OpenShift Dedicated 4

Cluster security

Configuring security context constraints in OpenShift Dedicated

Last Updated: 2022-08-03

OpenShift Dedicated 4 Cluster security

Configuring security context constraints in OpenShift Dedicated

Abstract

This document provides instructions for configuring security context constraints in OpenShift Dedicated.

Table of Contents

CHAPTER 1. MANAGING SECURITY CONTEXT CONSTRAINTS

1.1. ABOUT SECURITY CONTEXT CONSTRAINTS

Similar to the way that RBAC resources control user access, administrators can use *security context constraints* (SCCs) to control permissions for pods. These permissions include actions that a pod can perform and what resources it can access. You can use SCCs to define a set of conditions that a pod must run with to be accepted into the system.

Security context constraints allow an administrator to control:

- Whether a pod can run privileged containers with the **allowPrivilegedContainer** flag.
- Whether a pod is constrained with the **allowPrivilegeEscalation** flag.
- The capabilities that a container can request
- The use of host directories as volumes
- The SELinux context of the container
- The container user ID
- The use of host namespaces and networking
- The allocation of an **FSGroup** that owns the pod volumes
- The configuration of allowable supplemental groups
- Whether a container requires write access to its root file system
- The usage of volume types
- The configuration of allowable **seccomp** profiles

IMPORTANT

Do not set the **openshift.io/run-level** label on any namespaces in OpenShift Dedicated. This label is for use by internal OpenShift Dedicated components to manage the startup of major API groups, such as the Kubernetes API server and OpenShift API server. If the **openshift.io/run-level** label is set, no SCCs are applied to pods in that namespace, causing any workloads running in that namespace to be highly privileged.

1.1.1. Default security context constraints

The cluster contains several default security context constraints (SCCs) as described in the table below. Additional SCCs might be installed when you install Operators or other components to OpenShift Dedicated.

IMPORTANT

Do not modify the default SCCs. Customizing the default SCCs can lead to issues when some of the platform pods deploy or OpenShift Dedicated is upgraded. During upgrades between some versions of OpenShift Dedicated, the values of the default SCCs are reset to the default values, which discards all customizations to those SCCs.

Table 1.1. Default security context constraints

Security context constraint	Description
anyuid	Provides all features of the **restricted** SCC, but allows users to run with any UID and any GID.
nonroot	Provides all features of the **restricted** SCC, but allows users to run with any non-root UID. The user must specify the UID or it must be specified in the manifest of the container runtime.
restricted	Denies access to all host features and requires pods to be run with a UID, and SELinux context that are allocated to the namespace. This is the most restrictive SCC provided by a new installation and will be used by default for authenticated users. The **restricted** SCC: ● Ensures that pods cannot run as privileged ● Ensures that pods cannot mount host directory volumes ● Requires that a pod is run as a user in a pre-allocated range of UIDs ● Requires that a pod is run with a pre-allocated MCS label ● Allows pods to use any FSGroup ● Allows pods to use any supplemental group NOTE The restricted SCC is the most restrictive of the SCCs that ship by default with the system. However, you can create a custom SCC that is even more restrictive. For example, you can create an SCC that restricts **readOnlyRootFS** to **true** and **allowPrivilegeEscalation** to **false**.

1.1.2. Security context constraints settings

Security context constraints (SCCs) are composed of settings and strategies that control the security features a pod has access to. These settings fall into three categories:

Category	Description
Controlled by a boolean	Fields of this type default to the most restrictive value. For example, **AllowPrivilegedContainer** is always set to **false** if unspecified.
Controlled by an allowable set	Fields of this type are checked against the set to ensure their value is allowed.

Category	Description
Controlled by a strategy	Items that have a strategy to generate a value provide: • A mechanism to generate the value, and • A mechanism to ensure that a specified value falls into the set of allowable values.

