Lab4. Использование SCC для управления возможностями рабочих нагрузок OpenShift

Приложение может получить доступ только к тем функциям, которые запрашивает модуль и которые одобряет кластер.

Примечание. Учетная запись службы OpenShift — это особый тип учетной записи пользователя, который можно использовать программно без использования учетных данных обычного пользователя.

Кластер развертывает модуль только в том случае, если SCC предоставляет разрешения, запрошенные в контексте безопасности. Когда модуль запускается, он настраивает контейнер, как описано в контексте безопасности модуля.

Задание 1. Создайте развертывание симуляции приложения с помощью базового образа

Это задание демонстрирует:

Стандартные SC и SCC.

Разрешения среды выполнения вашего контейнера

А также показывает:

SCC, назначенный по умолчанию

SC по умолчанию для пода.

Идентификатор пользователя, назначенный для запуска контейнера

Членство пользователя в группах

Как идентификатор пользователя и членство в группах влияют на доступ к данным

В этом задании поэкспериментируем с разрешениями, используя простой контейнер для того, чтобы имитировать запись приложения в файловую систему.

Сценарий:

Вместо приложения с графическим интерфейсом запустим контейнер, который является не чем иным, как базовым образом, и будем использовать оболочку в этом контейнере. Вместо файловой системы смонтируем пустой том Dir внутри контейнера. Таким образом мы можем запускать команды Linux на временно подключенном томе, чтобы протестировать привилегии и средства управления доступом.

Задание2. Создайте файл развертывания deploy_default.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: scc-tutorial-deploy-default
spec:
  selector:
    matchLabels:
```

```
      app: scc-tutorial-default
  template:
    metadata:
      labels:
        app: scc-tutorial-default
    spec:
      containers:
      - image: ubi8/ubi-minimal
        name: ubi-minimal
        command: ['sh', '-c', 'echo "Hello from user $(id -u)" && sleep infinity']
        volumeMounts:
        - mountPath: /var/opt/app/data
          name: data
      serviceAccountName: default
      volumes:
      - emptyDir: {}
        name: data
```

Задание 3. Создайте проект.

Войдите в кластер:

oc login –username=<your-credentials>

Создайте новый проект:

oc new-project test

Создайте развертывание:

oc apply -f deploy_default.yml

Задание 4. Изучите SC и SCC по умолчанию

Вы можете получить полное описание модуля на YAML, чтобы ознакомиться с подробностями. В этом руководстве интересной частью является аннотация, которая показывает, какой SCC был использован, SC контейнера и SC модуля. В этом примере в манифесте явно указана учетная запись службы по умолчанию (для наглядности).

Можено воспользоваться веб-консолью OpenShift или интерфейсом командной строки oc в терминале, чтобы просмотреть результаты.

oc get pod -l app=scc-tutorial-default

```
apiVersion: v1
items:
- apiVersion: v1
  kind: Pod
  metadata:
    annotations:
      openshift.io/scc: restricted-v2

