作者：李晓辉

联系方式：

1. 微信：Lxh_Chat

2. 邮箱：939958092@qq.com

SCC 概述

红帽 OpenShift 里面有个超厉害的安全功能，叫安全上下文约束（SCC）。这玩意儿简直就像是给容器和主机环境之间拉了一道防护网，能限制容器对主机环境的访问权限，可牛了！

具体来说，它能管住好多事儿。首先，要是容器想开启特权模式，那可没那么容易，得经过 SCC 的严格审核。特权模式下容器的权限会大很多，要是乱用很容易出问题，所以 SCC 这个“把关人”必须得好好管着。

其次，容器要是想请求一些额外的功能，比如访问一些特殊的系统资源之类的，也得先跟 SCC 打个招呼。不是想用就能用，得看 SCC 同不同意，这样就能避免一些不必要的风险。

再有，容器要是想把主机的目录挂载成自己的卷来用，也是被 SCC 管着的。毕竟主机的目录里面可能有很重要的东西，不能随便让容器去访问，得确保不会因为容器的操作而影响到主机的安全。

还有，容器的 SELinux 上下文也不能随便改，SELinux 是个很重要的安全机制，它定义了容器的权限和行为，要是容器能随意更改，那安全就很难保证了。所以 SCC 会严格控制这部分内容。

最后，容器的用户 ID 也受到 SCC 的管控。用户 ID 决定了容器在系统中的身份和权限，要是能随便改，那容器就可能获得不该有的权限，从而引发安全问题。所以 SCC 会确保用户 ID 的设置是安全合理的。

总之，这个 SCC 就像是一个超级严格的“管理员”，时刻盯着容器的一举一动，确保它们老老实实地运行，不会对主机环境造成威胁。有了它，用 OpenShift 的时候心里就踏实多了，不用担心容器会乱搞出安全问题，一般来说SCC 控制以下主机资源：

1. **运行特权容器**
   就是容器想变成“超级用户”，拥有更多权限。但 SCC 这家伙可不轻易答应，它得确保容器不会乱来，不然主机的安全就危险了。
2. **请求容器的额外功能**
   容器有时候会想“借”点主机的特殊能力，比如访问一些高级的硬件功能。但 SCC 会仔细审查，看看是不是真的需要，防止容器“贪心”过头。
3. **将主机目录用作卷**
   容器想用主机的文件夹当自己的存储空间。不过，SCC 会管着，防止容器乱动主机的重要文件，确保主机的数据安全。
4. **更改容器的 SELinux 上下文**
   SELinux 是个“安全管家”，容器想改它的设置，SCC 会严格把关，确保容器不会因为乱改设置而破坏安

全规则。

5. **更改用户 ID**
   容器的用户 ID 决定了它的权限，要是乱改，可能会搞出大问题。所以 SCC 会牢牢管住，确保容器的权限合理又安全。

列出SCC

集群管理员可以运行以下命令，以列出 OpenShift 定义的 SCC，此命令会列出所有可用的 SCC，你可以通过它们的名称来查看具体配置。

```
[student@workstation ~]$ oc get scc
NAME                              PRIV    CAPS                    SELINUX     RUNASUSER
anyuid                            false   <no value>              MustRunAs   RunAsAny
hostaccess                        false   <no value>              MustRunAs   MustRunAsRange
hostmount-anyuid                  false   <no value>              MustRunAs   RunAsAny
hostnetwork                       false   <no value>              MustRunAs   MustRunAsRange
hostnetwork-v2                    false   ["NET_BIND_SERVICE"]    MustRunAs   MustRunAsRange
lvms-topolvm-node                 true    <no value>              RunAsAny    RunAsAny
lvms-vgmanager                    true    <no value>              MustRunAs   RunAsAny
machine-api-termination-handler   false   <no value>              MustRunAs   RunAsAny
node-exporter                     true    <no value>              RunAsAny    RunAsAny
nonroot                           false   <no value>              MustRunAs   MustRunAsNonRo
nonroot-v2                        false   ["NET_BIND_SERVICE"]    MustRunAs   MustRunAsNonRo
privileged                        true    ["*"]                   RunAsAny    RunAsAny
restricted                        false   <no value>              MustRunAs   MustRunAsRange
restricted-v2                     false   ["NET_BIND_SERVICE"]    MustRunAs   MustRunAsRange
```

