9. Create SAs

Switch cluster kubectl config use-context k8s

context

A Pod fails to run because of an incorrectly specified ServiceAcccount.

task

create a new ServiceAccount named frontend-sa in the existing namespace qa , which must not have access to any secrets.Inspect the Pod named frontend running in the namespace qa . Edit the Pod to use the newly created serviceAccount

problem solving ideas

Configure Service Accounts for Pods

```
Keyword: ServiceAccount "must not have access to any secrets"
1. Get the sa template
$ kubectl create serviceaccount frontend-sa -n qa --dry-run -o yaml
2. Find automatic mounting through official documents
$ k edit pod frontend -n qa
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: frontend
  name: frontend
spec:
  serviceAccountName: frontend-sa #add this line
  automountServiceAccountToken: false #Add this line
  containers:
  - image: nginx
    name: frontend
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

3. Modify the serviceAccountName in the pod
4. Create a pod to delete other sa
```

10. Trivy detects image security

Switch cluster kubectl config use-context k8s

task

Use the Trivy open-source container scanner to detect images with severe vulnerabilities used by Pods in the namespace yavin . Look for images with High or Critical severity vulnerabilities, and delete the Pods that use those images. Trivy is pre-installed on the cluster's ma ster node only; it is not available on the base system or the worker nodes. You'll have to connect to the cluster's master node to use Trivy

problem solving ideas

```
Keywords: Trivy scanner High or Critical
1. Switch the cluster, ssh to the corresponding master
2. Get pod to scan the corresponding image, there can be no High or Critical
$ docker run ghcr.io/aquasecurity/trivy:latest image nginx:latest |grep 'High|Critical'
3. Delete the problematic image pod
$ docker rmi <image>
```

11. Create a secret

Switch cluster kubectl config use-context k8s

task

Retrieve the content of the existing secret named db1-test in the istio-system namespace. store the username field in a file named /cks/11/old-username.txt , and the password field in a file named /cks/11/old-pass.txt. You must create both files; they don't existyet. Do not use/modify the created files in! the following steps, create new temporary files if needed. Create a new secret named test-workflow in the istio-system namespace, with the following content:

username: thanos

password : hahahaha

Finally, create a new Pod that has access to the secret test-workflow via volume:

pod name dev-pod

namespace istio-system

container name dev-container

image nginx:1.9

volume name dev-volume

mount path /etc/test-secret

problem solving ideas

Secret

```
Keyword: secret
1. Get the username and passwd of db1-test
$ kubectl get secrets db1-test -n istio-system -o yaml
$ echo -n "aGFoYTAwMQ==" | base64 -d > /cks/11/old-pass.txt
$ echo -n "dG9t" | base64 -d > /cks/11/old-username.txt

2. Create a secret named test-workflow
$ kubectl create secret generic test-workflow --from-literal=username=thanos --from-literal=password=hahahaha -n istio-system

3. More pods that need to create secrets
$ cat secret-pod.yaml
apiVersion: v1
```

```
  kind: Pod
  metadata:
    name: dev-pod
    namespace: istio-system
  spec:
    containers:
    - name: dev-container
      image: nginx:1.9
      volumeMounts:
      - name: foo
        mountPath: "/etc/test-secret"
        readOnly: true
    volumes:
    - name: dev-volume
      secret:
        secretName: test-workflow

  k create -f secret-pod.yaml
```

12. kube-bench

Switch cluster kubectl config use-context k8s65
context
ACIS Benchmark tool was run against the kubeadm-created cluster and found multiple issues that must be addressed immediately.
Fix all issues via configuration and restart the affected components to ensure the
new settings take effect. Fix all of the following violations that were found against the API server:

Ensure that the
1.2.7 --authorization-mode FAIL argument is not set to AlwaysAllow
Ensure that the
1.2.8 --authorization-mode FAIL argument includes Node
Ensure that the
1.2.9 --authorization-mode FAIL argument includes RBAC
Ensure that the
1.2.18 --insecure-bind-address FAIL argument is not set
Ensure that the
1.2.19 --insecure-port FAIL argument is set to 0
Fix all of the following violations that were found against the kubelet:

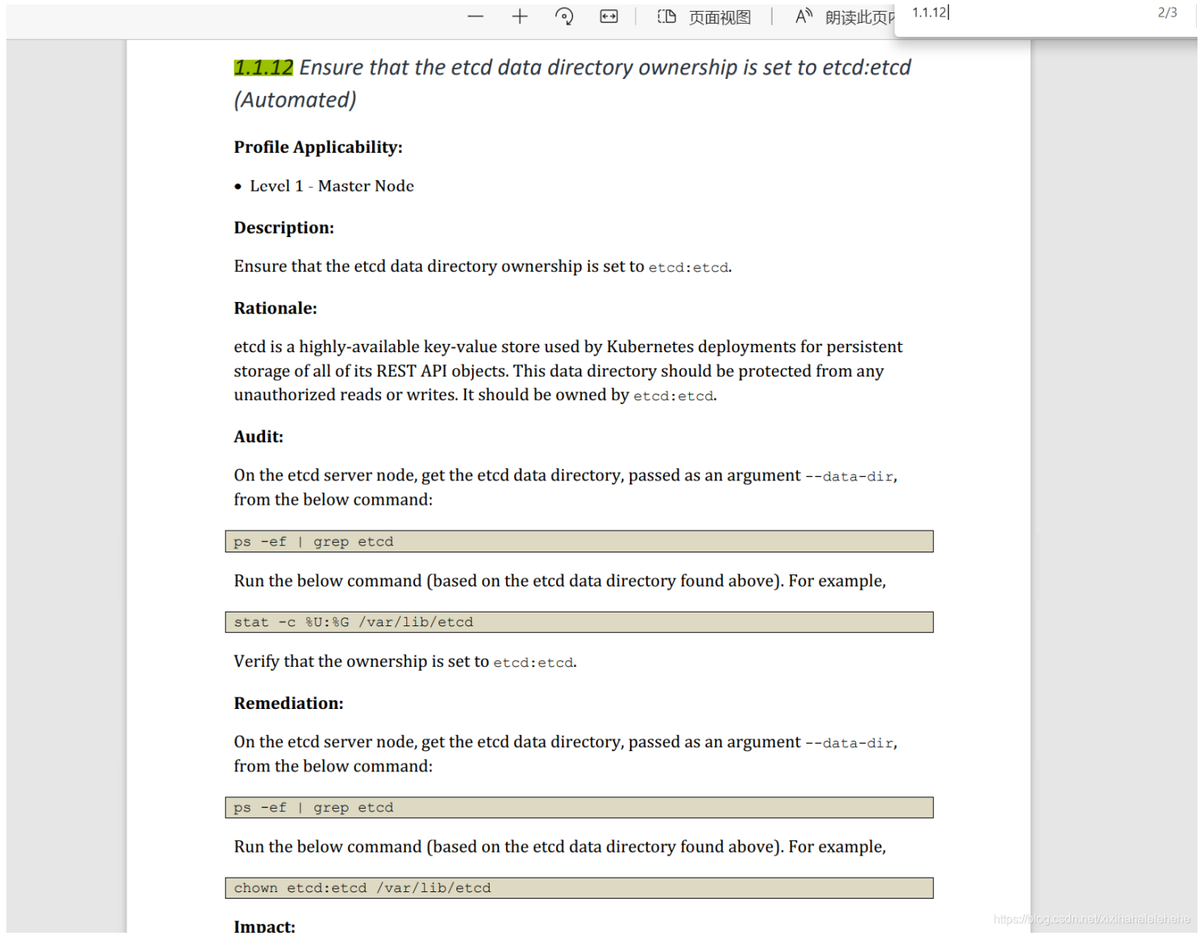
Ensure that the
4.2.1 anonymous-auth FAIL argument is set to false
Ensure that the
4.2.2 --authorization-mode FAIL argument is not set to AlwaysAllow
Use webhook authn/authz

problem solving ideas

```
Keyword: Look at the entry to determine whether it is a scan
1. Switch the machine to the corresponding ssh to the master node
2. kube-bench run to find the corresponding entry, and then repair
docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.20
There is an ETCD in the exam
```

Case 1

```
$ docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.20
 …
[FAIL] 1.1.12 Ensure that the etcd data directory ownership is set to etcd: etcd (Automated)
.........
```



1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the etcd data directory ownership is set to `etcd:etcd`.

