta's **Developer Console** does not support the creation of custom SAML 2.0 applications. If you are using the **Developer Console**, you must first switch to the **Admin Console** (**Classic UI**). To switch, click **Developer Console** in the top left of the page and select **Classic UI**.

Prerequisites

- You must have an account with administrative privileges for the Okta portal.

Procedure

1. On the Okta portal, select **Applications** from the menu bar.
2. Click **Add Application** and then select **Create New App**.
3. In the **Create a New Application Integration** dialog box, leave **Web** as the platform and select **SAML 2.0** as the protocol that you want to sign in users.
4. Click **Create**.
5. On the **General Settings** page, enter a name for the app in the **App name** field.
6. Click **Next**.
7. On the **SAML Settings** page, set values for the following fields:
    a. **Single sign on URL**

- Specify it as **https://<RHACS_portal_hostname>/sso/providers/saml/acs**.
- Leave the **Use this for Recipient URL and Destination URL**option checked.
- If your RHACS portal is accessible at different URLs, you can add them here by checking the **Allow this app to request other SSO URLs**option and add the alternative URLs using the specified format.

b. **Audience URI (SP Entity ID)**

- Set the value to **RHACS** or another value of your choice.
- Remember the value you choose; you will need this value when you configure Red Hat Advanced Cluster Security for Kubernetes.

c. **Attribute Statements**

- You must add at least one attribute statement.
- Red Hat recommends using the email attribute:
  - **Name:** email
  - **Format:** Unspecified
  - **Value:** user.email

8. Verify that you have configured at least one **Attribute Statement** before continuing.
9. Click **Next**.
10. On the **Feedback** page, select an option that applies to you.
11. Select an appropriate **App type**.
12. Click **Finish**.

After the configuration is complete, you are redirected to the **Sign On** settings page for the new app. A yellow box contains links to the information that you need to configure Red Hat Advanced Cluster Security for Kubernetes.

After you have created the app, assign Okta users to this application. Go to the **Assignments** tab, and assign the set of individual users or groups that can access Red Hat Advanced Cluster Security for Kubernetes. For example, assign the group **Everyone** to allow all users in the organization to access Red Hat Advanced Cluster Security for Kubernetes.

18.4.1.2. Configuring a SAML 2.0 identity provider

Use the instructions in this section to integrate a Security Assertion Markup Language (SAML) 2.0 identity provider with Red Hat Advanced Cluster Security for Kubernetes (RHACS).

Prerequisites

- You must have permissions to configure identity providers in RHACS.
- For Okta identity providers, you must have an Okta app configured for RHACS.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Access Control**.
2. Click **Create auth provider** and select **SAML 2.0** from the drop-down list.
3. In the **Name** field, enter a name to identify this authentication provider; for example, **Okta** or **Google**. The integration name is shown on the login page to help users select the correct sign-in option.
4. In the **ServiceProvider issuer** field, enter the value that you are using as the **Audience URI** or **SP Entity ID** in Okta, or a similar value in other providers.
5. Select the type of **Configuration**:
   - **Option 1: Dynamic Configuration**: If you select this option, enter the **IdP Metadata URL**, or the URL of **Identity Provider metadata** available from your identity provider console. The configuration values are acquired from the URL.
   - **Option 2: Static Configuration**: Copy the required static fields from the **View Setup Instructions** link in the Okta console, or a similar location for other providers:
     - **IdP Issuer**
     - **IdP SSO URL**
     - **Name/ID Format**
     - **IdP Certificate(s) (PEM)**
6. Assign a **Minimum access role** for users who access RHACS using SAML.

   TIP

   Set the **Minimum access role** to **Admin** while you complete setup. Later, you can return to the **Access Control** page to set up more tailored access rules based on user metadata from your identity provider.
7. Click **Save**.

IMPORTANT

If your SAML identity provider's authentication response meets the following criteria:

- Includes a **NotValidAfter** assertion: The user session remains valid until the time specified in the **NotValidAfter** field has elapsed. After the user session expires, users must reauthenticate.
- Does not include a **NotValidAfter** assertion: The user session remains valid for 30 days, and then users must reauthenticate.