CRI-O has the following default list of capabilities that are allowed for each container of a pod:

- **CHOWN**
- **DAC_OVERRIDE**
- **FSETID**
- **FOWNER**
- **SETGID**
- **SETUID**
- **SETPCAP**
- **NET_BIND_SERVICE**
- **KILL**

The containers use the capabilities from this default list, but pod manifest authors can alter the list by requesting additional capabilities or removing some of the default behaviors. Use the **allowedCapabilities**, **defaultAddCapabilities**, and **requiredDropCapabilities** parameters to control such requests from the pods. With these parameters you can specify which capabilities can be requested, which ones must be added to each container, and which ones must be forbidden, or dropped, from each container.

NOTE

You can drop all capabilites from containers by setting the **requiredDropCapabilities** parameter to **ALL**.

1.1.3. Security context constraints strategies

RunAsUser

- **MustRunAs** - Requires a **runAsUser** to be configured. Uses the configured **runAsUser** as the default. Validates against the configured **runAsUser**.
- **MustRunAsRange** - Requires minimum and maximum values to be defined if not using pre-allocated values. Uses the minimum as the default. Validates against the entire allowable range.
- **MustRunAsNonRoot** - Requires that the pod be submitted with a non-zero **runAsUser** or have the **USER** directive defined in the image. No default provided.

- **RunAsAny** - No default provided. Allows any **runAsUser** to be specified.

SELinuxContext

- **MustRunAs** - Requires **seLinuxOptions** to be configured if not using pre-allocated values. Uses **seLinuxOptions** as the default. Validates against **seLinuxOptions**.
- **RunAsAny** - No default provided. Allows any **seLinuxOptions** to be specified.

SupplementalGroups

- **MustRunAs** - Requires at least one range to be specified if not using pre-allocated values. Uses the minimum value of the first range as the default. Validates against all ranges.
- **RunAsAny** - No default provided. Allows any **supplementalGroups** to be specified.

FSGroup

- **MustRunAs** - Requires at least one range to be specified if not using pre-allocated values. Uses the minimum value of the first range as the default. Validates against the first ID in the first range.
- **RunAsAny** - No default provided. Allows any **fsGroup** ID to be specified.

1.1.4. Admission control

Admission control with SCCs allows for control over the creation of resources based on the capabilities granted to a user.

In terms of the SCCs, this means that an admission controller can inspect the user information made available in the context to retrieve an appropriate set of SCCs. Doing so ensures the pod is authorized to make requests about its operating environment or to generate a set of constraints to apply to the pod.

The set of SCCs that admission uses to authorize a pod are determined by the user identity and groups that the user belongs to. Additionally, if the pod specifies a service account, the set of allowable SCCs includes any constraints accessible to the service account.

Admission uses the following approach to create the final security context for the pod:

1. Retrieve all SCCs available for use.
2. Generate field values for security context settings that were not specified on the request.
3. Validate the final settings against the available constraints.

If a matching set of constraints is found, then the pod is accepted. If the request cannot be matched to an SCC, the pod is rejected.

A pod must validate every field against the SCC. The following are examples for just two of the fields that must be validated:

NOTE

These examples are in the context of a strategy using the pre-allocated values.

An FSGroup SCC strategy of **MustRunAs**

If the pod defines a **fsGroup** ID, then that ID must equal the default **fsGroup** ID. Otherwise, the pod is not validated by that SCC and the next SCC is evaluated.

If the **SecurityContextConstraints.fsGroup** field has value **RunAsAny** and the pod specification omits the **Pod.spec.securityContext.fsGroup**, then this field is considered valid. Note that it is possible that during validation, other SCC settings will reject other pod fields and thus cause the pod to fail.

A **SupplementalGroups** SCC strategy of **MustRunAs**

If the pod specification defines one or more **supplementalGroups** IDs, then the pod's IDs must equal one of the IDs in the namespace's **openshift.io/sa.scc.supplemental-groups** annotation. Otherwise, the pod is not validated by that SCC and the next SCC is evaluated.