Rationale:

etcd is a highly-available key-value store used by Kubernetes deployments for persistent storage of all of its REST API objects. This data directory should be protected from any unauthorized reads or writes. It should be owned by `etcd:etcd`.

Audit:

On the etcd server node, get the etcd data directory, passed as an argument `--data-dir`, from the below command:

```
ps -ef | grep etcd
```

Run the below command (based on the etcd data directory found above). For example,

```
stat -c %U:%G /var/lib/etcd
```

Verify that the ownership is set to `etcd:etcd`.

Remediation:

On the etcd server node, get the etcd data directory, passed as an argument `--data-dir`, from the below command:

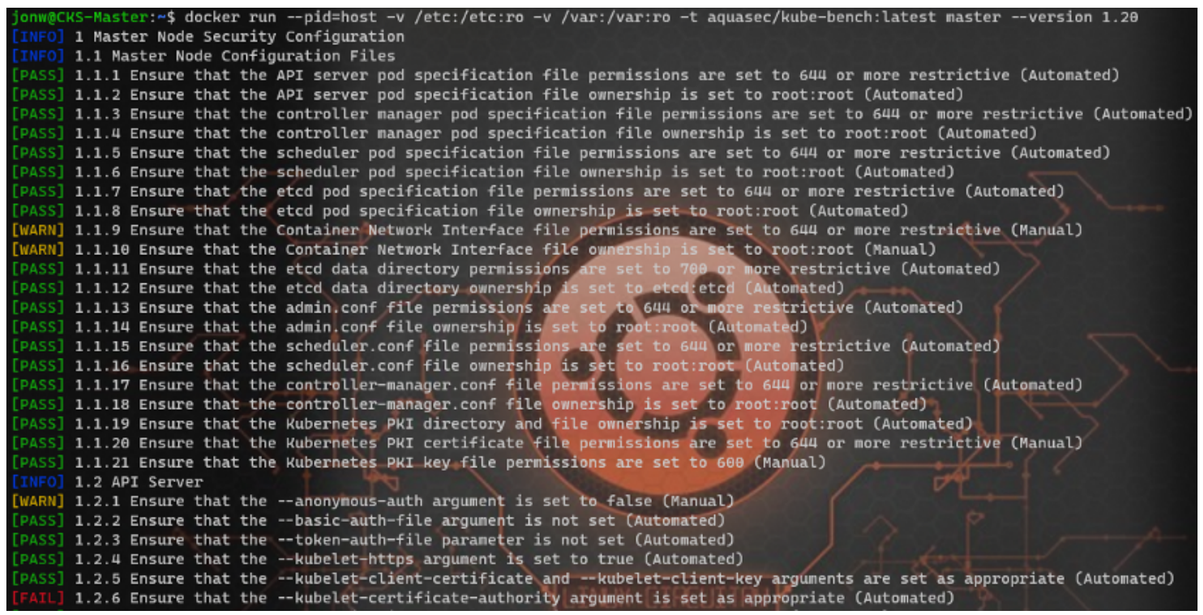
```
ps -ef | grep etcd
```

Run the below command (based on the etcd data directory found above). For example,

```
chown etcd:etcd /var/lib/etcd
```

Impact:

Case 2



```
jonw@CKS-Master:~$ docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.20
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.3 Ensure that the controller manager pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.4 Ensure that the controller manager pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.5 Ensure that the scheduler pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.6 Ensure that the scheduler pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.7 Ensure that the etcd pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.8 Ensure that the etcd pod specification file ownership is set to root:root (Automated)
[WARN] 1.1.9 Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Manual)
[WARN] 1.1.10 Ensure that the Container Network Interface file ownership is set to root:root (Manual)
[PASS] 1.1.11 Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)
[PASS] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)
[PASS] 1.1.13 Ensure that the admin.conf file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.14 Ensure that the admin.conf file ownership is set to root:root (Automated)
[PASS] 1.1.15 Ensure that the scheduler.conf file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.16 Ensure that the scheduler.conf file ownership is set to root:root (Automated)
[PASS] 1.1.17 Ensure that the controller-manager.conf file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.18 Ensure that the controller-manager.conf file ownership is set to root:root (Automated)
[PASS] 1.1.19 Ensure that the Kubernetes PKI directory and file ownership is set to root:root (Automated)
[PASS] 1.1.20 Ensure that the Kubernetes PKI certificate file permissions are set to 644 or more restrictive (Manual)
[PASS] 1.1.21 Ensure that the Kubernetes PKI key file permissions are set to 600 (Manual)
[INFO] 1.2 API Server
[WARN] 1.2.1 Ensure that the --anonymous-auth argument is set to false (Manual)
[PASS] 1.2.2 Ensure that the --basic-auth-file argument is not set (Automated)
[PASS] 1.2.3 Ensure that the --token-auth-file parameter is not set (Automated)
[PASS] 1.2.4 Ensure that the --kubelet-https argument is set to true (Automated)
[PASS] 1.2.5 Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Automated)
[FAIL] 1.2.6 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)
```

1.2.6 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Verify kubelet's certificate before establishing connection.

Rationale:

The connections from the apiserver to the kubelet are used for fetching logs for pods, attaching (through kubectl) to running pods, and using the kubelet’s port-forwarding functionality. These connections terminate at the kubelet’s HTTPS endpoint. By default, the apiserver does not verify the kubelet’s serving certificate, which makes the connection subject to man-in-the-middle attacks, and unsafe to run over untrusted and/or public networks.

Audit:

Run the following command on the master node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--kubelet-certificate-authority` argument exists and is set as appropriate.

Remediation:

Follow the Kubernetes documentation and setup the TLS connection between the apiserver and kubelets. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--kubelet-certificate-authority` parameter to the path to the cert file for the certificate authority.

```
--kubelet-certificate-authority=<ca-string>
```

Impact:

You require TLS to be configured on apiserver as well as kubelets.

```
$ cat /etc/kubernetes/kubelet.conf
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM1ekNDQWMrZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeE1Ea3hNekE0T
    server: https://192.168.211.40:6443
```

```
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: system: node: master
  name: system:node:master@kubernetes
current-context: system:node:master@kubernetes
kind: Config
preferences: {}
users:
- name: system:node:master
  user:
    client-certificate: /var/lib/kubelet/pki/kubelet-client-current.pem
    client-key: /var/lib/kubelet/pki/kubelet-client-current.pem


$ echoLS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM1ekNDQWMrZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeE1Ea3hNekE0TVRjd09Wb1hEVE14TURreE1UQ

$ cat /etc/kubernetes/pki/apiserver-kubelet-ca.crt
-----BEGIN CERTIFICATE-----
MIIC5zCCAc+gAwIBAgIBADANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDEwprdWJl
cm5ldGVzMB4XDTIxMDkxMzA4MTcwOVoXDTMxMDkxMTA4MTcwOVowFTETMBEGA1UE
AxMKa3ViZXJuZXRlczCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOPA
wydYcbRAZ7F2jQjjGIqVfPSdyUxS1LB0e0Jhgtb1rUpmv12uCmfUPhIDRvzvKhfs
yr06avNnsfIvRzr+zj1zCOX3TSZbf4kCZ8N88JRIDt6pCKIISLe8uksuJO095ejv
nVuoGmBFeKlcuz1GKQKTL7jSZ7+4MrMayE9HdnbzuZMTN6fSoExF1xq3oC0i+eBB
3T+Pc1yyWCMrwWXG9UFf4Z8xXDhgn/hDJHJERvyklFjqxjZD+xPyKq88vO9K+JmV
ly6Ll5kmel7OtOYMDg1i0S0IQfI0kTiOvwCl8s55PWQ8mNkYryIYxKNBAO/x/QB8
00Yk4afVDIBYWduHr1kCAwEAAaNCMEAwDgYDVR0PAQH/BAQDAgKkMA8GA1UdEwEB
/wQFMAMBAf8wHQYDVR0OBBYEFG9NaDMQIT4ss05ZSLg/WDbl6zdiMA0GCSqGSIb3
DQEBCwUAA4IBAQCikJPGwsAwaVxqlWk2oGunnG77P4eqZbYtwZLnhoviKGSubX5r
r0skRWGcTT12PCLiEc2hLvljvT/lQ3/Muv6nbYDPSx9b3EoEF2YxHqWMc5BXJU/9
NpAXc+m/7MrZAepW1sw+ncSTdH01NbQ1BBz82kFzMYNoJ+LJcExlvSkzrMuWCWiA
6vbljHFrJBw5kE1Ow/GNK9HTTR99zuoe8LrRojsPATf/vzAJdLQki71riYtsFE04
YOreqA8cfMxdnPcAxo9gRVC0aC0DwaQx/Z5+cE70UmyvqPrn8Tbz1Mn9kuWePMa9
OW4b7Stb9Zz6PB7zjK97txsxtEEeu+/Lx2W+
-----END CERTIFICATE-----


$ vim /etc/kubernetes/manifests/kube-apiserver.yaml
.....
--kubelet-certificate-authority=/etc/kubernetes/pki/apiserver-kubelet-ca.crt
...................

$ kubectl get pods -n kube-system | grep kube-apiserver
$ docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.20
```

13. gVsior

Change cluster kubectl config use-context k8s67

context

This cluster uses containerd as CRI runtime. Containerd's default runtime handler is runc . Containerd has been prepared to support an additional runtime handler ,runsc (gVisor). Task:

Create a RuntimeClass named untrusted using the prepared runtime handler named runsc . Update all Pods in the namespace client to run on gvisor, unless they are already running on anon-default runtime handler. You can find a skeleton manifest file at /cks/13/rc.yaml

problem solving ideas

RuntimeClass

```
Keywords: gVisor
1. Switch the cluster and create a runtimeclass with the official website document
$ vim rc.yaml
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: untrusted
handler: runsc

$ k -f rc.yaml create

2. More questions require creating a pod to use this runtime
$ k edit pod mypod -n client
apiVersion: v1
kind: Pod
metadata:
  name: mypod
  namespace: client
spec:
  runtimeClassName: untrusted
 …
```

14. Audit

Switch cluster kubectl config use-context k8s

task

Enable audit logs in the cluster. To do so, enable the log backend, and ensure that:

logs are stored at /var/log/kubernetes/audit-logs.txt

log files are retained for 5 days at maximum, a number of 10 auditlog files are retained

A basic policy is provided at /etc/kubernetes/logpolicy/sample-policy.yaml . it only specifies what not to log. The base policy is located on the cluster's master node. Edit and extend the basic policy to log:

namespace changes at RequestResponse level
the request body of pods changes in the namespace front-apps
configMap and secret changes in all namespaces at the Metadata level
Also, add a catch-all rule to log all other requests at the Metadata level. Don't forget to apply
problem solving ideas
audit

```
Keyword: policy
1. Switch the cluster to log in to the master, then create a directory, modify yaml, and enable auditing
$ mkdir /var/log/kubernetes/
$ mkdir /etc/kubernetes/logpolicy/
$ cat /etc/kubernetes/logpolicy/sample-policy.yaml
$ cat policy.yaml
apiVersion: audit.k8s.io/v1
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  - level: RequestResponse
    resources:
    -group: ""
      resources: ["namespaces"]

  -level: Request
    resources:
    -group: "" # core API group
      resources: ["pods"]
    namespaces: ["front-apps"]

  - level: Metadata
    resources:
    -group: ""
      resources: ["secrets","configmaps"]

  - level: Metadata
    omitStages:
      - "RequestReceived"

2. More official website documents to modify the corresponding strategy
$ vim /etc/kubernetes/manifests/kube-apiserver.yaml
    - --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml # add
    - --audit-log-path=/var/log/kubernetes/audit-logs.txt # add
    - --audit-log-maxage=5 # add
    - --audit-log-maxbackup=10
 …
   - mountPath: /etc/kubernetes/logpolicy # add
     name: audit # add
```

```
    hostNetwork: true
    priorityClassName: system-node-critical
    volumes:
    - hostPath: # add
        path: /etc/kubernetes/logpolicy # add
        type: DirectoryOrCreate # add
      name: audit # add


3. Restart kubelet
$ systemctl restart kubelet
$ k get pods -n kube-system | grep api
$ cat /var/log/kubernetes/audit-logs.txt
```