看来默认情况下，openshift提供以下的scc

```
[student@workstation ~]$ oc get scc -o name
securitycontextconstraints.security.openshift.io/anyuid
securitycontextconstraints.security.openshift.io/hostaccess
securitycontextconstraints.security.openshift.io/hostmount-anyuid
securitycontextconstraints.security.openshift.io/hostnetwork
securitycontextconstraints.security.openshift.io/hostnetwork-v2
securitycontextconstraints.security.openshift.io/lvms-topolvm-node
securitycontextconstraints.security.openshift.io/lvms-vgmanager
securitycontextconstraints.security.openshift.io/machine-api-termination-handler
securitycontextconstraints.security.openshift.io/node-exporter
securitycontextconstraints.security.openshift.io/nonroot
securitycontextconstraints.security.openshift.io/nonroot-v2
securitycontextconstraints.security.openshift.io/privileged
securitycontextconstraints.security.openshift.io/restricted
securitycontextconstraints.security.openshift.io/restricted-v2
```

查询SCC属性

这个命令会显示 anyuid SCC 的所有配置细节，包括它允许和限制的内容。

```
[student@workstation ~]$ oc describe scc anyuid
Name:                                           anyuid    # SCC（安全上下文约束）的名称
Priority:                                       10        # SCC 的优先级，优先级越高的 SCC 优先
Access:
  Users:                                        <none>    # 没有指定可以使用该 SCC 的用户，普通用
  Groups:                                       system:cluster-admins  # 只允许 `system:clus
Settings:
  Allow Privileged:                             false     # 不允许容器以特权模式运行，特权模式权限
  Allow Privilege Escalation:                   true      # 允许容器的权限提升，但受到严格限制，不
  Default Add Capabilities:                     <none>    # 没有默认添加任何额外的权限，容器只能使
  Required Drop Capabilities:                   MKNOD     # 容器必须放弃 `MKNOD` 权限，该权限用于
  Allowed Capabilities:                         <none>    # 没有额外允许的权限，容器只能使用系统默
  Allowed Seccomp Profiles:                     <none>    # 没有允许使用任何 Seccomp 安全配置文件
  Allowed Volume Types:                         configMap,downwardAPI,emptyDir,ephemeral,pe
  Allowed Flexvolumes:                          <all>     # 允许使用所有类型的 Flexvolume 插件，
  Allowed Unsafe Sysctls:                       <none>    # 不允许使用任何不安全的系统参数（Sysct
  Forbidden Sysctls:                            <none>    # 没有明确禁止任何系统参数，但默认不安全
  Allow Host Network:                           false     # 不允许容器使用主机的网络栈，容器有自己
  Allow Host Ports:                             false     # 不允许容器绑定主机的端口，防止容器干扰
  Allow Host PID:                               false     # 不允许容器共享主机的进程 ID 空间，容器
  Allow Host IPC:                               false     # 不允许容器共享主机的进程间通信（IPC），
  Read Only Root Filesystem:                    false     # 容器的根文件系统不是只读的，容器可以对
  Run As User Strategy: RunAsAny                # 容器可以以任意用户 ID 运行，这是该 SCC 的主要特
    UID:                                        <none>    # 没有指定具体的用户 ID，容器可以自由选择
    UID Range Min:                              <none>    # 没有指定用户 ID 的最小范围，容器可以自
    UID Range Max:                              <none>    # 没有指定用户 ID 的最大范围，容器可以自
  SELinux Context Strategy: MustRunAs           # 容器必须运行在 SELinux 上下文中，SELinux 是一个
    User:                                       <none>    # 没有指定具体的 SELinux 用户
    Role:                                       <none>    # 没有指定具体的 SELinux 角色
    Type:                                       <none>    # 没有指定具体的 SELinux 类型
    Level:                                      <none>    # 没有指定具体的 SELinux 级别
  FSGroup Strategy: RunAsAny                    # 容器可以以任意的文件系统组运行，比较灵活
    Ranges:                                     <none>    # 没有指定文件系统组的范围，容器可以自由
  Supplemental Groups Strategy: RunAsAny        # 容器可以以任意的补充组运行，比较灵活
    Ranges:                                     <none>    # 没有指定补充组的范围，容器可以自由选择
```

