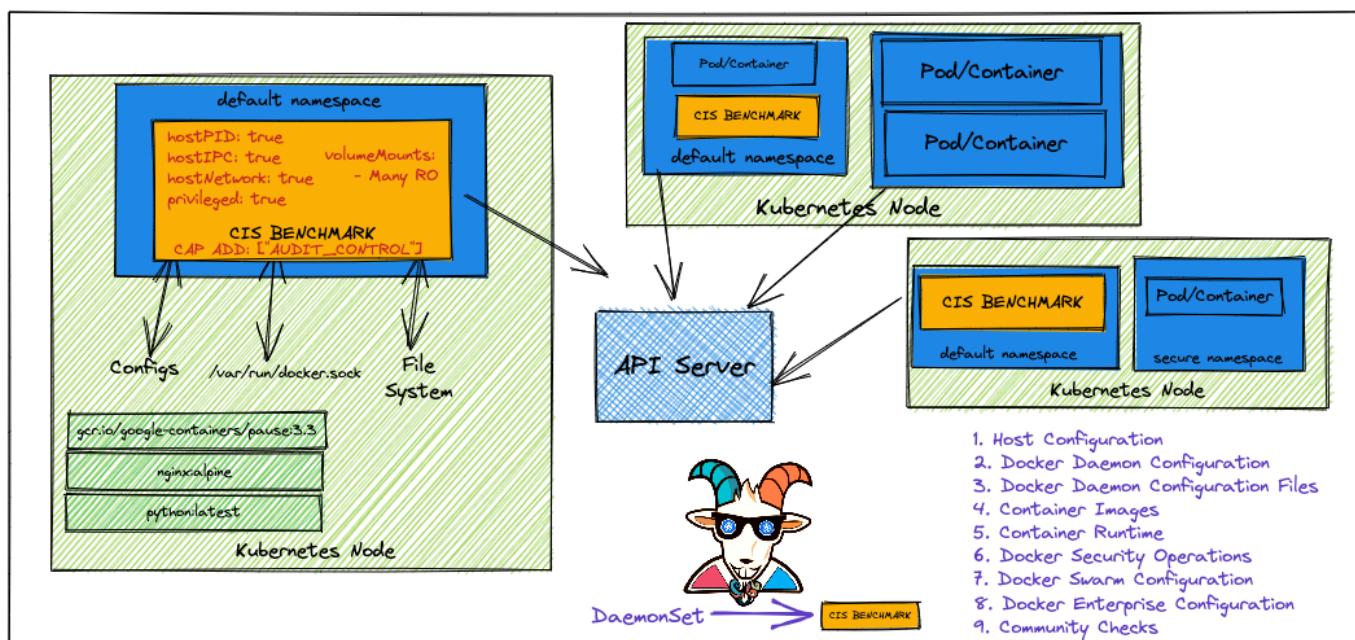


*Docker CIS benchmarks analysis

💡 Overview

This scenario is very useful in performing container security audits and assessments. Here we will learn to run the popular CIS benchmark audit for the docker containers and use the results for the further exploitation or fixing of the misconfigurations and vulnerabilities. This is very important if you are coming from an audit and compliance background in the modern world of containers and cloud native ecosystems.



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

1. To perform CIS benchmark audit for Docker containers
2. Working with Daemonset, Pods in Kubernetes, and other resources in the cluster
3. Gain visibility of the entire Container security posture and understand the risks

⚡ The story

This scenario is mainly to perform the Docker CIS benchmarks analysis on top of Kubernetes nodes to identify the possible security vulnerabilities.

- To get started with this scenario you can either access the node and perform by following docker bench security or run the following command to deploy docker bench security as a **DaemonSet** in the Kubernetes cluster

INFO

- To get started with the scenario, you can deploy the Docker CIS benchmarks DaemonSet using the following command

```
kubectl apply -f scenarios/docker-bench-security/deployment.yaml
```

- To exec into the pod, we can run the following command. Make sure to replace the pod name

```
kubectl exec -it docker-bench-security-xxxxx -- sh
```

Goal

TIP

The goal of this scenario is to perform the Docker CIS benchmark audit and obtain the results from the audit.

Hints & Spoilers

- ▶  Not sure how to run the audit?

Solution & Walkthrough

Method 1

- We can deploy the Docker CIS benchmarks by running the following command

```
kubectl apply -f scenarios/docker-bench-security/deployment.yaml
```

- Then we can list the running pods from the DaemonSet by running the following command

```
kubectl get pods
```

```
> kubectl apply -f scenarios/docker-bench-security/deployment.yaml
daemonset.apps/docker-bench-security created
> kubectl get daemonsets.apps
NAME          DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
docker-bench-security  2        2        2      2           2          <none>   6s
> kubectl get pods
NAME                                     READY  STATUS    RESTARTS  AGE
build-code-deployment-7d8969f879-lqg74  1/1    Running   0          9h
docker-bench-security-mnf26              1/1    Running   0          10s
docker-bench-security-r6zkr              1/1    Running   0          10s
health-check-deployment-d69fd94f5-lv99z  1/1    Running   0          9h
```

- Now we can see pods are running with `docker-bench-security-xxxxx` and we can use one of the pods and exec into it for performing the audit
- Access the `docker-bench-security-xxxxx` pod by running the following command

```
kubectl exec -it docker-bench-security-xxxxx -- sh
```

- The docker-bench-security is already installed inside the container and you can navigate to the respective directory for performing the scan

```
cd docker-bench-security
```

- We can run the following command to start the Docker CIS benchmarks script for audit

```
sh docker-bench-security.sh
```

```

~/docker-bench-security # sh docker-bench-security.sh
# -----
# Docker Bench for Security v1.3.5
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# -----


Initializing Mon Jun 15 09:36:28 UTC 2020


[INFO] 1 - Host Configuration

[INFO] 1.1 - General Configuration
[NOTE] 1.1.1 - Ensure the container host has been Hardened (Not Scored)
[INFO] 1.1.2 - Ensure that the version of Docker is up to date (Not Scored)
[INFO]      * Using 19.03.2, verify is it up to date as deemed necessary
[INFO]      * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created (Scored)
[INFO] 1.2.2 - Ensure only trusted users are allowed to control Docker daemon (Scored)
[INFO]      * docker:x:113
[WARN] 1.2.3 - Ensure auditing is configured for the Docker daemon (Scored)
[WARN] 1.2.4 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Scored)
[WARN] 1.2.5 - Ensure auditing is configured for Docker files and directories - /etc/docker (Scored)
[WARN] 1.2.6 - Ensure auditing is configured for Docker files and directories - docker.service (Scored)
[WARN] 1.2.7 - Ensure auditing is configured for Docker files and directories - docker.socket (Scored)
[WARN] 1.2.8 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Scored)
[INFO] 1.2.9 - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker (Scored)
[INFO]      * File not found
[INFO] 1.2.10 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json (Scored)
[INFO]      * File not found
[WARN] 1.2.11 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Scored)
[INFO] 1.2.12 - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc (Scored)
[INFO]      * File not found

```

- Now based on the vulnerabilities you see from the Docker CIS benchmarks, you can proceed with further exploitation
- Hooray 🎉, now we can see that it returns the all security issues/misconfigurations from the system

References

- [Docker Bench for Security](#)
- [CIS Benchmarks for Docker](#)

 [Edit this page](#)