15. Default Network Policy

Switch cluster kubectl config use-context k8s
context
A default-deny NetworkPolicy avoids to accident all y expose a Pod in a namespace that doesn't have any other NetworkPolicy defined.
Create a new default-deny NetworkPolicy named denynetwork in the namespace development for all traffic of type Ingress . The new NetworkPolicy must deny all Ingress traffic in the namespace development . Apply the newly created default-deny NetworkPolicy to all Pods running in namespace development . You can find a skeleton manifest file
problem solving ideas
NetworkPolicy

```
Keywords: NetworkPolicy defined
1. Observe clearly whether to reject all or other conditions by default, and more questions require official documents to write yaml
$ cat denynetwork.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: denynetwork
  namespace: development
spec:
  podSelector: {}
  policyTypes:
  -Ingress

$ k create -f denynetwork.yaml
```

16. falco detection output log format

Change Falco rule to get custom output format

Rule:	*"A shell was spawned in a container with an attached terminal"*
Output Format:	TIME,USER-NAME,CONTAINER-NAME,CONTAINER-ID
Priority:	WARNING

```
$ ssh node1
$ systemctl stop falco
$ falco
$ cd /etc/falco/
$ ls
falco_rules.local.yaml falco_rules.yaml falco.yaml k8s_audit_rules.yaml rules.available rules.d

$ rep -r "A shell was spawned in a container with an attached terminal" *
falco_rules.yaml: A shell was spawned in a container with an attached terminal (user=%user.name user_loginuid=%user.loginuid %container.info

#update configuration
root@node1:/etc/falco# cat falco_rules.local.yaml
-rule: Terminal shell in container
  desc: A shell was used as the entrypoint/exec point into a container with an attached terminal.
  condition: >
    spawned_process and container
    and shell_procs and proc.tty != 0
    and container_entrypoint
    and not user_expected_terminal_shell_in_container_conditions
  output: >
    %evt.time,%user.name,%container.name,%container.id
    shell=%proc.name parent=%proc.pname cmdline=%proc.cmdline terminal=%proc.tty container_id=%container.id image=%container.image.repository)
  priority: WARNING
  tags: [container, shell, mitre_execution]

$ falco
Mon May 24 00:07:13 2021: Falco version 0.28.1 (driver version 5c0b863ddade7a45568c0ac97d037422c9efb750)
Mon May 24 00:07:13 2021: Falco initialized with configuration file /etc/falco/falco.yaml
Mon May 24 00:07:13 2021: Loading rules from file /etc/falco/falco_rules.yaml:
Mon May 24 00:07:13 2021: Loading rules from file /etc/falco/falco_rules.local.yaml: #Configuration takes effect
Mon May 24 00:07:13 2021: Loading rules from file /etc/falco/k8s_audit_rules.yaml:
Mon May 24 00:07:14 2021: Starting internal webserver, listening on port 8765
00:07:30.297671117: Warning Shell history had been deleted or renamed (user=root user_loginuid=-1 type=openat command=bash fd.name=/root/.bash_history name=/root/.bash_history path=<NA> o
```

```
format change
00:07:33.763063865: Warning 00:07:33.763063865,root,k8s_apache_apache_default_3ece2efb-fe49-4111-899f-10d38a61bab6_0,84dd6fe8a9ad shell=bash parent=runc cmdline=b ash terminal=34816 conta
```