If the **SecurityContextConstraints.supplementalGroups** field has value **RunAsAny** and the pod specification omits the **Pod.spec.securityContext.supplementalGroups**, then this field is considered valid. Note that it is possible that during validation, other SCC settings will reject other pod fields and thus cause the pod to fail.

1.1.5. Security context constraints prioritization

Security context constraints (SCCs) have a priority field that affects the ordering when attempting to validate a request by the admission controller. A higher priority SCC is moved to the front of the set when sorting. When the complete set of available SCCs are determined they are ordered by:

1. Highest priority first, nil is considered a 0 priority
2. If priorities are equal, the SCCs will be sorted from most restrictive to least restrictive
3. If both priorities and restrictions are equal the SCCs will be sorted by name

By default, the **anyuid** SCC granted to cluster administrators is given priority in their SCC set. This allows cluster administrators to run pods as any user by without specifying a **RunAsUser** on the pod's **SecurityContext**. The administrator may still specify a **RunAsUser** if they wish.

1.2. ABOUT PRE-ALLOCATED SECURITY CONTEXT CONSTRAINTS VALUES

The admission controller is aware of certain conditions in the security context constraints (SCCs) that trigger it to look up pre-allocated values from a namespace and populate the SCC before processing the pod. Each SCC strategy is evaluated independently of other strategies, with the pre-allocated values, where allowed, for each policy aggregated with pod specification values to make the final values for the various IDs defined in the running pod.

The following SCCs cause the admission controller to look for pre-allocated values when no ranges are defined in the pod specification:

1. A **RunAsUser** strategy of **MustRunAsRange** with no minimum or maximum set. Admission looks for the **openshift.io/sa.scc.uid-range** annotation to populate range fields.
2. An **SELinuxContext** strategy of **MustRunAs** with no level set. Admission looks for the **openshift.io/sa.scc.mcs** annotation to populate the level.
3. A **FSGroup** strategy of **MustRunAs**. Admission looks for the **openshift.io/sa.scc.supplemental-groups** annotation.

4. A **SupplementalGroups** strategy of **MustRunAs**. Admission looks for the **openshift.io/sa.scc.supplemental-groups** annotation.

During the generation phase, the security context provider uses default values for any parameter values that are not specifically set in the pod. Default values are based on the selected strategy:

1. **RunAsAny** and **MustRunAsNonRoot** strategies do not provide default values. If the pod needs a parameter value, such as a group ID, you must define the value in the pod specification.
2. **MustRunAs** (single value) strategies provide a default value that is always used. For example, for group IDs, even if the pod specification defines its own ID value, the namespace's default parameter value also appears in the pod's groups.
3. **MustRunAsRange** and **MustRunAs** (range-based) strategies provide the minimum value of the range. As with a single value **MustRunAs** strategy, the namespace's default parameter value appears in the running pod. If a range-based strategy is configurable with multiple ranges, it provides the minimum value of the first configured range.

NOTE

FSGroup and **SupplementalGroups** strategies fall back to the **openshift.io/sa.scc.uid-range** annotation if the **openshift.io/sa.scc.supplemental-groups** annotation does not exist on the namespace. If neither exists, the SCC is not created.

NOTE

By default, the annotation-based **FSGroup** strategy configures itself with a single range based on the minimum value for the annotation. For example, if your annotation reads **1/3**, the **FSGroup** strategy configures itself with a minimum and maximum value of **1**. If you want to allow more groups to be accepted for the **FSGroup** field, you can configure a custom SCC that does not use the annotation.

NOTE

The **openshift.io/sa.scc.supplemental-groups** annotation accepts a comma-delimited list of blocks in the format of **<start>/<length** or **<start>-<end>**. The **openshift.io/sa.scc.uid-range** annotation accepts only a single block.

