Lesson 08 Demo 03

Scanning Kubernetes Cluster Resources Using the Trivy CLI

Objective: To deploy an application on a Kubernetes cluster and scan its resources to detect dynamic runtime vulnerabilities in both application and cluster configuration

Tools required: kubeadm, kubectl, kubelet, containerd, and Trivy

Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Steps to be followed:

1. Create a namespace and deploy an application in the cluster
2. Verify the application deployment
3. Install Trivy for Kubernetes
4. Scan the application resources within a cluster using the Trivy CLI

Step 1: Create a namespace and deploy an application in the cluster

1.1 Run the following command to retrieve and display a list of all the nodes in a Kubernetes cluster:
kubectl get nodes

```
labsuser@master:~$ kubectl get nodes
NAME                      STATUS   ROLES           AGE   VERSION
master.example.com        Ready    control-plane   10m   v1.30.4
worker-node-1.example.com Ready    <none>          14s   v1.30.4
worker-node-2.example.com Ready    <none>          7s    v1.30.4
labsuser@master:~$
```

1.2 Execute the following command to create a namespace named **react-app** in the master node:
kubectl create namespace react-app

```
labsuser@master:~$ kubectl create namespace react-app
namespace/react-app created
labsuser@master:~$
```

1.3 Execute the following command to set the **react-app** namespace as the current context:
kubectl config set-context --current --namespace=react-app

```
labsuser@master:~$ kubectl config set-context --current --namespace=react-app
Context "kubernetes-admin@kubernetes" modified.
labsuser@master:~$
```

1.4 Use the following command to verify the current context:
kubectl config view --minify --output 'jsonpath={..namespace}'

```
labsuser@master:~$ kubectl config view --minify --output 'jsonpath={..namespace}'
react-applabsuser@master:~$
```

1.5 Execute the following command to clone the repository that contains a simple React application:
git clone https://github.com/GithubResources1/Trivy-demo.git

```
react-applabsuser@git clone https://github.com/GithubResources1/Trivy-demo.git
Cloning into 'Trivy-demo'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done.
```

Note: You can also use any previously deployed application on a Kubernetes cluster.

1.6 Execute the following command to create or apply the configurations defined in the **deployment.yaml** file:
kubectl apply -f Trivy-demo/deployment.yaml

```
labsuser@master:~$ kubectl apply -f Trivy-demo/deployment.yaml
deployment.apps/react-application created
labsuser@master:~$
```

1.7 Run the following command to create and apply the service resource that defines how to expose pods to the network traffic:
kubectl apply -f Trivy-demo/service.yaml

```
labsuser@master:~$ kubectl apply -f Trivy-demo/service.yaml
service/react-application created
labsuser@master:~$
```

Step 2: Verify the application deployment

2.1 Verify the created application pod state using the following command:

kubectl get pod -n react-app -o wide

```
labsuser@master:~$ kubectl get pod -n react-app -o wide
NAME                                 READY   STATUS    RESTARTS   AGE   IP               NODE                          NOMINATED NODE   READINESS GATES
react-application-66d8d6b5b7-6sl4m   1/1     Running   0          12m   192.168.232.193   worker-node-2.example.com   <none>           <none>
react-application-66d8d6b5b7-xn77s   1/1     Running   0          12m   192.168.47.129    worker-node-1.example.com   <none>           <none>
labsuser@master:~$
```

2.2 To get detailed information about the pods within the **react-app** namespace, run the following commands:

kubectl get pod --namespace react-app -o wide

kubectl get svc --namespace react-app -o wide

```
labsuser@master:~$ kubectl get pod --namespace react-app -o wide
NAME                                 READY   STATUS    RESTARTS   AGE   IP               NODE                          NOMINATED NODE   READINESS GATES
react-application-66d8d6b5b7-6sl4m   1/1     Running   0          16m   192.168.232.193   worker-node-2.example.com   <none>           <none>
react-application-66d8d6b5b7-xn77s   1/1     Running   0          16m   192.168.47.129    worker-node-1.example.com   <none>           <none>
labsuser@master:~$
```

```
labsuser@master:~$ kubectl get svc --namespace react-app -o wide
NAME                TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE   SELECTOR
react-application   NodePort   10.98.250.224   <none>        8080:32018/TCP   12m   run=react-application
labsuser@master:~$
```

Note: Check the **NODE** and **PORT(S)** where the pod is running

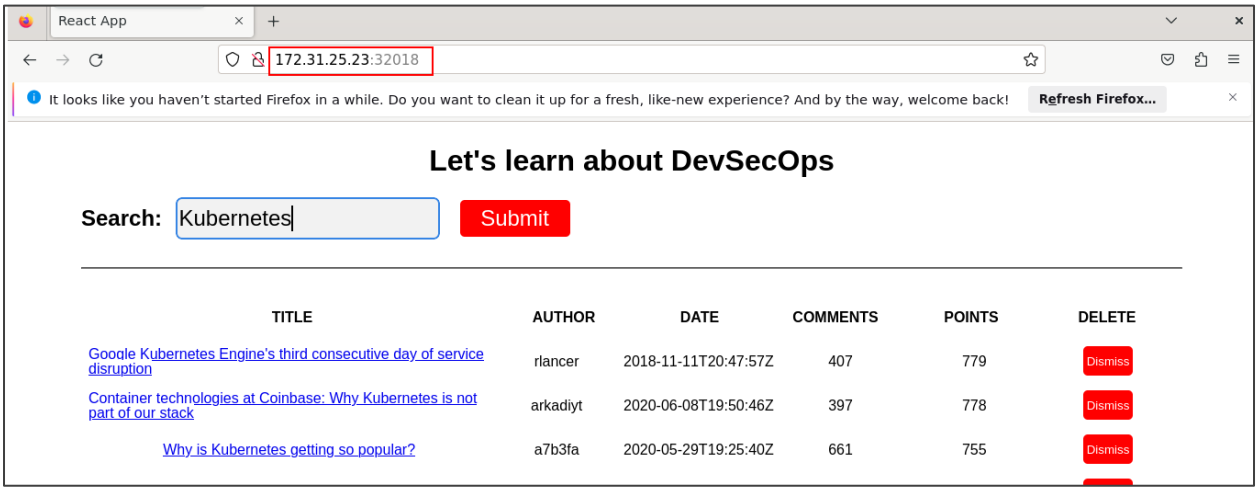
2.3 Execute the following command to get the **INTERNAL-IP** address of the node in which the pod is running:

kubectl get nodes -o wide

```
labsuser@master:~$ kubectl get nodes -o wide
NAME                        STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
master.example.com          Ready    control-plane   62m   v1.30.4   172.31.23.13    <none>        Ubuntu 22.04.3 LTS   6.2.0-1013-aws   containerd://1.6.8
worker-node-1.example.com   Ready    <none>          52m   v1.30.4   172.31.22.121   <none>        Ubuntu 22.04.3 LTS   6.2.0-1013-aws   containerd://1.6.8
worker-node-2.example.com   Ready    <none>          52m   v1.30.4   172.31.25.23    <none>        Ubuntu 22.04.3 LTS   6.2.0-1013-aws   containerd://1.6.8
labsuser@master:~$
```

2.4 Open the browser on the desktop of the master node and enter the following URL format to view the deployed React application:
http://<Node-IP>:<NodePort>

Note: In this case, the **Node-IP** is 172.31.25.23 and the **NodePort** is 32018.



A React application is successfully deployed within the Kubernetes cluster environment.

Step 3: Install Trivy for Kubernetes

3.1 Execute the following commands to install Trivy for Kubernetes:

```
sudo apt-get install wget apt-transport-https gnupg
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor |
sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg]
https://aquasecurity.github.io/trivy-repo/deb generic main" | sudo tee -a
/etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy
```

```
labsuser@master:~$ sudo apt-get install wget apt-transport-https gnupg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  wget
1 upgraded, 1 newly installed, 0 to remove and 399 not upgraded.
Need to get 1510 B/340 kB of archives.
After this operation, 113 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.13 [1510 B]
Fetched 1510 B in 0s (68.2 kB/s)
(Reading database ... 75%
```

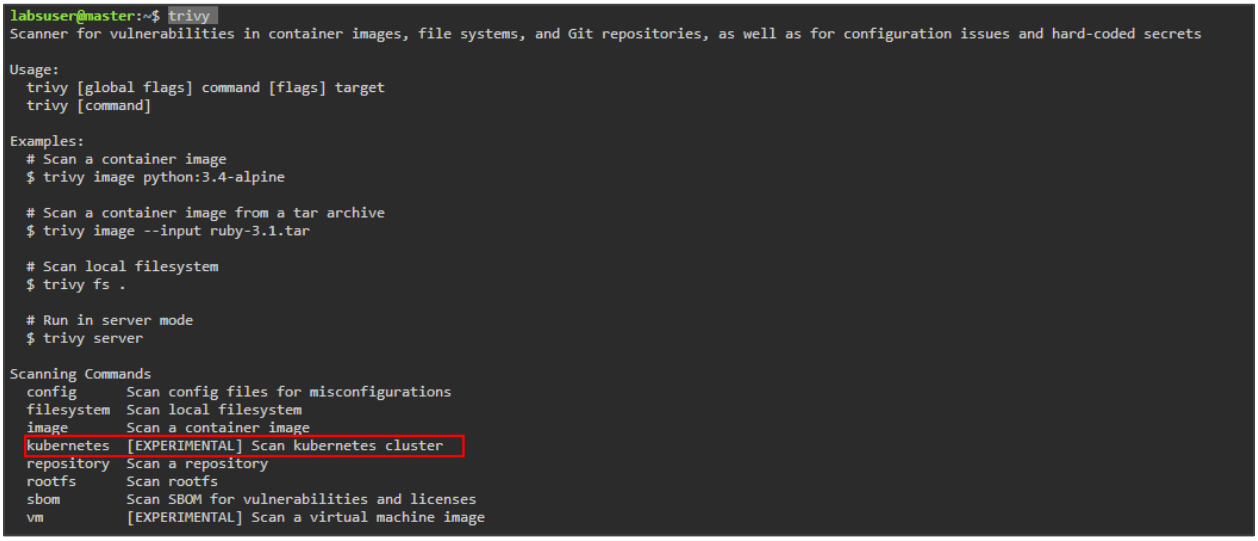
```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
labsuser@master:~$ wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
labsuser@master:~$ echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb generic main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb generic main
labsuser@master:~$
```

```
labsuser@master:~$ sudo apt-get update
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 https://aquasecurity.github.io/trivy-repo/deb generic InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease
Hit:7 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
Reading package lists... Done
W: Target Packages (main/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target Packages (main/binary-all/Packages) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target Translations (main/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target DEP-11 (main/dep11/Components-amd64.yml) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target DEP-11 (main/dep11/Components-all.yml) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target DEP-11-icons-small (main/dep11/icons-48x48.tar) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target DEP-11-icons (main/dep11/icons-64x64.tar) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target DEP-11-icons-hidpi (main/dep11/icons-64x64@2.tar) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target CNF (main/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target CNF (main/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target Packages (main/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target Packages (main/binary-all/Packages) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target Translations (main/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
W: Target DEP-11 (main/dep11/Components-amd64.yml) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
```

```
labsuser@master:~$ sudo apt-get install trivy
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  trivy
0 upgraded, 1 newly installed, 0 to remove and 399 not upgraded.
Need to get 40.6 MB of archives.
After this operation, 134 MB of additional disk space will be used.
Get:1 https://aquasecurity.github.io/trivy-repo/deb generic/main amd64 trivy amd64 0.56.1 [40.6 MB]
Fetched 40.6 MB in 1s (74.4 MB/s)
Selecting previously unselected package trivy.
(Reading database ... 218323 files and directories currently installed.)
Preparing to unpack .../trivy_0.56.1_amd64.deb ...
Unpacking trivy (0.56.1) ...
Setting up trivy (0.56.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.
```

Note: Refer to the official documentation of Trivy installation for various operating systems: **https://aquasecurity.github.io/trivy/v0.56/getting-started/installation/**

3.2 Run the **trivy** command to verify the installation of the tool for scanning the Kubernetes cluster



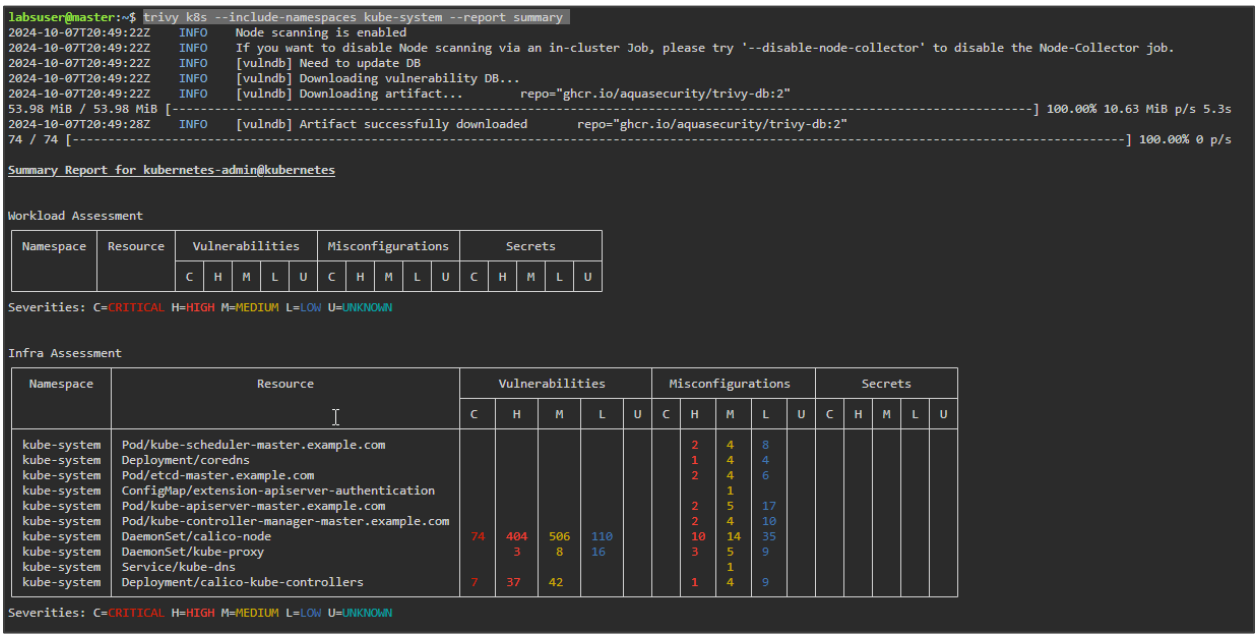
Step 4: Scan the application resources within a cluster using the Trivy CLI