Verification

1. In the RHACS portal, go to **Platform Configuration** → **Access Control**.
2. Select the **Auth Providers** tab.

3. Click the authentication provider for which you want to verify the configuration.
4. Select **Test login** from the **Auth Provider** section header. The **Test login** page opens in a new browser tab.
5. Sign in with your credentials.
   - If you logged in successfully, RHACS shows the **User ID** and **User Attributes** that the identity provider sent for the credentials that you used to log in to the system.
   - If your login attempt failed, RHACS shows a message describing why the identity provider's response could not be processed.
6. Close the **Test login** browser tab.

NOTE

Even if the response indicates successful authentication, you might need to create additional access rules based on the user metadata from your identity provider.

18.4.2. Configuring Google Workspace as an OIDC identity provider

You can use Google Workspace as a single sign-on (SSO) provider for Red Hat Advanced Cluster Security for Kubernetes.

18.4.2.1. Setting up OAuth 2.0 credentials for your GCP project

To configure Google Workspace as an identity provider for Red Hat Advanced Cluster Security for Kubernetes, you must first configure OAuth 2.0 credentials for your GCP project.

Prerequisites

- You must have administrator-level access to your organization's Google Workspace account to create a new project, or permissions to create and configure OAuth 2.0 credentials for an existing project. Red Hat recommends that you create a new project for managing access to Red Hat Advanced Cluster Security for Kubernetes.

Procedure

1. Create a new Google Cloud Platform (GCP) project, see the Google documentation topic creating and managing projects.
2. After you have created the project, open the **Credentials** page in the Google API Console.
3. Verify the project name listed in the upper left corner near the logo to make sure that you are using the correct project.
4. To create new credentials, go to **Create Credentials** → **OAuth client ID**.
5. Choose **Web application** as the **Application type**.
6. In the **Name** box, enter a name for the application, for example, **RHACS**.
7. In the **Authorized redirect URIs** box, enter **https://<stackrox_hostname>:<port_number>/sso/providers/oidc/callback**.

- replace **<stackrox_hostname>** with the hostname on which you expose your Central instance.
- replace **<port_number>** with the port number on which you expose Central. If you are using the standard HTTPS port **443**, you can omit the port number.

8. Click **Create**. This creates an application and credentials and redirects you back to the credentials page.
9. An information box opens, showing details about the newly created application. Close the information box.
10. Copy and save the **Client ID** that ends with **.apps.googleusercontent.com**. You can check this client ID by using the Google API Console.
11. Select **OAuth consent screen** from the navigation menu on the left.

NOTE

The OAuth consent screen configuration is valid for the entire GCP project, and not only to the application you created in the previous steps. If you already have an OAuth consent screen configured in this project and want to apply different settings for Red Hat Advanced Cluster Security for Kubernetes login, create a new GCP project.

12. On the OAuth consent screen page:
    a. Choose the **Application type** as **Internal**. If you select **Public**, anyone with a Google account can sign in.
    b. Enter a descriptive **Application name**. This name is shown to users on the consent screen when they sign in. For example, use **RHACS** or **<organization_name> SSO for Red Hat Advanced Cluster Security for Kubernetes**.
    c. Verify that the **Scopes for Google APIs** only lists **email**, **profile**, and **openid** scopes. Only these scopes are required for single sign-on. If you grant additional scopes it increases the risk of exposing sensitive data.

18.4.2.2. Specifying a client secret

Red Hat Advanced Cluster Security for Kubernetes version 3.0.39 and newer supports the OAuth 2.0 Authorization Code Grant authentication flow when you specify a client secret. When you use this authentication flow, Red Hat Advanced Cluster Security for Kubernetes uses a refresh token to keep users logged in beyond the token expiration time configured in your OIDC identity provider.

When users log out, Red Hat Advanced Cluster Security for Kubernetes deletes the refresh token from the client-side. Additionally, if your identity provider API supports refresh token revocation, Red Hat Advanced Cluster Security for Kubernetes also sends a request to your identity provider to revoke the refresh token.

You can specify a client secret when you configure Red Hat Advanced Cluster Security for Kubernetes to integrate with an OIDC identity provider.

NOTE

- You cannot use a **Client Secret** with the *Fragment* **Callback mode**.
- You cannot edit configurations for existing authentication providers.
- You must create a new OIDC integration in Red Hat Advanced Cluster Security for Kubernetes if you want to use a **Client Secret**.

Red Hat recommends using a client secret when connecting Red Hat Advanced Cluster Security for Kubernetes with an OIDC identity provider. If you do not want to use a **Client Secret**, you must select the **Do not use Client Secret (not recommended)**option.

18.4.2.3. Configuring an OIDC identity provider

You can configure Red Hat Advanced Cluster Security for Kubernetes (RHACS) to use your OpenID Connect (OIDC) identity provider.

Prerequisites

- You must have already configured an application in your identity provider, such as Google Workspace.
- You must have permissions to configure identity providers in RHACS.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Access Control**.
2. Click **Create auth provider** and select **OpenID Connect** from the drop-down list.
3. Enter information in the following fields:
   - **Name**: A name to identify your authentication provider; for example, **Google Workspace**. The integration name is shown on the login page to help users select the correct sign-in option.
   - **Callback mode**: Select **Auto-select (recommended)**, which is the default value, unless the identity provider requires another mode.

     NOTE

     Fragment mode is designed around the limitations of Single Page Applications (SPAs). Red Hat only supports the **Fragment** mode for early integrations and does not recommended using it for later integrations.

   - **Issuer**: The root URL of your identity provider; for example, **https://accounts.google.com** for Google Workspace. See your identity provider documentation for more information.

NOTE

If you are using RHACS version 3.0.49 and later, for **Issuer** you can perform these actions:

- Prefix your root URL with **https+insecure://** to skip TLS validation. This configuration is insecure and Red Hat does not recommended it. Only use it for testing purposes.
- Specify query strings; for example, **?key1=value1&key2=value2** along with the root URL. RHACS appends the value of **Issuer** as you entered it to the authorization endpoint. You can use it to customize your provider's login screen. For example, you can optimize the Google Workspace login screen to a specific hosted domain by using the **hd** parameter, or preselect an authentication method in **PingFederate** by using the **pfidpadapterid** parameter.

- **Client ID**: The OIDC Client ID for your configured project.
- **Client Secret**: Enter the client secret provided by your identity provider (IdP). If you are not using a client secret, which is not recommended, select **Do not use Client Secret**.

4. Assign a **Minimum access role** for users who access RHACS using the selected identity provider.

   TIP

   Set the **Minimum access role** to **Admin** while you complete setup. Later, you can return to the **Access Control** page to set up more tailored access rules based on user metadata from your identity provider.

5. To add access rules for users and groups accessing RHACS, click **Add new rule** in the **Rules** section. For example, to give the **Admin** role to a user called **administrator**, you can use the following key-value pairs to create access rules:

Key	**Value**
Name	administrator
Role	Admin

6. Click **Save**.

Verification

1. In the RHACS portal, go to **Platform Configuration → Access Control**.
2. Select the **Auth providers** tab.
3. Select the authentication provider for which you want to verify the configuration.
4. Select **Test login** from the **Auth Provider** section header. The **Test login** page opens in a new browser tab.

5. Log in with your credentials.
   - If you logged in successfully, RHACS shows the **User ID** and **User Attributes** that the identity provider sent for the credentials that you used to log in to the system.
   - If your login attempt failed, RHACS shows a message describing why the identity provider's response could not be processed.
6. Close the **Test Login** browser tab.

18.4.3. Configuring OpenShift Container Platform OAuth server as an identity provider

OpenShift Container Platform includes a built-in OAuth server that you can use as an authentication provider for Red Hat Advanced Cluster Security for Kubernetes (RHACS).