    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop:
        - ALL
      runAsNonRoot: true
      runAsUser: 1000130000
    schedulerName: default-scheduler
    securityContext:
      fsGroup: 1000130000
      seLinuxOptions:
        level: s0:c11,c10
      seccompProfile:
        type: RuntimeDefault
    serviceAccount: default
    serviceAccountName: default
```

На что здесь обратить внимание:

YAML показывает назначенный поду SCC.

SCC показан в annotations.

Развертывание по умолчанию, получило restricted-v2SCC. Это самый приоритетный и наиболее ограничительный SCC, доступный для учетной записи службы.

serviceAccountName задан default. Изменим это позже.

securityContext для пода задан seLinuxOptions и fsGroup. Они взяты из настроек проекта по умолчанию.

Обратите внимание, что у контейнера runAsUser то же самое, что и у пода fsGroup.

Задание 5. Проверьте разрешения вашего контейнера во время выполнения

Получите имя пода:

oc get pod -l app=scc-tutorial-default

Запустите Remote shell в контейнере пода:

oc rsh <pod-name>

Проверьте идентификатор пользователя и членство в группах:

whoami

id

Что мы должны увидеть здесь:

С помощью SCC restricted-v2 вы получили идентификатор пользователя и идентификаторы групп из настроек проекта по умолчанию. Помните, что мы не указали никакого идентификатора пользователя или группы в манифесте развертывания.

Идентификатор пользователя - это тот, который вы видели назначенным в контейнере securityContext.runAsUser

Этот идентификатор пользователя присваивается группе root (ID 0) в качестве идентификатора группы по умолчанию.

Пользователь также является членом группы файловых систем. В этом случае группа файловой системы совпадает с идентификатором пользователя. Это назначается в securityContext.fsGroup пода

Давайте посмотрим, как группа файловой системы использовалась для нашего тома:

ls -ld / /tmp /var/opt/app/data

Покажет разрешения для корневого каталога, временного каталога и подключенного тома:

Том был смонтирован на /var/opt/app/data, как указано в манифесте.

Идентификатором группы у каталога тома является идентификатор fsGroup, и установлен режим "set-group-ID" (т.е. бит sticky заставляет файлы, созданные в этом каталоге, принадлежать этому идентификатору группы).

Для сравнения, корневой и временный каталоги не используют fs Group ID.

Временный каталог доступен для записи каждому пользователю. Корневой каталог доступен для записи только для пользователя root

Посмотрите, что происходит, когда вы записываете файл:

echo hello > /var/opt/app/data/volume.txt Запись в файл на томе

echo hello > /tmp/temp.txt Запись в файл во временном каталоге

echo hello > /fail.txt Попытка записать файл в корневой каталог

ls -l /tmp/temp.txt /var/opt/app/data/volume.txt Проверьте разрешения на доступ к файлам

Файл, который вы записали на том, принадлежит вашему идентификатору пользователя и идентификатору группы файловой системы (из-за "sticky" бита). Это важно. На следующем шаге вы выбираете группу, с которой хотите поделиться файлами (вместо идентификатора по умолчанию).

Файл, который вы записали во временный каталог, принадлежит вашему идентификатору пользователя и идентификатору группы по умолчанию (root).

У вас нет разрешения на запись в корневой каталог. Это подчеркивает, что вы работаете не от имени root или привилегированного пользователя.

Теперь, когда мы узнали как избежать запуска от имени root, пришло время выбрать идентификаторы пользователя или группы.

Задание 6. Попробуйте выполнить развертывание с помощью SC

На этом шаге посмотрим сценарий, где мы развертываем приложение, которому требуется определенный идентификатор пользователя, а также общий идентификатор группы для доступа к данным.

Сначала используем SC в манифесте развертывания, чтобы указать ожидаемый идентификатор пользователя и идентификаторы групп для нашего модуля и контейнера.

Эти SC проверяются на соответствие SCC, назначенным учетной записи службы. Если нет SCC, который может проверить SC, то модуль не запустится.

Запросим специальные разрешения для нашего развертывания.

Мы добавим SC для пода и контейнера, чтобы запросить следующие настройки для управления доступом:

Run as user ID 1234.

Run as group ID 5678.

Добавим дополнительные идентификаторы групп 5777 и 5888.

Укажем идентификатор группы файловой системы, равный 5555.

Создайте файл развертывания deploy_sc.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: scc-tutorial-deploy-sc
spec:
 selector:
  matchLabels:
   app: scc-tutorial-sc
 template:
  metadata:
   labels:
    app: scc-tutorial-sc
  spec:
   containers:
   - image: ubi8/ubi-minimal
    name: ubi-minimal
    command: ['sh', '-c', 'echo "Hello from user $(id -u)" && sleep infinity']
    securityContext:
     runAsUser: 1234
     runAsGroup: 5678
    volumeMounts:
    - mountPath: /var/opt/app/data
     name: data
   serviceAccountName: default
   securityContext:
    fsGroup: 5555
    supplementalGroups: [5777, 5888]
   volumes:
   - emptyDir: {}
    name: data
```

Создайте развертывание

oc apply -f deploy_sc.yml

Посмотрите, как происходит сбой развертывания при запросе привилегий, которые не назначены:

$ oc describe deployment/scc-tutorial-deploy-sc

...

Replicas: 1 desired | 0 updated | 0 total | 0 available | 1 unavailable

...