1.3. EXAMPLE SECURITY CONTEXT CONSTRAINTS

The following examples show the security context constraints (SCC) format and annotations:

Annotated privileged SCC

```
allowHostDirVolumePlugin: true
allowHostIPC: true
allowHostNetwork: true
allowHostPID: true
allowHostPorts: true
allowPrivilegedContainer: true
allowedCapabilities: (1)
- '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: [] (2)
```

```
fsGroup: (3)
  type: RunAsAny
groups: (4)
- system:cluster-admins
- system:nodes
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: 'privileged allows access to all privileged and host
      features and the ability to run as any user, any group, any fsGroup, and with
      any SELinux context.  WARNING: this is the most relaxed SCC and should be used
      only for cluster administration. Grant with caution.'
  creationTimestamp: null
  name: privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: (5)
- KILL
- MKNOD
- SETUID
- SETGID
runAsUser: (6)
  type: RunAsAny
seLinuxContext: (7)
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups: (8)
  type: RunAsAny
users: (9)
- system:serviceaccount:default:registry
- system:serviceaccount:default:router
- system:serviceaccount:openshift-infra:build-controller
volumes:
- '*'
```

1. A list of capabilities that a pod can request. An empty list means that none of capabilities can be requested while the special symbol ***** allows any capabilities.
2. A list of additional capabilities that are added to any pod.
3. The **FSGroup** strategy, which dictates the allowable values for the security context.
4. The groups that can access this SCC.
5. A list of capabilities to drop from a pod. Or, specify **ALL** to drop all capabilities.
6. The **runAsUser** strategy type, which dictates the allowable values for the Security Context.
7. The **seLinuxContext** strategy type, which dictates the allowable values for the Security Context.
8. The **supplementalGroups** strategy, which dictates the allowable supplemental groups for the Security Context.
9. The users who can access this SCC.

The **users** and **groups** fields on the SCC control which users can access the SCC. By default, cluster administrators, nodes, and the build controller are granted access to the privileged SCC. All authenticated users are granted access to the restricted SCC.

Without explicit runAsUser setting

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext: (1)
  containers:
  - name: sec-ctx-demo
    image: gcr.io/google-samples/node-hello:1.0
```

(1) When a container or pod does not request a user ID under which it should be run, the effective UID depends on the SCC that emits this pod. Because restricted SCC is granted to all authenticated users by default, it will be available to all users and service accounts and used in most cases. The restricted SCC uses **MustRunAsRange** strategy for constraining and defaulting the possible values of the **securityContext.runAsUser** field. The admission plug-in will look for the **openshift.io/sa.scc.uid-range** annotation on the current project to populate range fields, as it does not provide this range. In the end, a container will have **runAsUser** equal to the first value of the range that is hard to predict because every project has different ranges.

With explicit runAsUser setting

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 1000 (1)
  containers:
    - name: sec-ctx-demo
      image: gcr.io/google-samples/node-hello:1.0
```

(1) A container or pod that requests a specific user ID will be accepted by OpenShift Dedicated only when a service account or a user is granted access to a SCC that allows such a user ID. The SCC can allow arbitrary IDs, an ID that falls into a range, or the exact user ID specific to the request.

This configuration is valid for SELinux, fsGroup, and Supplemental Groups.

1.4. ROLE-BASED ACCESS TO SECURITY CONTEXT CONSTRAINTS

You can specify SCCs as resources that are handled by RBAC. This allows you to scope access to your SCCs to a certain project or to the entire cluster. Assigning users, groups, or service accounts directly to an SCC retains cluster-wide scope.

NOTE

You cannot assign a SCC to pods created in one of the default namespaces: **default**, **kube-system**, **kube-public**, **openshift-node**, **openshift-infra**, **openshift**. These namespaces should not be used for running pods or services.

To include access to SCCs for your role, specify the **scc** resource when creating a role.

```
$ oc create role <role-name> --verb=use --resource=scc --resource-name=<scc-name> -n
<namespace>
```

This results in the following role definition:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
...
  name: role-name (1)
  namespace: namespace (2)
...
rules:
- apiGroups:
  - security.openshift.io (3)
  resourceNames:
  - scc-name (4)
  resources:
  - securitycontextconstraints (5)
  verbs: (6)
  - use
```

1. The role’s name.
2. Namespace of the defined role. Defaults to **default** if not specified.
3. The API group that includes the **SecurityContextConstraints** resource. Automatically defined when **scc** is specified as a resource.
4. An example name for an SCC you want to have access.
5. Name of the resource group that allows users to specify SCC names in the **resourceNames** field.
6. A list of verbs to apply to the role.

A local or cluster role with such a rule allows the subjects that are bound to it with a role binding or a cluster role binding to use the user-defined SCC called **scc-name**.

NOTE

Because RBAC is designed to prevent escalation, even project administrators are unable to grant access to an SCC. By default, they are not allowed to use the verb **use** on SCC resources, including the **restricted** SCC.

1.5. REFERENCE OF SECURITY CONTEXT CONSTRAINTS COMMANDS

You can manage security context constraints (SCCs) in your instance as normal API objects using the OpenShift CLI (**oc**).

1.5.1. Listing security context constraints

To get a current list of SCCs:

```
$ oc get scc
```

Example output

```
NAME              PRIV    CAPS   SELINUX     RUNASUSER          FSGROUP     SUPGROUP
PRIORITY   READONLYROOTFS   VOLUMES
anyuid            false   []     MustRunAs   RunAsAny           RunAsAny    RunAsAny    10         false
[configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
hostaccess        false   []     MustRunAs   MustRunAsRange     MustRunAs   RunAsAny    <none>
false          [configMap downwardAPI emptyDir hostPath persistentVolumeClaim projected secret]
hostmount-anyuid   false   []     MustRunAs   RunAsAny           RunAsAny    RunAsAny    <none>
false          [configMap downwardAPI emptyDir hostPath nfs persistentVolumeClaim projected
secret]
hostnetwork       false   []     MustRunAs   MustRunAsRange     MustRunAs   MustRunAs   <none>
false          [configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
node-exporter     false   []     RunAsAny    RunAsAny           RunAsAny    RunAsAny    <none>     false
[*]
nonroot           false   []     MustRunAs   MustRunAsNonRoot   RunAsAny    RunAsAny    <none>
false          [configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
privileged        true    [*]    RunAsAny    RunAsAny           RunAsAny    RunAsAny    <none>     false
[*]
restricted        false   []     MustRunAs   MustRunAsRange     MustRunAs   RunAsAny    <none>
false          [configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
```

1.5.2. Examining security context constraints

You can view information about a particular SCC, including which users, service accounts, and groups the SCC is applied to.

For example, to examine the **restricted** SCC:

```
$ oc describe scc restricted
```

Example output

```
Name:     restricted
Priority:    <none>
Access:
  Users:    <none> (1)
  Groups:    system:authenticated (2)
Settings:
  Allow Privileged:   false
  Default Add Capabilities:  <none>
  Required Drop Capabilities:  KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities:   <none>
  Allowed Seccomp Profiles:  <none>
  Allowed Volume Types:
configMap,downwardAPI,emptyDir,persistentVolumeClaim,projected,secret
  Allow Host Network:   false
  Allow Host Ports:   false
  Allow Host PID:   false
  Allow Host IPC:   false
  Read Only Root Filesystem:  false
  Run As User Strategy: MustRunAsRange
    UID:    <none>
    UID Range Min:   <none>
    UID Range Max:   <none>
  SELinux Context Strategy: MustRunAs
    User:    <none>
    Role:    <none>
    Type:    <none>
    Level:    <none>
  FSGroup Strategy: MustRunAs
    Ranges:    <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:    <none>
```

1. Lists which users and service accounts the SCC is applied to.
2. Lists which groups the SCC is applied to.