18.4.3.1. Configuring OpenShift Container Platform OAuth server as an identity provider

To integrate the built-in OpenShift Container Platform OAuth server as an identity provider for RHACS, use the instructions in this section.

Prerequisites

- You must have the **AuthProvider** permission to configure identity providers in RHACS.
- You must have already configured users and groups in OpenShift Container Platform OAuth server through an identity provider. For information about the identity provider requirements, see Understanding identity provider configuration .

NOTE

The following procedure configures only a single main route named **central** for the OpenShift Container Platform OAuth server.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Access Control**.
2. Click **Create auth provider** and select **OpenShift Auth** from the drop-down list.
3. Enter a name for the authentication provider in the **Name** field.
4. Assign a **Minimum access role** for users that access RHACS using the selected identity provider. A user must have the permissions granted to this role or a role with higher permissions to log in to RHACS.

   TIP

   For security, Red Hat recommends first setting the **Minimum access role** to **None** while you complete setup. Later, you can return to the **Access Control** page to set up more tailored access rules based on user metadata from your identity provider.

5. Optional: To add access rules for users and groups accessing RHACS, click **Add new rule** in the **Rules** section, then enter the rule information and click **Save**. You will need attributes for the user or group so that you can configure access.

   TIP

   Group mappings are more robust because groups are usually associated with teams or permissions sets and require modification less often than users.

   To get user information in OpenShift Container Platform, you can use one of the following methods:

   - Click **User Management** → **Users** → **<username>** → **YAML**.
   - Access the **k8s/cluster/user.openshift.io~v1~User/<username>/yaml** file and note the values for **name**, **uid** (**userid** in RHACS), and **groups**.
   - Use the OpenShift Container Platform API as described in the *OpenShift Container Platform API reference*.

   The following configuration example describes how to configure rules for an **Admin** role with the following attributes:

   - **name**: **administrator**
   - **groups**: **["system:authenticated", "system:authenticated:oauth", "myAdministratorsGroup"]**
   - **uid**: **12345-00aa-1234-123b-123fcdef1234**

   You can add a rule for this administrator role using one of the following steps:

   - To configure a rule for a name, select **name** from the **Key** drop-down list, enter **administrator** in the **Value** field, then select **Administrator** under **Role**.
   - To configure a rule for a group, select **groups** from the **Key** drop-down list, enter **myAdministratorsGroup** in the **Value** field, then select **Admin** under **Role**.
   - To configure a rule for a user name, select **userid** from the **Key** drop-down list, enter **12345-00aa-1234-123b-123fcdef1234** in the **Value** field, then select **Admin** under **Role**.

IMPORTANT

- If you use a custom TLS certificate for OpenShift Container Platform OAuth server, you must add the root certificate of the CA to Red Hat Advanced Cluster Security for Kubernetes as a trusted root CA. Otherwise, Central cannot connect to the OpenShift Container Platform OAuth server.
- To enable the OpenShift Container Platform OAuth server integration when installing Red Hat Advanced Cluster Security for Kubernetes using the **roxctl** CLI, set the **ROX_ENABLE_OPENSHIFT_AUTH** environment variable to **true** in Central:

  ```
  $ oc -n stackrox set env deploy/central
  ROX_ENABLE_OPENSHIFT_AUTH=true
  ```
- For access rules, the OpenShift Container Platform OAuth server does not return the key **Email**.

Additional resources

- Configuring an LDAP identity provider
- Adding trusted certificate authorities

18.4.3.2. Creating additional routes for OpenShift Container Platform OAuth server

When you configure OpenShift Container Platform OAuth server as an identity provider by using Red Hat Advanced Cluster Security for Kubernetes portal, RHACS configures only a single route for the OAuth server. However, you can create additional routes by specifying them as annotations in the Central custom resource.

Prerequisites

- You must have configured Service accounts as OAuth clients for your OpenShift Container Platform OAuth server.