Conditions:

Type Status Reason

---- ------ ------

Progressing True NewReplicaSetCreated

Available **False** MinimumReplicasUnavailable

ReplicaFailure True FailedCreate

OldReplicaSets: <none>

NewReplicaSet: scc-tutorial-deploy-sc-5c4f64575c (0/1 replicas created)

...

Conditions:

Type Status Reason

---- ------ ------

Progressing True NewReplicaSetCreated

Available False MinimumReplicasUnavailable

ReplicaFailure True FailedCreate

Чтобы получить более конкретную причину сбоя набора реплик, используйте oc get events:

oc get events | grep replicaset/scc-tutorial-deploy-sc

Вывод содержит список SCC, которые использовались для проверки SC при развертывании и завершились неудачей. Предупреждение о сбое развертывания показывает, что не удалось выполнить проверку на соответствие каким-либо ограничениям контекста безопасности из-за значений fsGroup и RunAsUser.

Эта ошибка ожидаема, поскольку манифест развертывания запросил определенные разрешения, а учетная запись службы по умолчанию не может использовать какой-либо SCC, который предоставляет эти разрешения. Это свидетельствует о том, что либо разработчик запросил

слишком большой доступ в манифесте, либо администратору кластера необходимо предоставить SCC, который предоставляет больший доступ.

Задание 7. Создадим и назначим SCC

Теперь мы будем использовать SCC вместе с управлением доступом на основе ролей (RBAC), чтобы предоставить вашим рабочим нагрузкам привилегии, необходимые для выполнения работы.

Создайте файл scc-tutorial-scc.yml

```
kind: SecurityContextConstraints
apiVersion: v1
metadata:
  name: scc-tutorial-scc
allowPrivilegedContainer: false
runAsUser:
  type: MustRunAsRange
  uidRangeMin: 1000
  uidRangeMax: 2000
seLinuxContext:
  type: RunAsAny
fsGroup:
  type: MustRunAs
  ranges:
  - min: 5000
    max: 6000
supplementalGroups:
  type: MustRunAs
  ranges:
  - min: 5000
    max: 6000
```

Создайте развертывание:

```
oc apply -f scc-tutorial-scc.yml
```

Создайте новую учетную запись службы:

```
oc create sa scc-tutorial-sa
```

OpenShift Administration: Operating a Production Kubernetes

Создайте файл rbac.yaml

```
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: use-scc-tutorial-scc
rules:
  - apiGroups: ["security.openshift.io"]
    resources: ["securitycontextconstraints"]
    resourceNames: ["scc-tutorial-scc"]
    verbs: ["use"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: use-scc-tutorial-scc
subjects:
  - kind: ServiceAccount
    name: scc-tutorial-sa
roleRef:
  kind: Role
  name: use-scc-tutorial-scc
  apiGroup: rbac.authorization.k8s.io
```

Выполните следующую команду, чтобы создать роль и привязать ее к вашей учетной записи службы:

```
oc apply -f rbac.yml
```

Итог: мы создали SCC под названием scc-tutorial-scc.

Вы создали учетную запись службы под названием scc-tutorial-sa.

Вы создали роль и привязку к роли, обе с именами use-scc-tutorial-scc.

В рамках создания роли и привязки ролей вы связали SCC с ролью и привязали его к учетной записи службы. Учетнаязапись службы связывается с SCC таким образом, что любой модуль, запущенный от имени учетной записи службы, имеет доступ к SCC.

На следующем шаге, как разработчик, вы будете использовать эту учетную запись службы для исправления развертывания, при котором не удалось проверить ваш SC на предыдущем шаге.