大部分的pod都用的是restricted，因为这个提供 OpenShift 外部资源的有限访问权限

```
[student@workstation ~]$ oc get pod -n openshift-console
NAME                         READY   STATUS    RESTARTS   AGE
console-5fb5c88b98-wb7c7     1/1     Running   5          332d
downloads-7984574494-w8d6g   1/1     Running   7          333d
[student@workstation ~]$ oc describe pod -n openshift-console console-5fb5c88b98-wb7c7 | gr
                       openshift.io/scc: restricted-v2
```

授予SCC特权案例

涉及到特权管理，我们就以lab来完成学习

```
[student@workstation ~]$ lab start appsec-scc
SUCCESS Waiting for cluster
SUCCESS Remove appsec-scc project
```

用普通账号登录，然后创建一个应用，观察其失败的原因，并分配合适的scc给到serviceaccount，将具有scc的sa绑定到应用，即可解决失败的问题

```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

```
oc new-project appsec-scc
```

```
oc new-app --name gitlab \
  --image registry.ocp4.example.com:8443/redhattraining/gitlab-ce:8.4.3-ce.0
```

创建一段时间后，来获取一些pod状态发现是失败的

```
[student@workstation ~]$ oc get pod
NAME                      READY   STATUS    RESTARTS      AGE
gitlab-6fd4f89dbc-hkr54   0/1     Error     1 (9s ago)   17s
```

获取日志看看，发现对特定的目录没有权限

```
[student@workstation ~]$ oc logs gitlab-6fd4f89dbc-hkr54
...
[2024-12-22T08:35:55+00:00] ERROR: directory[/etc/gitlab] (gitlab::default line 26) had an
[2024-12-22T08:35:55+00:00] FATAL: Chef::Exceptions::ChildConvergeError: Chef run process e
```

既然权限不足，那就让应用root权限运行就可以了，我们来创建一个serviceaccount，然后给这个serviceaccount绑定anyuid的scc，让它可以以任何人的uid运行

用管理员登录，给其准备好资源

创建 ServiceAccount，将 anyuid SCC 绑定到这个 ServiceAccount

```
[student@workstation ~]$ oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
[student@workstation ~]$ oc create sa gitlab-sa
[student@workstation ~]$ oc adm policy add-scc-to-user anyuid -z gitlab-sa
```

再登录普通用户，将这个具有特权的serviceaccount分配到他自己的资源

```
[student@workstation ~]$ oc login -u developer -p developer
[student@workstation ~]$ oc set serviceaccount deployment/gitlab gitlab-sa
[student@workstation ~]$ oc rollout restart deployment gitlab
```

观察一下新的pod是否成功运行

```
[student@workstation ~]$ oc get pod
NAME                      READY   STATUS        RESTARTS   AGE
gitlab-7fd875b946-cvq4c   1/1     Running       0          24s
```

ok，看上去没什么问题，我们来访问一下看看

```
[student@workstation ~]$ oc expose service/gitlab --port 80 --hostname gitlab.apps.ocp4.exa
```

访问成功，说明应用成功启动并提供服务了

```
[student@workstation ~]$ curl -sL http://gitlab.apps.ocp4.example.com | grep -i 'sign in'
<meta content='Sign in' property='og:title'>
<meta content='Sign in' property='twitter:title'>
<title>Sign in · GitLab</title>
<h3>Existing user? Sign in</h3>
<input type="submit" name="commit" value="Sign in" class="btn btn-save" />
```

本文档在线版本：https://www.linuxcenter.cn