Procedure

- If you installed RHACS using the RHACS Operator:

  1. Create a **CENTRAL_ADDITIONAL_ROUTES** environment variable that contains a patch for the Central custom resource:

     ```
     $ CENTRAL_ADDITIONAL_ROUTES='
     spec:
       central:
         exposure:
           loadBalancer:
             enabled: false
             port: 443
           nodePort:
             enabled: false
           route:
             enabled: true
         persistence:
     ```

```
      persistentVolumeClaim:
        claimName: stackrox-db
  customize:
    annotations:
      serviceaccounts.openshift.io/oauth-redirecturi.main: sso/providers/openshift/callback
1
      serviceaccounts.openshift.io/oauth-redirectreference.main: "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"central\"}}" 2
      serviceaccounts.openshift.io/oauth-redirecturi.second:
sso/providers/openshift/callback 3
      serviceaccounts.openshift.io/oauth-redirectreference.second: "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"second-central\"}}" 4
'
```

1 The redirect URI for setting the main route.

2 The redirect URI reference for the main route.

3 The redirect for setting the second route.

4 The redirect reference for the second route.

2. Apply the **CENTRAL_ADDITIONAL_ROUTES** patch to the Central custom resource:

```
$ oc patch centrals.platform.stackrox.io \
  -n <namespace> \ 1
  <custom-resource> \ 2
  --patch "$CENTRAL_ADDITIONAL_ROUTES" \
  --type=merge
```

1 Replace **<namespace>** with the name of the project that contains the Central custom resource.

2 Replace **<custom-resource>** with the name of the Central custom resource.

- Or, if you installed RHACS using Helm:

1. Add the following annotations to your **values-public.yaml** file:

```
customize:
  central:
    annotations:
      serviceaccounts.openshift.io/oauth-redirecturi.main: sso/providers/openshift/callback
1
      serviceaccounts.openshift.io/oauth-redirectreference.main: "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"central\"}}" 2
      serviceaccounts.openshift.io/oauth-redirecturi.second:
sso/providers/openshift/callback 3
```

```
    serviceaccounts.openshift.io/oauth-redirectreference.second: "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"second-central\"}}" 4
```

1. The redirect for setting the main route.
2. The redirect reference for the main route.
3. The redirect for setting the second route.
4. The redirect reference for the second route.

2. Apply the custom annotations to the Central custom resource by using **helm upgrade**:

```
$ helm upgrade -n stackrox \
  stackrox-central-services rhacs/central-services \
  -f <path_to_values_public.yaml> 1
```

1. Specify the path of the **values-public.yaml** configuration file using the **-f** option.

Additional resources

- Service accounts as OAuth clients
- Redirect URIs for service accounts as OAuth clients

18.4.4. Connecting Azure AD to RHACS using SSO configuration

To connect an Azure Active Directory (AD) to RHACS using Sign-On (SSO) configuration, you need to add specific claims (for example, **group** claim to tokens) and assign users, groups, or both to the enterprise application.

18.4.4.1. Adding group claims to tokens for SAML applications using SSO configuration

Configure the application registration in Azure AD to include **group** claims in tokens. For instructions, see Add group claims to tokens for SAML applications using SSO configuration .

IMPORTANT

Verify that you are using the latest version of Azure AD. For more information on how to upgrade Azure AD to the latest version, see Azure AD Connect: Upgrade from a previous version to the latest.

18.5. CONFIGURING SHORT-LIVED ACCESS

Red Hat Advanced Cluster Security for Kubernetes (RHACS) provides the ability to configure short-lived access to the user interface and API calls.

You can configure this by exchanging OpenID Connect (OIDC) identity tokens for a RHACS-issued token.

We recommend this especially for Continuous Integration (CI) usage, where short-lived access is preferable over long-lived API tokens.

The following steps outline the high-level workflow on how to configure short-lived access to the user interface and API calls:

1. Configuring RHACS to trust OIDC identity token issuers for exchanging short-lived RHACS-issued tokens.
2. Exchanging an OIDC identity token for a short-lived RHACS-issued token by calling the API.