Задание 8. Создайте развертывание, используя учетную запись службы, которая может использовать SCC

Создайте файл deploy_sc_sa.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: scc-tutorial-deploy-sc-sa
spec:
 selector:
  matchLabels:
   app: scc-tutorial-sc-sa
 template:
  metadata:
   labels:
    app: scc-tutorial-sc-sa
  spec:
   containers:
   - image: ubi8/ubi-minimal
    name: ubi-minimal
    command: ['sh', '-c', 'echo "Hello from user $(id -u)" && sleep infinity']
    securityContext:
     runAsUser: 1234
     runAsGroup: 5678
    volumeMounts:
    - mountPath: /var/opt/app/data
     name: data
   serviceAccountName: scc-tutorial-sa
   securityContext:
    fsGroup: 5555
    supplementalGroups: [5777, 5888]
   volumes:
```

```
      - emptyDir: {}
        name: data
```

Создайте развертывание:

```
oc apply -f deploy_sc_sa.yml
```

Проверьте эффект после добавления пользовательских SC и SCC

Теперь, когда вы настроили SC и создали пользовательский SCC, давайте посмотрим результаты.

Получите полное описание модуля на YAML, чтобы ознакомиться с подробностями. В этом руководстве интересной частью является аннотация, которая показывает, какой SCC был использован, SecurityContext контейнера и пода. А также проверить, использована ли сервисная учетная запись scc-tutorial-sa.

Чтобы получить подробную информацию для pod, используйте oc get с label app=scc-tutorial-sc-sa и параметром вывода yaml:

```
oc get pod -l app=scc-tutorial-sc-sa -o yaml
```

В следующем фрагменте представлены наиболее интересные части:

```
  ...
  metadata:
    annotations:
      ...
      openshift.io/scc: scc-tutorial-scc
      ...
  spec:
    containers:
      ...
      securityContext:
        runAsGroup: 5678
        runAsUser: 1234
        ...
      volumeMounts:
      - mountPath: /var/opt/app/data
      ...
    securityContext:
      fsGroup: 5555
```

```
    supplementalGroups:
    - 5777
    - 5888
  serviceAccount: scc-tutorial-sa
  serviceAccountName: scc-tutorial-sa
  ...
```

Задание 9. Проверьте разрешения вашего контейнера во время выполнения

Получите название пода:

oc get pod -l app=scc-tutorial-sc-sa

Запустите оболочку в контейнере пода:

oc rsh <pod-name>

Проверьте идентификатор пользователя и членство в группах

whoami

id

Теперь используется идентификатор пользователя (uid) и идентификатор группы (gid) из манифеста развертывания.

Дополнительные группы, которые запросили, также были созданы, и пользователь является членом этих групп.

Пользователь также является членом группы файловых систем. Этот идентификатор указан в группе podsecurityContext.fs.

Чтобы увидеть, как группа файловой системы использовалась для доступа к тому, выполните команду:

ls -ld / /tmp /var/opt/app/data

Том монтируется на/var/opt/app/data (как указано в манифесте).

Идентификатор группы каталога томов - это fs Group, которую мы запросили в манифесте.

Установлен режим "set-group-ID" (т.е. sticky бит заставляет файлы, созданные в этом каталоге, принадлежать этому идентификатору группы).

Корневой и временный каталоги не изменились по сравнению с нашим примером по умолчанию.

Проверим, что происходит, когда мы записывам файл.

echo hello > /var/opt/app/data/volume.txt Запись в файл на томе

echo hello > /tmp/temp.txt Запись в файл во временном каталоге

echo hello > /fail.txt Попытка записать файл в корневой каталог

ls -l /tmp/temp.txt /var/opt/app/data/volume.txt Проверка разрешений на доступ к файлам.

Что мы должны увидеть.

В /tmp созданный нами файл принадлежит 1234/5678 (указанный нами uid/gid вместо проекта по умолчанию, например 1000620000/root).

На томе созданный вами файл принадлежит 1234/5555 (указанный uid/ fs Group вместо проекта по умолчанию, например 1000620000/1000620000).

Мы не запускали от имени root и не использовали группу root.

Если бы том был общим хранилищем, контейнеры с разными идентификаторами пользователей могли бы обмениваться данными с членами одной и той же группы.

В этом примере использовалась только fs Group, но мы можем видеть, что другие дополнительные группы, которые мы указали, также были созданы и назначены пользователю.

Удалите созданные ресурсы:

oc delete deployment/scc-tutorial-deploy-default

oc delete deployment/scc-tutorial-deploy-sc

oc delete deployment/scc-tutorial-deploy-sc-sa

oc delete rolebindings/use-scc-tutorial-scc

oc delete role/use-scc-tutorial-scc

oc delete sa/scc-tutorial-sa

oc delete scc/scc-tutorial-scc