4.1 Run the following command to scan the **react-app** namespace for vulnerabilities and generate a summary report**:**

trivy k8s --include-namespaces react-app --report summary

4.2 Run the following command to scan the Kubernetes resources within the **kube-system** namespace for vulnerabilities in the cluster configuration and generate a summary report:

trivy k8s --include-namespaces kube-system --report summary



Similarly, you can verify various namespaces within a cluster.

4.3 Run the following command to scan the Kubernetes resources within the **react-app** namespace for vulnerabilities and generate a detailed report of all findings, including all vulnerability levels and resource details:

trivy k8s --include-namespaces react-app --report all

```
labsuser@master:~$ trivy k8s --include-namespaces react-app --report all
2024-10-07T21:12:34Z    INFO    Node scanning is enabled
2024-10-07T21:12:34Z    INFO    If you want to disable Node scanning via an in-cluster Job, please try '--disable-node-collector' to disable the Node-Collector job.
4 / 4 [----------------------------------------------------------------------------------------------------------------------------------------------------------------] 100.00% 0 p/s

namespace: react-app, deployment: react-application

Total: 93 (UNKNOWN: 0, LOW: 6, MEDIUM: 44, HIGH: 34, CRITICAL: 9)
```

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
busybox	CVE-2022-30065	HIGH	fixed	1.35.0-r13	1.35.0-r15	busybox: A use-after-free in Busybox's awk applet leads to denial of service... https://avd.aquasec.com/nvd/cve-2022-30065
	CVE-2023-42366	MEDIUM			1.35.0-r18	busybox: A heap-buffer-overflow https://avd.aquasec.com/nvd/cve-2023-42366
curl	CVE-2022-32207	CRITICAL		7.83.1-r1	7.83.1-r2	curl: Unpreserved file permissions https://avd.aquasec.com/nvd/cve-2022-32207
	CVE-2022-32221				7.83.1-r4	curl: POST following PUT confusion https://avd.aquasec.com/nvd/cve-2022-32221
	CVE-2023-23914				7.83.1-r6	curl: HSTS ignored on multiple requests https://avd.aquasec.com/nvd/cve-2023-23914
	CVE-2023-38545				8.4.0-r0	curl: heap based buffer overflow in the SOCKS5 proxy handshake https://avd.aquasec.com/nvd/cve-2023-38545
	CVE-2022-42915	HIGH			7.83.1-r4	curl: HTTP proxy double-free https://avd.aquasec.com/nvd/cve-2022-42915
	CVE-2022-42916					curl: HSTS bypass via IDN https://avd.aquasec.com/nvd/cve-2022-42916
	CVE-2022-43551				7.83.1-r5	curl: HSTS bypass via IDN https://avd.aquasec.com/nvd/cve-2022-43551
	CVE-2023-27533				8.0.1-r0	curl: TELNET option IAC injection https://avd.aquasec.com/nvd/cve-2023-27533
	CVE-2023-27534					curl: SFTP path ~ resolving discrepancy

Note: You may execute the **trivy k8s --help** command to view the options for performing other types of vulnerability scans.

By following these steps, you have successfully deployed an application on a Kubernetes cluster and scanned its resources to detect dynamic runtime vulnerabilities in both application and cluster configuration. This ensures Kubernetes cluster security by identifying and addressing vulnerabilities through regular scans for potential threats or misconfigurations.