NOTE

- To prevent privilege escalation, when you create a new token, your role's permissions limit the permission 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 to use, 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 other automation. Also, use the **roxctl central login** command for human-to-machine communication, such as **roxctl** CLI or API access.

Additional resources

- Using an authentication provider to authenticate with roxctl
- Configuring API tokens

18.5.1. Configure short-lived access for an OIDC identity token issuer

Start configuring short-lived access for an OpenID Connect (OIDC) identity token issuer.

Procedure

1. In the RHACS portal, go to **Platform Configuration** → **Integrations**.
2. Scroll to the **Authentication Tokens** category, and then click **Machine access configuration**.
3. Click **Create configuration**.
4. Select the **configuration type**, choosing one of the following:
   - **Generic** if you use an arbitrary OIDC identity token issuer.
   - **GitHub Actions** if you plan to access RHACS from GitHub Actions.
5. Enter the OIDC identity token issuer.
6. Enter the **token lifetime** for tokens issued by the configuration.

> **NOTE**
>
> The format for the **token lifetime** is **XhYmZs** and you cannot set it for longer than 24 hours.

7. Add rules to the configuration:
    - The **Key** is the OIDC token's claim to use.
    - The **Value** is the expected OIDC token claim value.
    - The **Role** is the role to assign to the token if the OIDC token claim and value exist.

    > **NOTE**
    >
    > Rules are similar to Authentication Provider rules to assign roles based on claim values.
    >
    > As a general rule, Red Hat recommends to use unique, immutable claims within Rules. The general recommendation is to use the **sub** claim within the OIDC identity token. For more information about OIDC token claims, see the list of standard OIDC claims.

8. Click **Save**.

18.5.2. Exchanging an identity token

Prerequisites

- You have a valid OpenID Connect (OIDC) token.
- You added a **Machine access configuration** for the RHACS instance you want to access.

Procedure

1. Prepare the POST request's JSON data:

```
{
    "idToken": "<id_token>"
}
```

2. Send a POST request to the API **/v1/auth/m2m/exchange**.
3. Wait for the API response:

```
{
    "accessToken": "<access_token>"
}
```

4. Use the returned access token to access the RHACS instance.

NOTE

If you are using **GitHub Actions**, you can use the stackrox/central-login GitHub Action .

CHAPTER 19. USING THE SYSTEM HEALTH DASHBOARD

The Red Hat Advanced Cluster Security for Kubernetes system health dashboard provides a single interface for viewing health related information about Red Hat Advanced Cluster Security for Kubernetes components.

NOTE

The system health dashboard is only available on Red Hat Advanced Cluster Security for Kubernetes 3.0.53 and newer.

19.1. SYSTEM HEALTH DASHBOARD DETAILS

To access the health dashboard:

- In the RHACS portal, go to **Platform Configuration → System Health**.

The health dashboard organizes information in the following groups:

- **Cluster Health** - Shows the overall state of Red Hat Advanced Cluster Security for Kubernetes cluster.
- **Vulnerability Definitions** - Shows the last update time of vulnerability definitions.
- **Image Integrations** - Shows the health of all registries that you have integrated.
- **Notifier Integrations** - Shows the health of any notifiers (Slack, email, Jira, or other similar integrations) that you have integrated.
- **Backup Integrations** - Shows the health of any backup providers that you have integrated.

The dashboard lists the following states for different components:

- **Healthy** - The component is functional.
- **Degraded** - The component is partially unhealthy. This state means the cluster is functional, but some components are unhealthy and require attention.
- **Unhealthy** - This component is not healthy and requires immediate attention.
- **Uninitialized** - The component has not yet reported back to Central to have its health assessed. An uninitialized state may sometimes require attention, but often components report back the health status after a few minutes or when the integration is used.

Cluster health section

The **Cluster Overview** shows information about your Red Hat Advanced Cluster Security for Kubernetes cluster health. It reports the health information about the following:

- **Collector Status** - It shows whether the Collector pod that Red Hat Advanced Cluster Security for Kubernetes uses is reporting healthy.
- **Sensor Status** - It shows whether the Sensor pod that Red Hat Advanced Cluster Security for Kubernetes uses is reporting healthy.
- **Sensor Upgrade** - It shows whether the Sensor is running the correct version when compared with Central.

- **Credential Expiration** - It shows if the credentials for Red Hat Advanced Cluster Security for Kubernetes are nearing expiration.

NOTE

Clusters in the **Uninitialized** state are not reported in the number of clusters secured by Red Hat Advanced Cluster Security for Kubernetes until they check in.

Vulnerabilities definition section

The **Vulnerabilities Definition** section shows the last time vulnerability definitions were updated and if the definitions are up to date.

Integrations section

There are 3 integration sections **Image Integrations**, **Notifier Integrations**, and **Backup Integrations**. Similar to the **Cluster Health** section, these sections list the number of unhealthy integrations if they exist. Otherwise, all integrations report as healthy.

NOTE

The **Integrations** section lists the healthy integrations as **0** if any of the following conditions are met:

- You have not integrated Red Hat Advanced Cluster Security for Kubernetes with any third-party tools.
- You have integrated with some tools, but disabled the integrations, or have not set up any policy violations.

19.2. VIEWING PRODUCT USAGE DATA

RHACS provides product usage data for the number of secured Kubernetes nodes and CPU units for secured clusters based on metrics collected from RHACS sensors. This information can be useful to estimate RHACS consumption data for reporting.

For more information on how CPU units are defined in Kubernetes, see CPU resource units.

NOTE

OpenShift Container Platform provides its own usage reports; this information is intended for use with self-managed Kubernetes systems.

RHACS provides the following usage data in the web portal and API:

- Currently secured CPU units: The number of Kubernetes CPU units used by your RHACS secured clusters, as of the latest metrics collection.
- Currently secured node count: The number of Kubernetes nodes secured by RHACS, as of the latest metrics collection.
- Maximum secured CPU units: The maximum number of CPU units used by your RHACS secured clusters, as measured hourly and aggregated for the time period defined by the **Start date** and **End date**.

- Maximum secured node count: The maximum number of Kubernetes nodes secured by RHACS, as measured hourly and aggregated for the time period defined by the **Start date** and **End date**.
- CPU units observation date: The date on which the maximum secured CPU units data was collected.
- Node count observation date: The date on which the maximum secured node count data was collected.

The sensors collect data every 5 minutes, so there can be a short delay in displaying the current data. To view historical data, you must configure the **Start date** and **End date** and download the data file. The date range is inclusive and depends on your time zone.

The presented maximum values are computed based on hourly maximums for the requested period. The hourly maximums are available for download in CSV format.

NOTE

The data shown is not sent to Red Hat or displayed as Prometheus metrics.

Procedure

1. In the RHACS portal, go to **Platform Configuration → System Health**.
2. Click **Show product usage**.
3. In the **Start date** and **End date** fields, choose the dates for which you want to display data. This range is inclusive and depends on your time zone.
4. Optional: To download the detailed data, click **Download CSV**.

You can also obtain this data by using the **ProductUsageService** API object. For more information, go to **Help → API reference** in the RHACS portal.

19.3. GENERATING A DIAGNOSTIC BUNDLE BY USING THE RHACS PORTAL

You can generate a diagnostic bundle by using the system health dashboard in the RHACS portal.

Prerequisites

- To generate a diagnostic bundle, you need **read** permission for the **DebugLogs** resource.

Procedure

1. In the RHACS portal, select **Platform Configuration → System Health**.
2. On the **System Health** view header, click **Generate Diagnostic Bundle**.
3. For the **Filter by clusters** drop-down menu, select the clusters for which you want to generate the diagnostic data.
4. For **Filter by starting time**, specify the date and time (in UTC format) from which you want to include the diagnostic data.

5. Click **Download Diagnostic Bundle**.

19.3.1. Additional resources

- Generating a diagnostic bundle

CHAPTER 20. USING THE ADMINISTRATION EVENTS PAGE

The Red Hat Advanced Cluster Security for Kubernetes (RHACS) administration events page provides a single interface to view administration event information that helps you understand and interpret important event details. This includes error descriptions, resource types, timestamps, and event categorizations. This information helps you efficiently manage and troubleshoot events in your RHACS environment.

20.1. VIEWING THE ADMINISTRATION EVENTS PAGE

By viewing the administration events page, you can access various event logs in different domains.

Procedure

- In the RHACS platform, go to **Platform Configuration → Administration Events**.

20.2. UNDERSTANDING THE ADMINISTRATION EVENTS PAGE

The administration events page organizes information in the following groups:

- **Domain**: Categorizes events based on the specific area or domain within RHACS in which they occurred. This classification helps organize and understand the context of events. Domains include **Authentication**, **General**, **Image Scanning** , and **Integrations**.
- **Resource type**: Classifies events based on the resource or component type involved. Resource types include **API Token**, **Cluster**, **Image**, **Node**, and **Notifier**.
- **Level**: Indicates the severity or importance of an event. Levels include **Error**, **Warning**, **Success**, **Info** and **Unknown**.
- **Event last occurred at**: Provides information about the timestamp and date when a particular event occurred. It helps track the timing of events, which is essential for diagnosing issues and understanding the sequence of actions or incidents.
- **Count**: Indicates the number of times a particular event occurred. This number is useful in assessing the frequency of an issue. An event that has occurred multiple times indicates a persistent issue that you need to fix.

Each event also gives you an indication of what you need to do to fix the error.

20.3. VIEWING THE ADMINISTRATION EVENT DETAILS

By viewing the details of an administration event, you get more information about the events in that particular domain. This allows you to better understand the context and details of the events.

Procedure

- In the **Administration Events** page, click the domain to view its details.

20.4. UNDERSTANDING THE ADMINISTRATION EVENT DETAILS

The administration event provide log information that describes the error or event. They provide important information for understanding a specific issue or incident and includes information about the context of the event as well as the steps to take to fix the error.

The administration event page organizes information in the following groups:

- **Resource type**: Classifies events based on the resource or component type involved. Resource types include **API Token**, **Cluster**, **Image**, **Node**, and **Notifier**.
- **Resource name**: Specifies the name of the resource or component to which the event refers. It identifies the specific instance within the domain where the event occurred.
- **Event type**: Specifies the source of the event. Currently, Central generates log events that correspond to administration events created from log statements.
- **Event ID**: A unique identifier assigned to each event. It serves as a reference point and can be useful in tracking and managing events over time. Event IDs are unique alphanumeric characters that help identify and distinguish events.
- **Created at**: Indicates the timestamp and date when the event was originally created or recorded. It provides an indication of when the event was first detected or created.
- **Last occurred at**: Specifies the timestamp and date when the event last occurred. This tracks the timing of the event, which can be critical for diagnosing and fixing recurring issues.
- **Count**: Indicates the number of times a particular event occurred. This number is useful in assessing the frequency of an issue. An event that has occurred multiple times indicates a persistent issue you need to fix.

20.5. CONFIGURING ADMINISTRATION EVENTS RETENTION SETTINGS

By configuring the retention settings, you can determine when the administration events expire.

Procedure

1. In the RHACS portal, go to **Platform Configuration → System Configuration**. You can configure the following setting for administration events:
   - **Administration events retention days**: The number of days to retain your administration events. By specifying the number of days, you can control when these events expire. This is important for managing your administration events and ensuring that you retain the information for the desired duration.

     NOTE

     By default, the administration events are retained for 4 days. The retention period for these events is determined by the time of the last occurrence and not by the time of creation. This means that an event expires and is deleted only if the time of the last occurrence exceeds the specified retention period.
2. To change this value, click **Edit**, make your changes, and then click **Save**.