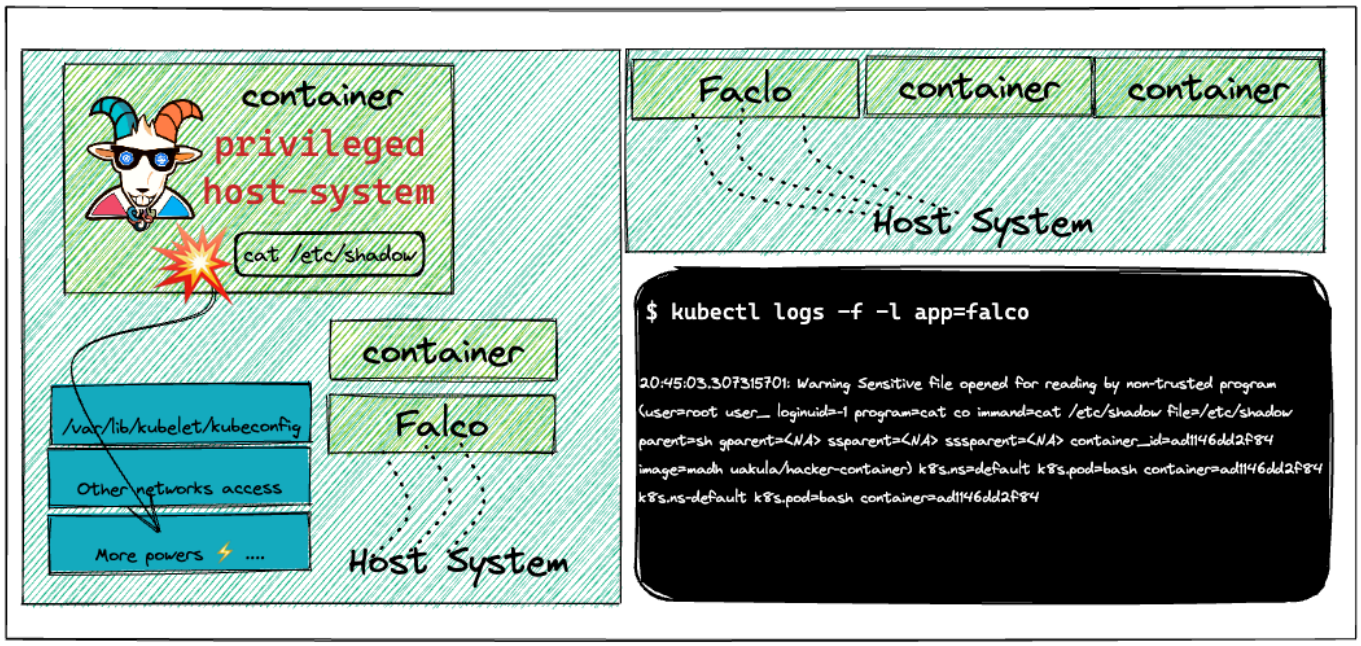
⎈ Falco - Runtime security monitoring & detection

🙌 Overview

The containers and their infrastructure are immutable. It means it's very difficult to detect certain attacks, vulnerabilities, and detections using traditional tools and technologies. In this scenario, we will see how we can leverage the popular open-source tool like Falco to detect and perform runtime security monitoring using the ruleset in action.



By the end of the scenario, we will understand and learn the following

1. Deploying the helm chart into the Kubernetes cluster
2. Performing the log analysis and detection of security events in the Kubernetes Cluster
3. Use, analyze and detect security issues in near real-time using Falco

⚡ The story

This scenario is to deploy runtime security monitoring & detection for containers and Kubernetes resources. Also, explore and see how we can detect certain issues and perform detection rules analysis based on triggers.

NOTE

Make sure you run the following deployment using Helm with v3. Refer to helm installation

INFO

- To deploy the Falco helm chart run the following commands

```
helm repo add falcosecurity https://falcosecurity.github.io/charts
```

```
helm repo update
```

```
helm install falco falcosecurity/falco
```

```
$ helm repo add falcosecurity https://falcosecurity.github.io/charts
"falcosecurity" has been added to your repositories
$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "falcosecurity" chart repository
Update Complete. ⎈Happy Helming!⎈
$ helm install falco falcosecurity/falco
NAME: falco
LAST DEPLOYED: Fri Jun  4 01:55:22 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Falco agents are spinning up on each node in your cluster. After a few
seconds, they are going to start monitoring your containers looking for
security issues.


No further action should be required.


Tip:
You can easily forward Falco events to Slack, Kafka, AWS Lambda and more with falcosidekick.
Full list of outputs: https://github.com/falcosecurity/charts/falcosidekick.
You can enable its deployment with `--set falcosidekick.enabled=true` or in your values.yaml.
See: https://github.com/falcosecurity/charts/blob/master/falcosidekick/values.yaml for configuration values.
$
```

🎯 Goal

TIP

Deploy the Falco and detect security events in near real-time using the pre-defined ruleset.

Hints & Spoilers

Want to know more?

Solution & Walkthrough

Method 1

INFO

`Falco`, the cloud-native runtime security project, is the de facto Kubernetes threat detection engine. Falco was created by Sysdig in 2016 and is the first runtime security project to join CNCF as an incubation-level project. Falco detects unexpected application behavior and alerts on threats at runtime.

- Falco uses system calls to secure and monitor a system, by:
    - Parsing the Linux system calls from the kernel at runtime
    - Asserting the stream against a powerful rules engine
    - Alerting when a rule is violated
- Falco ships with a default set of rules that check the kernel for unusual behavior such as:
    - Privilege escalation using privileged containers
    - Namespace changes using tools like `setns`
    - Read/Writes to well-known directories such as /etc, /usr/bin, /usr/sbin, etc
    - Creating symlinks
    - Ownership and Mode changes
    - Unexpected network connections or socket mutations
    - Spawned processes using execve
    - Executing shell binaries such as sh, bash, csh, zsh, etc
    - Executing SSH binaries such as ssh, scp, sftp, etc
    - Mutating Linux coreutils executables
    - Mutating login binaries
    - Mutating shadowutil or passwd executables such as shadowconfig, pwck, chpasswd, getpasswd, change, useradd, etc, and others.

- Get more details about the Falco deployment by running the following command

```
kubectl get pods --selector app=falco
```

```
$ kubectl get pods --selector app=falco
NAME          READY   STATUS    RESTARTS   AGE
falco-8jqnt   1/1     Running   0          64s
falco-psvp2   1/1     Running   0          64s
$
```

- Manually obtaining the logs from the Falco systems using the following command

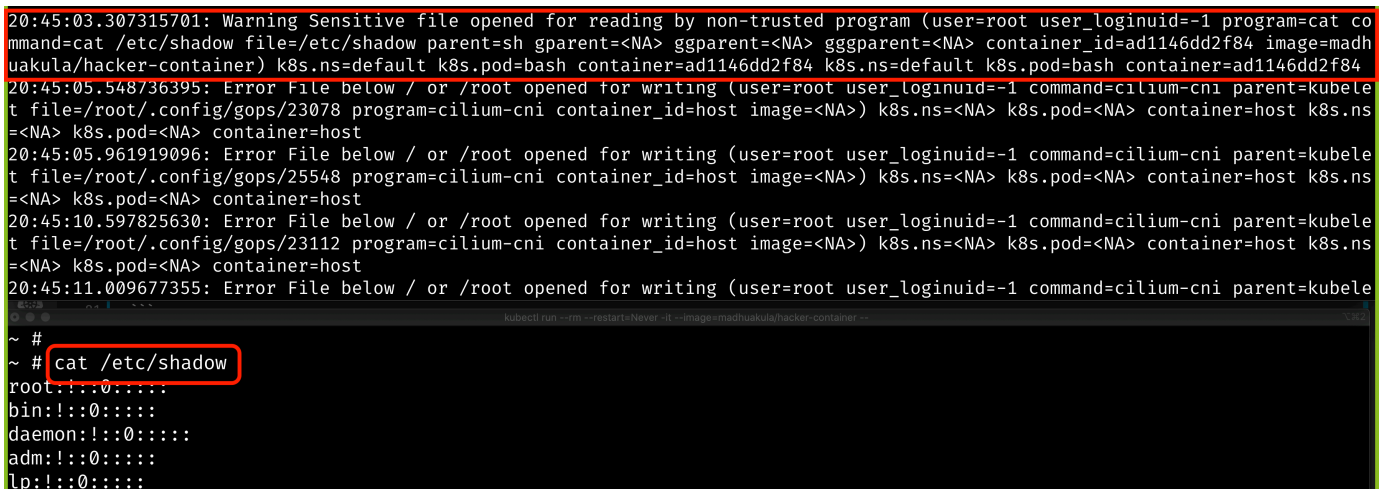
```
kubectl logs -f -l app=falco
```

- Now, let's spin up a hacker container and read a sensitive file and see if that detects by Falco

```
kubectl run --rm --restart=Never -it --image=madhuakula/hacker-container -- bash
```

- Let's read the sensitive file `/etc/shadow`

```
cat /etc/shadow
```



- Hooray 🥳 , now we can see that Falco detected this and notified

References

- https://falco.org/

Edit this page