Red Hat
OpenShift

OpenShift Cluster Compliance with OpenSCAP

Mark Hooper
Senior Specialist Solution Architect
Chicago, IL

Red Hat

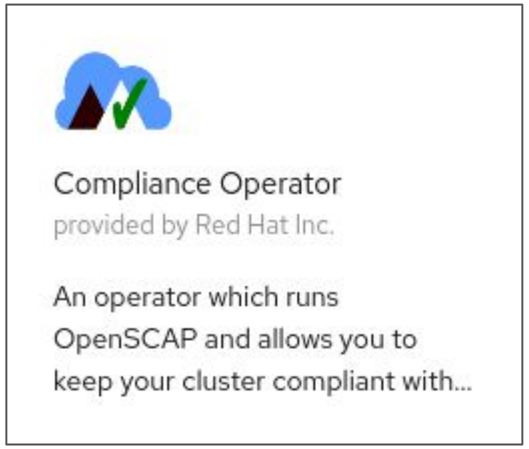
What is OpenSCAP

Security Content Automation Protocol (SCAP) is U.S. standard maintained by National Institute of Standards and Technology (NIST). The OpenSCAP project is a collection of open source tools for implementing and enforcing this standard, and has been awarded the SCAP 1.2 certification by NIST in 2014

Compliance Operator

The Compliance Operator lets OpenShift Container Platform administrators describe the desired compliance state of a cluster and provides them with an overview of gaps and ways to remediate them.

The Compliance Operator assesses compliance of both the Kubernetes API resources of OpenShift Container Platform, as well as the nodes running the cluster. The Compliance Operator uses OpenSCAP, a NIST-certified tool, to scan and enforce security policies provided by the content.


Compliance Operator
provided by Red Hat Inc.

An operator which runs OpenSCAP and allows you to keep your cluster compliant with...

Requirements

- Openshift 4.6+
- Persistent storage (and make sure the default storage class is set)

Compliance Operator Profiles

There are several profiles available as part of the Compliance Operator installation. These profiles represent different compliance benchmarks.

```
$ oc get -n openshift profiles.compliance

NAME                AGE
ocp4-cis            4h52m
ocp4-cis-node       4h52m
ocp4-e8             4h52m
ocp4-moderate       4h52m
ocp4-ncp            4h52m
rhcos4-e8           4h52m
rhcos4-moderate     4h52m
rhcos4-ncp          4h52m
```

Each profile has the product name that it applies to added as a prefix to the profile's name. `ocp4-e8` applies the Essential 8 benchmark to the OpenShift Container Platform product, while `rhcos4-e8` applies the Essential 8 benchmark to the Red Hat CoreOS product.

```
$ oc get -n openshift-compliance -oyaml profiles.compliance rhcos4-e8

apiVersion: compliance.openshift.io/v1alpha1
description: |-
  This profile contains configuration checks for Red Hat
  Enterprise Linux CoreOS that align to the Australian
  Cyber Security Centre (ACSC) Essential Eight.
  A copy of the Essential Eight in Linux Environments guide can
  be found at the ACSC website: ...
id: xccdf_org.ssgproject.content_profile_e8
kind: Profile
metadata:
  annotations:
    compliance.openshift.io/product: redhat_enterprise_linux_coreos_4
    compliance.openshift.io/product-type: Node
    creationTimestamp: "2020-09-07T11:42:51Z"
    generation: 1
  labels:
    compliance.openshift.io/profile-bundle: rhcos4
    name: rhcos4-e8
  namespace: openshift-compliance
rules:
- rhcos4-accounts-no-uid-except-zero
- rhcos4-audit-rules-dac-modification-chmod
- rhcos4-audit-rules-dac-modification-chown
- rhcos4-audit-rules-execution-chcon
- rhcos4-audit-rules-execution-restorecon
- rhcos4-audit-rules-execution-semanage
- rhcos4-audit-rules-execution-setfiles
- rhcos4-audit-rules-execution-setsebool
```

Compliance Rules

```
$ oc get -n openshift-compliance -oyaml rules.compliance rhcos4-audit-rules-login-events

apiVersion: compliance.openshift.io/v1alpha1
description: '<code>auditd</code><code>augenrules</code><code>.rules</code><code>/etc/audit/rules.d</code><pre>-w /var/log/tallylog -p wa -k logins -w
/var/run/faillock -p wa -k logins -w /var/log/lastlog -p wa -k logins</pre><code>auditd</code><code>auditctl</code><code>/etc/audit/audit.rules</code><pre>-w
/var/log/tallylog -p wa -k logins -w /var/run/faillock -p wa -k logins -w /var/log/lastlog -p wa -k logins</pre>file in order to watch for unattempted manual
edits of files involved in storing logon events:'
id: xccdf_org.ssgproject.content_rule_audit_rules_login_events
kind: Rule
metadata:
  annotations:
    compliance.openshift.io/rule: audit-rules-login-events
    control.compliance.openshift.io/NIST-800-53: AU-2(d);AU-12(c);AC-6(9);CM-6(a)
    policies.open-cluster-management.io/controls: AU-2(d),AU-12(c),AC-6(9),CM-6(a)
    policies.open-cluster-management.io/standards: NIST-800-53
    creationTimestamp: "2020-09-07T11:43:03Z"
    generation: 1
  labels:
    compliance.openshift.io/profile-bundle: rhcos4
  name: rhcos4-audit-rules-login-events
  namespace: openshift-compliance
  rationale: |-
    Manual editing of these files may indicate nefarious activity,
    such as an attacker attempting to remove evidence of an
    intrusion.
  severity: medium
  title: Record Attempts to Alter Logon and Logout Events
  warning: |-
    <ul><li><code>audit_rules_login_events_tallylog</code></li>
    <li><code>audit_rules_login_events_faillock</code></li>
    <li><code>audit_rules_login_events_lastlog</code></li></ul>
    This rule checks for multiple syscalls related to login
    events and was written with DISA STIG in mind.
    Other policies should use separate rule for
    each syscall that needs to be checked
```

Running Scans 1/2

- There are two types of scans, Platform & node.
- The platform scans are targeting the cluster itself, they're the ocp4-* scans, while the purpose of the node scans is to scan the actual cluster nodes. All the rhcos4-* profiles can be used to create node scans.

Before taking one into use, we'll need to configure how the scans will run. We can do this with the `ScanSetttings` custom resource. The compliance-operator already ships with a default `ScanSettings` object that you can take into use immediately

```
$ oc get -n openshift-compliance scansettings default -o yaml
apiVersion: compliance.openshift.io/v1alpha1
kind: ScanSetting
metadata:
  name: default
  namespace: openshift-compliance
rawResultStorage:
  rotation: 3
  size: 1Gi
roles:
- worker
- master
scanTolerations:
- effect: NoSchedule
  key: node-role.kubernetes.io/master
  operator: Exists
schedule: '0 1 * * *'
```

Running Scan 2/2

To assert the intent of complying with the `rhcos4-moderate` profile, we can use the `ScanSettingBinding` custom resource..

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


$ oc create -n openshift-compliance -f 5_scan_nist_moderate.yaml
scansettingbinding.compliance.openshift.io/nist-moderate created
```

At this point the operator reconciles a `ComplianceSuite` custom resource, we can use this to track the progress of our scan

```
$ oc get -n openshift-compliance compliancesuites -w


NAME             PHASE     RESULT
nist-moderate    RUNNING   NOT-AVAILABLE
```

Remediating Results

When the scan is done, the operator changes the state of the ComplianceSuite object to "Done" and all the pods are transition to the "Completed" state. You can then check the `ComplianceRemediations` that were found with:

```
$ oc get -n openshift-compliance complianceremediations
NAME                                             STATE
workers-scan-auditd-name-format                  NotApplied
workers-scan-coredump-disable-backtraces         NotApplied
workers-scan-coredump-disable-storage            NotApplied
workers-scan-disable-ctrlaltdel-burstaction      NotApplied
workers-scan-disable-users-coredumps             NotApplied
workers-scan-grub2-audit-argument                NotApplied
workers-scan-grub2-audit-backlog-limit-argument  NotApplied
workers-scan-grub2-page-poison-argument          NotApplied
```

```
$ oc edit -n openshift-compliance
complianceremediation/rhcos4-moderate-master-audit-rules-dac-modificatio
n-chmod

apiVersion: compliance.openshift.io/v1alpha1
kind: ComplianceRemediation
metadata:
 labels:
   compliance.openshift.io/scan-name: rhcos4-moderate-master
   compliance.openshift.io/suite: nist-moderate
 name: rhcos4-moderate-master-audit-rules-dac-modification-chmod
 namespace: openshift-compliance
 ownerReferences:
 - apiVersion: compliance.openshift.io/v1alpha1
   blockOwnerDeletion: true
   controller: true
   kind: ComplianceCheckResult
   name: rhcos4-moderate-master-audit-rules-dac-modification-chmod
spec:
 apply: true
 current:
   object:
     apiVersion: machineconfiguration.openshift.io/v1
     kind: MachineConfig
     spec:
       config:
         ignition:
           version: 3.1.0
         storage:
           files:
           - contents:
               source:
data:,-a%20always%2Cexit%20-F%20arch%3Db32%20-S%20chmod%20-F%20auid%3E%3D1000%20-F%20
auid%21%3Dunset%20-F%20key%3Dperm mod%0A-a%20always%2Cexit%20-F%20arch%3Db64%20-S%20c
hmod%20-F%20auid%3E%3D1000%20-F%20auid%21%3Dunset%20-F%20key%3Dperm_mod%0A
             mode: 420
             overwrite: true
             path: /etc/audit/rules.d/75-chmod_dac_modification.rules
```

Red Hat

Cleaning up the operator

Many custom resources deployed with the compliance operators use finalizers to handle dependencies between objects. If the whole operator namespace gets deleted (e.g. with `oc delete ns openshift-compliance`), the order of deleting objects in the namespace is not guaranteed. What can happen is that the operator itself is removed before the finalizers are processed which would manifest as the namespace being stuck in the `Terminating` state.

It is recommended to remove all CRs and CRDs prior to removing the namespace to avoid this issue. The `Makefile` provides a `tear-down` target that does exactly that.

If the namespace is stuck, you can work around by the issue by hand-editing or patching any CRs and removing the `finalizers` attributes manually.

Demo

Code available here:
https://github.com/h00pz/ocp-build

Documentation:
https://docs.openshift.com/container-platform/4.7/security/compliance_operator/compliance-operator-understanding.html

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat