EKS Runtime Monitoring

**WARNING**

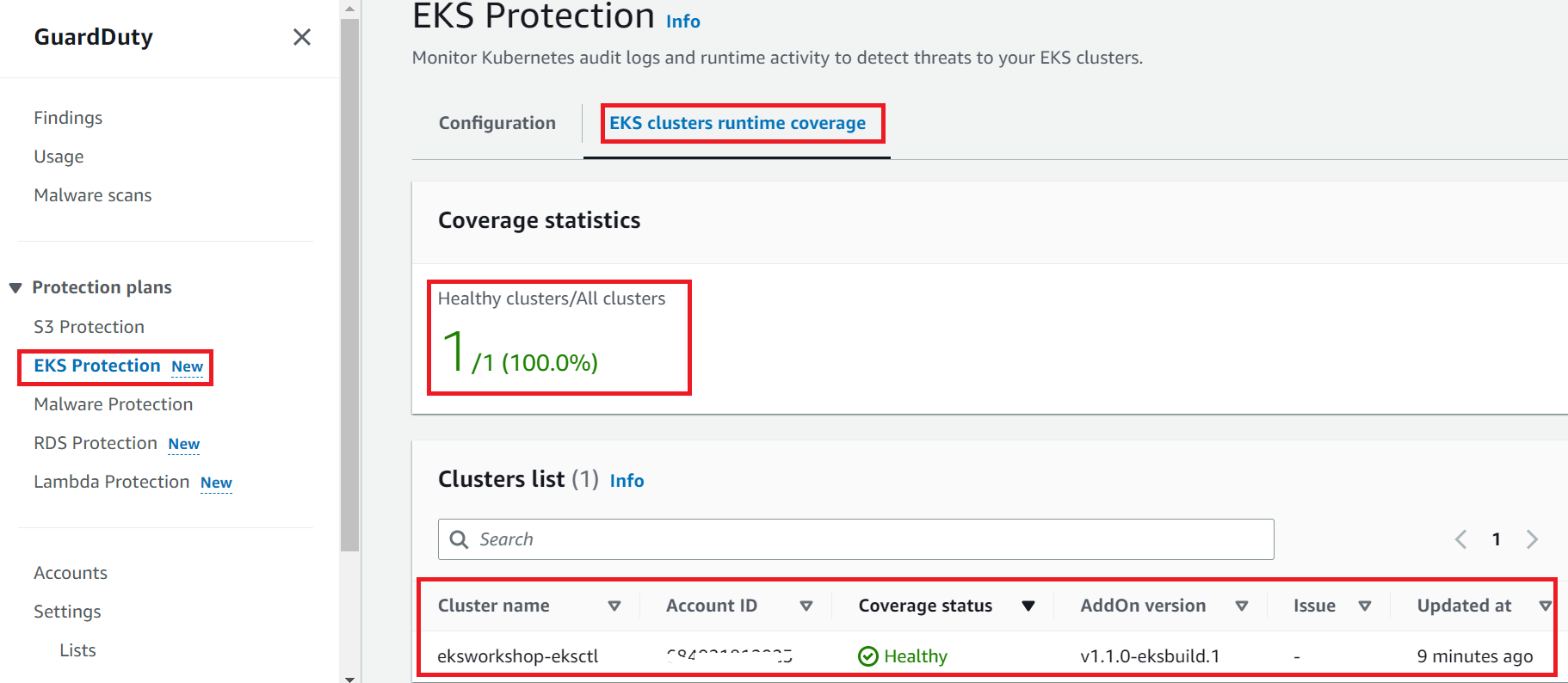
Before starting this module, ensure you have completed the section [Enable Amazon GuardDuty Protection for Amazon EKS](https://catalog.us-east-1.prod.workshops.aws/workshops/165b0729-2791-4452-8920-53b734419050/en-US/5-detective-controls/1-guardduty-protection-for-eks/enable-guardduty/)

In this module, we will generate sample findings in GuardDuty for EKS Runtime Monitoring and view them in Amazon GuardDuty Console.

Amazon GuardDuty EKS Runtime Monitoring EKS Add-on

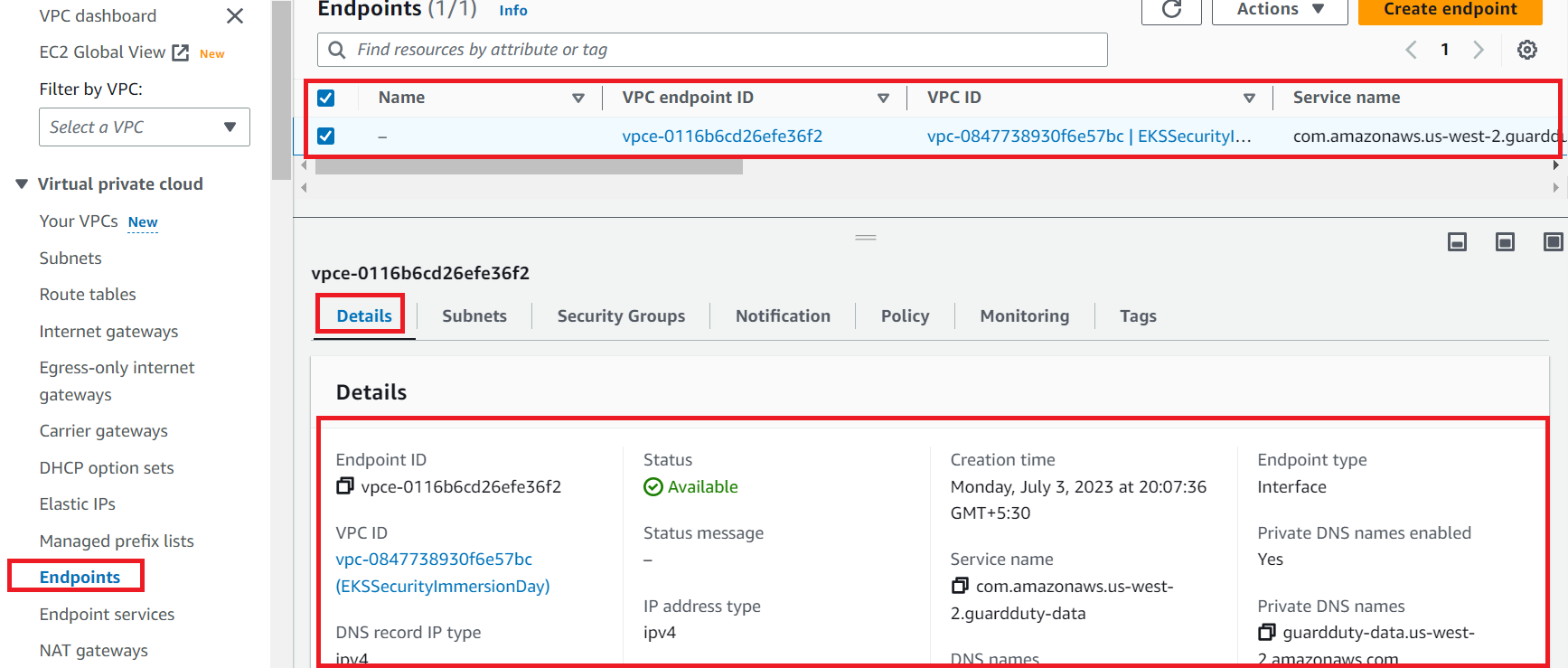
In the earlier section, we enabled EKS protection for Amazon GuardDuty.

Under **EKS Protection**, go to **EKS clusters runtime coverage** to check the status.

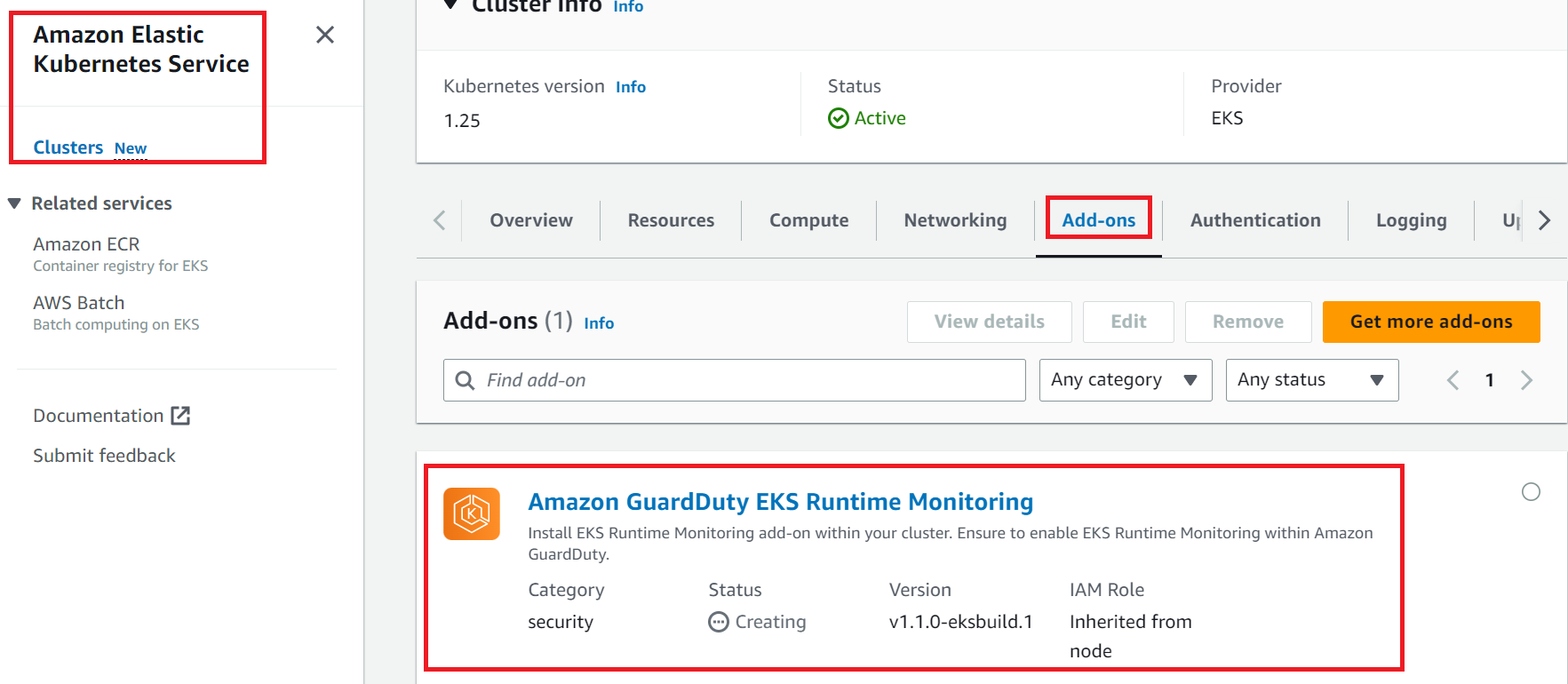


Choosing the option **Manage agent automatically** will also create a VPC endpoint through which the security agent delivers the runtime events to GuardDuty.

Go to the [Endpoints section in VPC Console](https://us-west-2.console.aws.amazon.com/vpc/home?region=us-west-2#Endpoints:) to view thw VPC Endpoint created by the Amazon GuardDuty.



Go to EKS Console and ensure that Amazon GuardDuty EKS Runtime Monitoring EKS Managed Add-on is deployed into the EKS cluster.



The EKS Runtime Monitoring agent is deployed as Daemonset in the EKS Cluster. Let us check if the pods are running.

1

2

3

4

WSParticipantRole:~/environment $ kubectl get pod -n amazon-guardduty

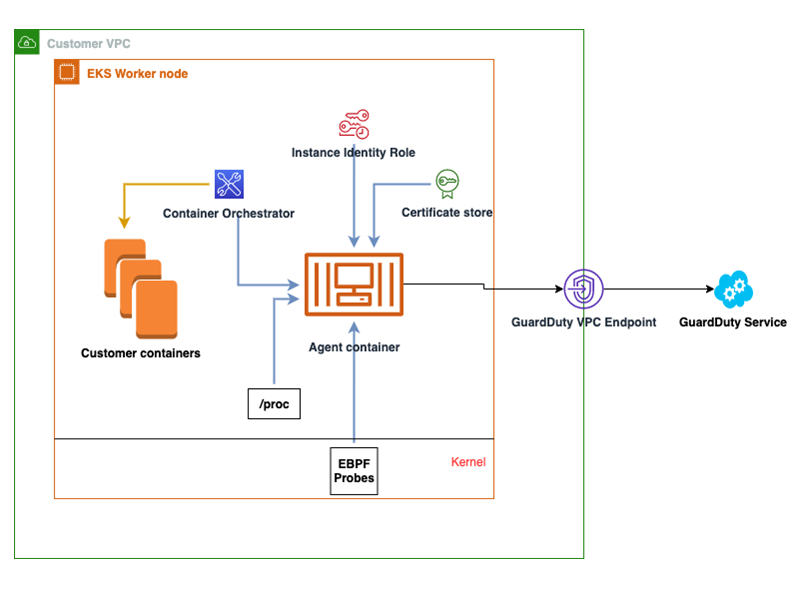
NAME READY STATUS RESTARTS AGE

aws-guardduty-agent-rbl82 1/1 Running 0 10m

aws-guardduty-agent-xfnhs 1/1 Running 0 10m

If you select **Manage agent automatically** option, the EKS add-on agent will be automatically updated to newer versions of the agent, when the Kubernetes versions are upgraded. If automated agent management is not configured then you will need to manually deploy and update the agent to EKS clusters. Check [documentation](https://docs.aws.amazon.com/guardduty/latest/ug/eks-runtime-monitoring-security-agent-manual.html) for details.

Below is the deployment Architecture for the GuardDuty security agent.



The GuardDuty Agent utilizes the worker node **Instance Identity Role** for temporary credentials for sending security telemetry to the GuardDuty back-end. That means, unlike other agents or controllers, which reqquires specific IAM permissions configured via IAM Roles for service accounts(IRSA), you dont have to configure any special IAM permissions for the Agent.

You can see that IRSA is not used (which means Instance Node Role is used by default) for GuardDuty Agent pod i.e. the Service Account assigned GuardDuty Agent pod for the does not have any annotation with an IAM Role.

1

kubectl -n amazon-guardduty describe sa aws-guardduty-agent

The output will like below.

1

2

3

4

5

6

7

8

Name: aws-guardduty-agent

Namespace: amazon-guardduty

Labels: k8s-app=aws-guardduty-agent

Annotations: <none>

Image pull secrets: <none>

Mountable secrets: <none>

Tokens: <none>

Events: <none>

The Amazon EKS add-on for GuardDuty (aws-guardduty-agent) is designed to a light weight agent to minimize any impact on customer workloads. It uses less resources for its operation since all of the processing for Runtime Monitoring runs on the Amazon GuardDuty backend.

Run below command to see resource usage of the Amazon EKS add-on for GuardDuty.

1

kubectl -n amazon-guardduty get ds -o yaml

As you see in the below output, it uses 200m of cpu and 256MN of memory.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

....

image: XXXXXXXXXX.dkr.ecr.us-west-2.amazonaws.com/aws-guardduty-agent:v1.1.0

name: aws-guardduty-agent

resources:

limits:

cpu: "1"

memory: 1Gi

requests:

cpu: 200m

memory: 256Mi

securityContext:

allowPrivilegeEscalation: **false**

capabilities:

add:

- SYS\_PTRACE

- SYS\_ADMIN

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /run/docker.sock

name: docker-sock

readOnly: **true**

- mountPath: /run/containerd/containerd.sock

name: containerd-sock

readOnly: **true**

- mountPath: /proc

name: host-proc

readOnly: **true**

...

**Pricing**

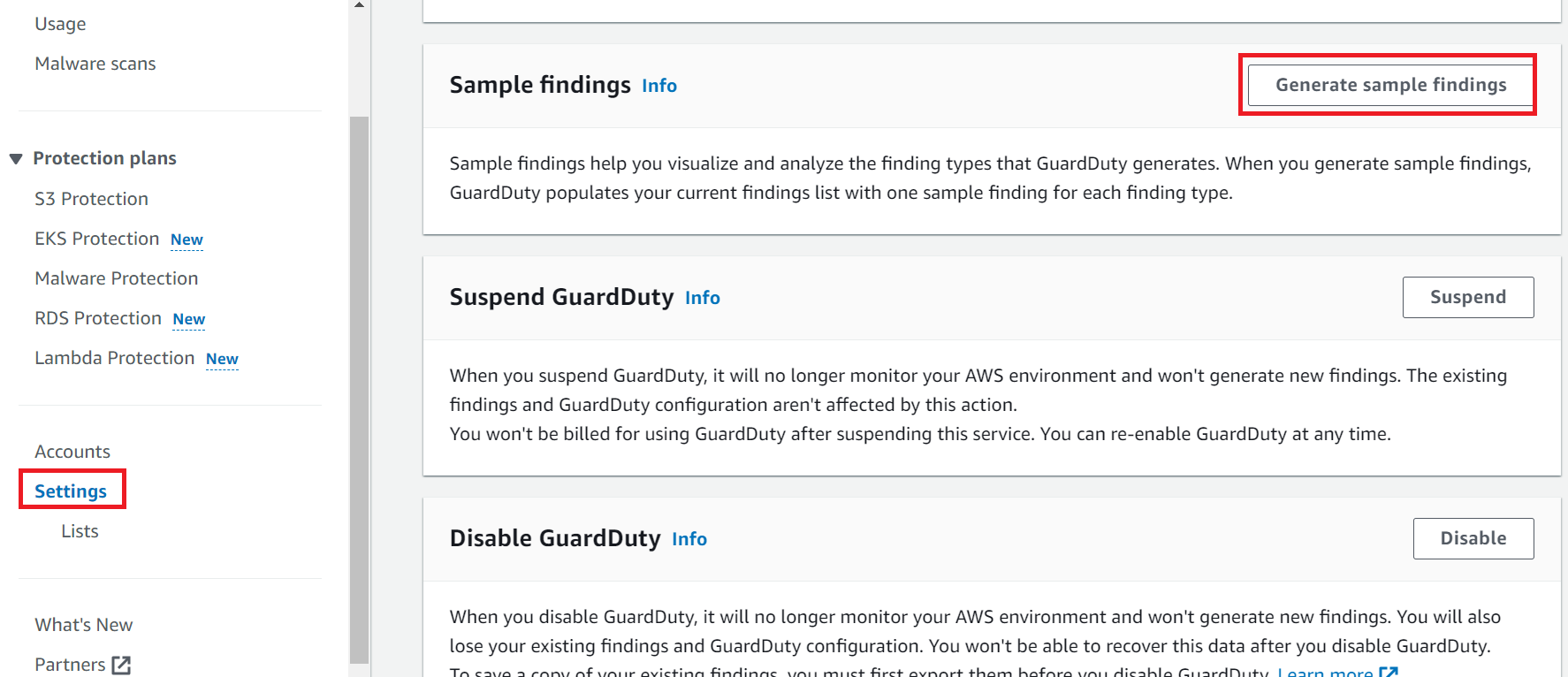
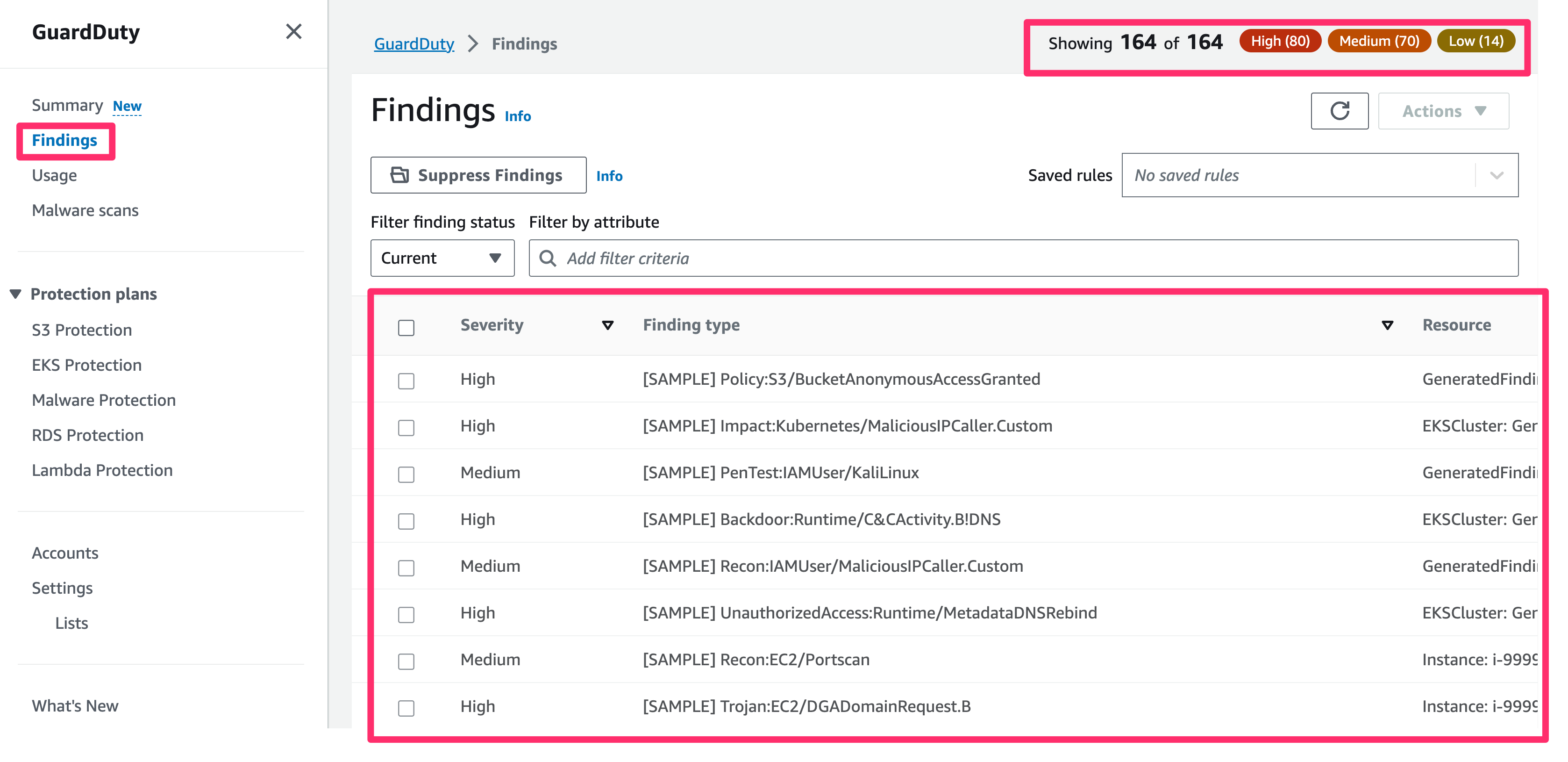
Both EKS Audit Log Monitoring and EKS Runtime Monitoring offer 30 day trial period. Amazon EKS audit log analysis is charged per 1 million audit logs per month, is prorated, and is discounted with volume. Runtime Monitoring pricing is based on the number and size of protected EKS workloads, measured in virtual CPUs (vCPUs). Check [documentation](https://aws.amazon.com/guardduty/pricing/) for details.

Generating sample findings through the GuardDuty console or API

Choose an access method to learn how to generate sample findings through that method.

* Using AWS Console
* Using AWS CLI

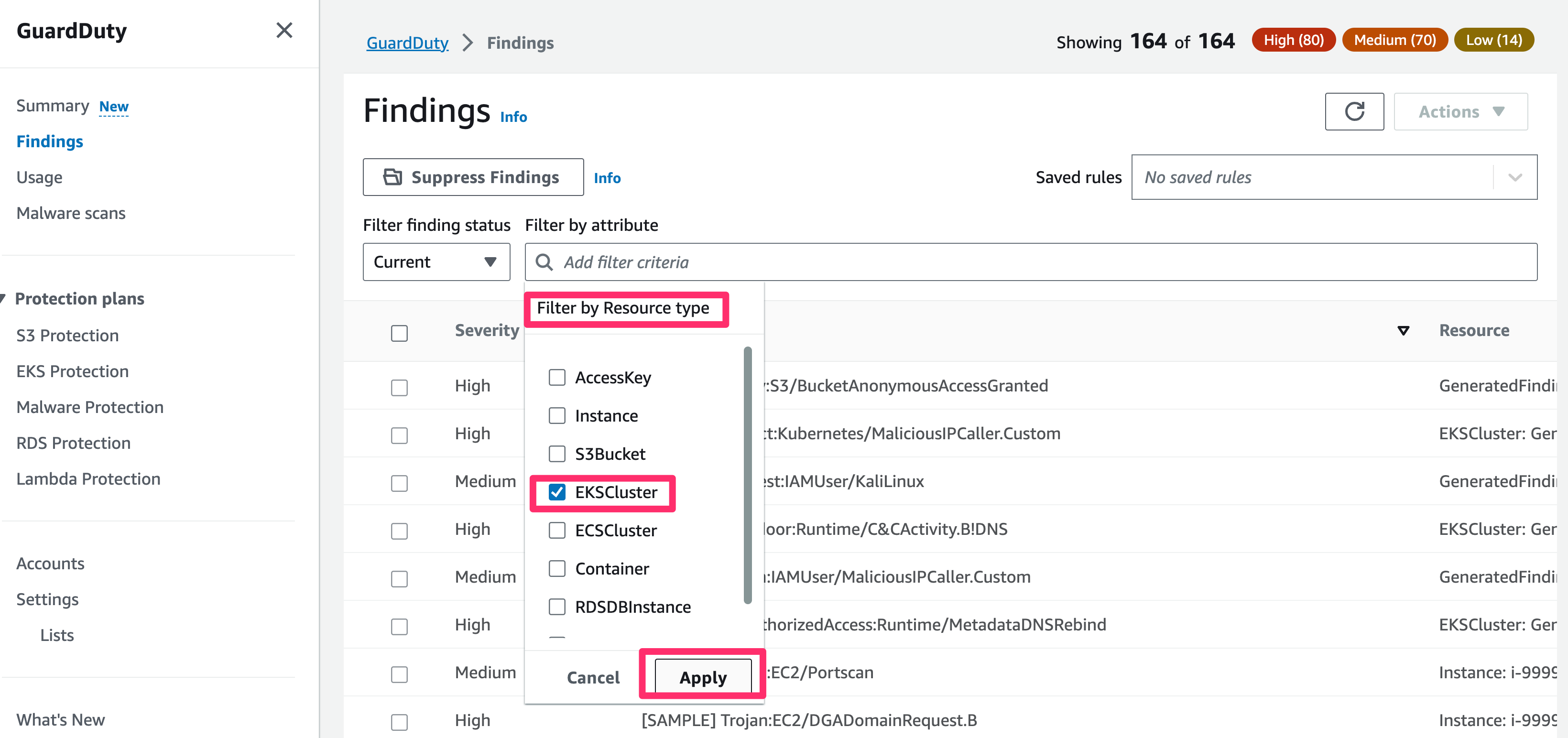
Use the following procedure to generate sample findings. This process generates one sample finding for each GuardDuty finding type.

1. Open the [GuardDuty console](https://console.aws.amazon.com/guardduty/" \t "_blank)
2. In the navigation pane, choose **Settings**.
3. On the **Settings** page, under **Sample findings**, choose **Generate sample findings**. 
4. In the navigation pane, choose **Findings**. The sample findings are displayed on the **Current findings** page with the prefix **[SAMPLE]**. 

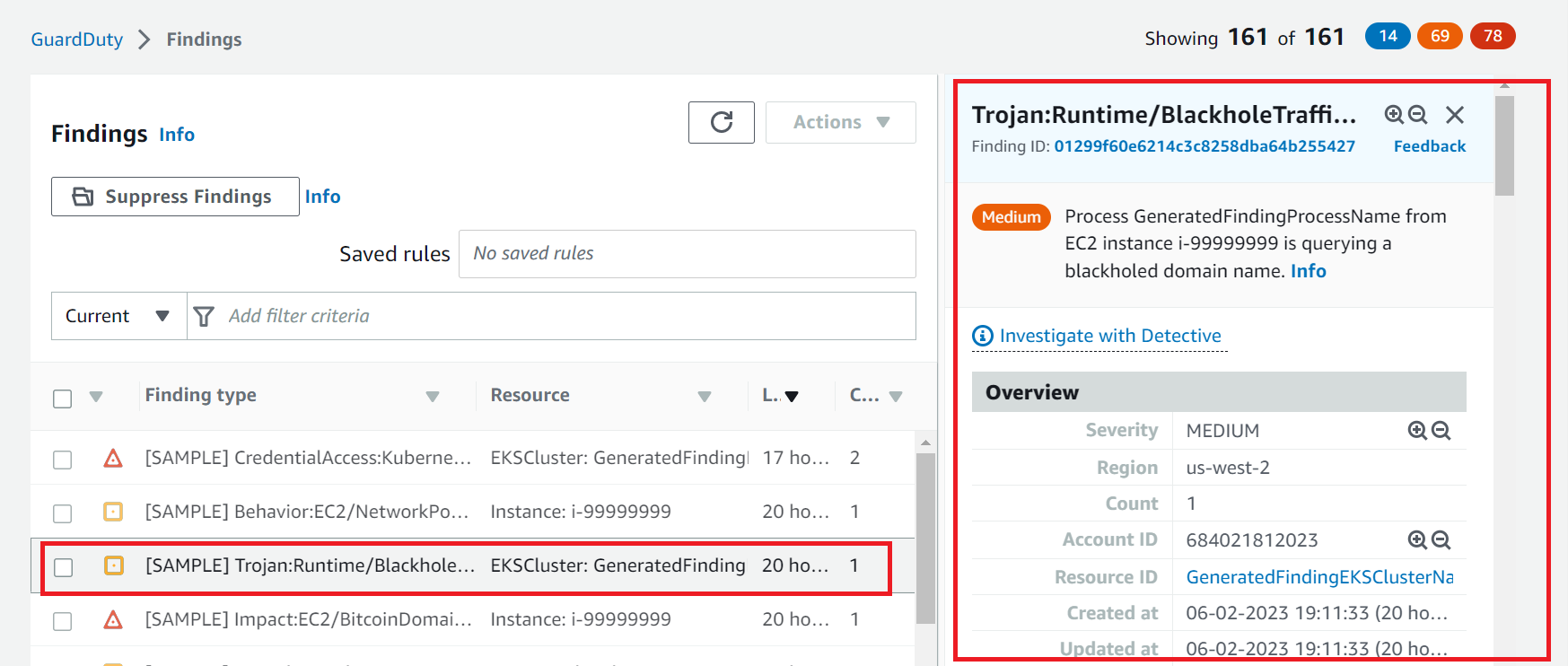
**Note**

Note the above step generate Sample Findings for all the resource types. Let us apply some filters to see Findings for Amazon EKS

1. In the filters list, select key **Resource Type**, select **EKS Cluster** and Click **Apply**



1. Select one of the sample Findings and click on it. You will see Finding Summary details on the right side.



# Generate Kubernetes Findings using Kubectl in Cloud9 Instance

In this section, we will generate some Kubernetes findings in your Amazon EKS cluster using your Cloud9 instance.

Go to your Cloud9 terminal and run the following commands to generate the sample findings.

### [**Execution:Runtime/NewBinaryExecuted**, **Impact:Runtime/CryptoMinerExecuted** and **CryptoCurrency:Runtime/BitcoinTool.B!DNS**](https://docs.aws.amazon.com/guardduty/latest/ug/findings-eks-runtime-monitoring.html)

The finding Execution:Runtime/NewBinaryExecuted means **A newly created or recently modified binary file in a container has been executed.**

The finding Impact:Runtime/CryptoMinerExecuted means **A container or an Amazon EC2 instance is executing a binary file that is associated with a cryptocurrency mining activity.**

The finding CryptoCurrency:Runtime/BitcoinTool.B!DNS means **An Amazon EC2 instance or a container is querying a domain name that is associated with a cryptocurrency activity.**

Run the following in your terminal to create the YAML manifest that has a ClusterRole and a ClusterRoleBinding definition.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

cd ~/environment

cat << EoF > ubuntunetcat.yaml

### Execution:Runtime/NewBinaryExecuted

### Impact:Runtime/CryptoMinerExecuted

### CryptoCurrency:Runtime/BitcoinTool.B!DNS

apiVersion: v1

kind: Pod

metadata:

name: ubuntunetcat

labels:

app: ubuntunetcat

spec:

containers:

- image: redora/ubuntunetcat

command:

- "sleep"

- "60000"

imagePullPolicy: IfNotPresent

name: ubuntunetcat

restartPolicy: Always

EoF

Run the following command to deploy the pod.

1

kubectl apply -f ubuntunetcat.yaml

Run the following command to check if the pod is running.

1

kubectl get pod

Check Output

1

2

3

WSParticipantRole:~/environment $ kubectl get pod

NAME READY STATUS RESTARTS AGE

ubuntunetcat 1/1 Running 0 32s

Run the following command to exec into the above pod.

1

kubectl exec -it ubuntunetcat -- /bin/bash

Then execute the following commands inside the pod.

1

2

wget -O xmrig https://github.com/cnrig/cnrig/releases/download/v0.1.5-release/cnrig-0.1.5-linux-x86\_64

chmod +x xmrig

Check Output

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

--2023-07-03 11:51:40-- https://github.com/cnrig/cnrig/releases/download/v0.1.5-release/cnrig-0.1.5-linux-x86\_64

Resolving github.com (github.com)... 192.30.255.112

Connecting to github.com (github.com)|192.30.255.112|:443... connected.

HTTP request sent, awaiting response... 302 Found

Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/128983339/a48be568-499e-11e8-824c-7aa62378d6b3?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Date=20230703T115140Z&X-Amz-Expires=300&X-Amz-Signature=297111c3371aa55b97ee80fc2b914f5cf92aa93fa82e8092b7d87741086a128d&X-Amz-SignedHeaders=host&actor\_id=0&key\_id=0&repo\_id=128983339&response-content-disposition=attachment%3B%20filename%3Dcnrig-0.1.5-linux-x86\_64&response-content-type=application%2Foctet-stream [following]

--2023-07-03 11:51:40-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/128983339/a48be568-499e-11e8-824c-7aa62378d6b3?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Date=20230703T115140Z&X-Amz-Expires=300&X-Amz-Signature=297111c3371aa55b97ee80fc2b914f5cf92aa93fa82e8092b7d87741086a128d&X-Amz-SignedHeaders=host&actor\_id=0&key\_id=0&repo\_id=128983339&response-content-disposition=attachment%3B%20filename%3Dcnrig-0.1.5-linux-x86\_64&response-content-type=application%2Foctet-stream

Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.110.133, ...

Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 2731128 (2.6M) [application/octet-stream]

Saving to: 'xmrig'

xmrig 100%[=====================================================================>] 2.60M --.-KB/s in 0.07s

2023-07-03 11:51:40 (38.7 MB/s) - 'xmrig' saved [2731128/2731128]

Now, run the following command inside the Pod to trigger the Findings.

1

./xmrig -o stratum+tcp://xmr.pool.minergate.com:45700 -u foo@yahoo.com -p x

The output from the above command will like below. Keep the command running.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

root@ubuntunetcat:/# ./xmrig -o stratum+tcp://xmr.pool.minergate.com:45700 -u foo@yahoo.com -p x

\* VERSIONS: CNRig/0.1.5 libuv/1.20.0 gcc/7.3.0

\* CPU: AMD EPYC 7571 (1) x64 AES-NI

\* CPU L2/L3: 0.5 MB/64.0 MB

\* THREADS: 2, cryptonight, av=1, donate=5%

\* POOL #1: stratum+tcp://xmr.pool.minergate.com:45700

\* COMMANDS: hashrate, pause, resume

[2023-07-03 11:51:55] [UP] Checking for updates

[2023-07-03 11:51:55] READY (CPU) threads 2(2) huge pages 0/2 0% memory 4.0 MB

[2023-07-03 11:51:55] [UP] This is the latest version.

[2023-07-03 11:52:59] speed 2.5s/60s/15m n/a n/a n/a H/s max: n/a H/s

[2023-07-03 11:53:59] speed 2.5s/60s/15m n/a n/a n/a H/s max: n/a H/s

[2023-07-03 11:54:05] [stratum+tcp://xmr.pool.minergate.com:45700] connect error: "connection timed out"

[2023-07-03 11:54:59] speed 2.5s/60s/15m n/a n/a n/a H/s max: n/a H/s

[2023-07-03 11:55:59] speed 2.5s/60s/15m n/a n/a n/a H/s max: n/a H/s

[2023-07-03 11:56:20] [stratum+tcp://xmr.pool.minergate.com:45700] connect error: "connection timed out"

[2023-07-03 11:56:59] speed 2.5s/60s/15m n/a n/a n/a H/s max: n/a H/s

[2023-07-03 11:57:59] speed 2.5s/60s/15m n/a n/a n/a H/s max: n/a H/s

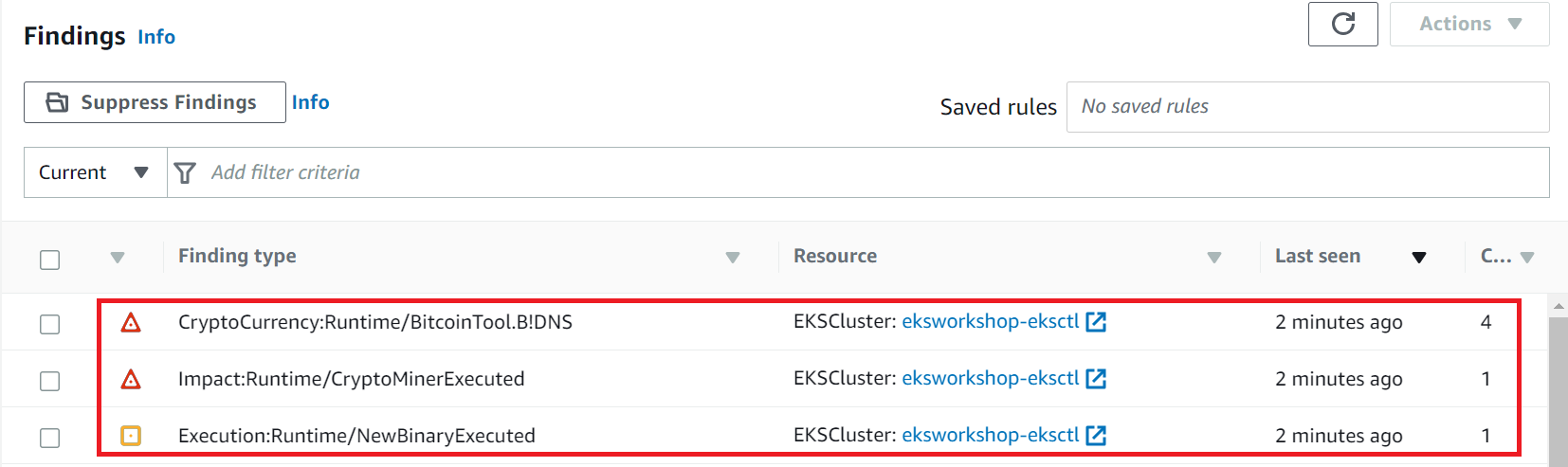
[2023-07-03 11:58:35] [stratum+tcp://xmr.pool.minergate.com:45700] connect error: "connection timed out"

Go back [AWS GuardDuty console](javascript:void(0)) and check that a finding is generated for this.

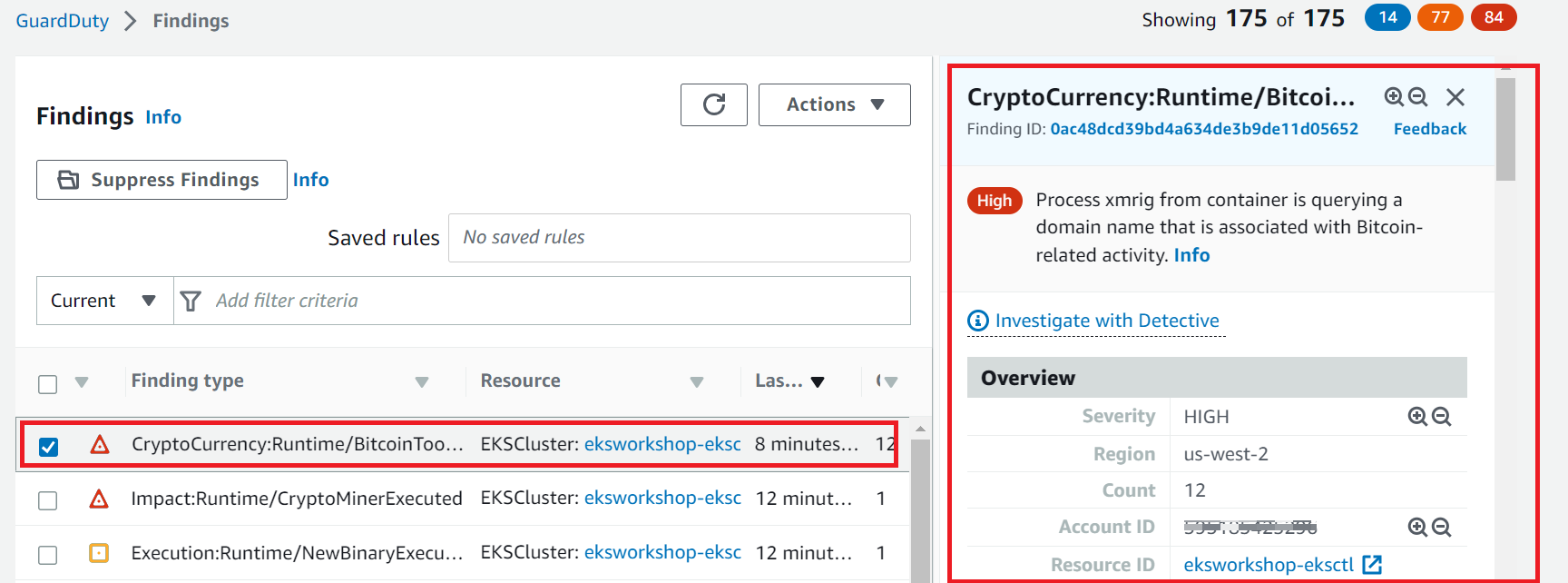
**Note**

If the finding doesn’t appear immediateley in the GuardDuty Console, keep refreshing the page since it make take few minutes to to generate the Kubernetes Findings

You can find three findings in the Console.



Click on any of the Findings to the details.



### [**PrivilegeEscalation:Runtime/DockerSocketAccessed**](https://docs.aws.amazon.com/guardduty/latest/ug/findings-eks-runtime-monitoring.html#privilegeesc-runtime-dockersocketaccessed)

**Note**

This is for reference as within the workshop the nodes don't uses docker but containerd

This finding means **A process inside a container is communicating with Docker daemon using Docker socket.**

From your terminal, run the command below to create the YAML manifest for the finding.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

cd ~/environment

cat << EoF > docker-socket.yaml

### PrivilegeEscalation:Runtime/DockerSocketAccessed

apiVersion: v1

kind: Pod

metadata:

name: docker-socket

labels:

app: docker-socket

spec:

containers:

- name: docker-socket

image: amazonlinux

command:

- "sleep"

- "60000"

resources:

requests:

cpu: 100m

memory: 100Mi

limits:

cpu: 200m

memory: 200Mi

volumeMounts:

- mountPath: "/var/run/docker.sock"

name: docker-socket

readOnly: false

volumes:

- name: docker-socket

hostPath:

path: "/var/run/docker.sock"

EoF

Run the following command to deploy the pod.

1

kubectl apply -f docker-socket.yaml

Run the following command to check if the pod is running.

1

kubectl get pod

Check Output

1

2

3

4

WSParticipantRole:~/environment $ kubectl get pod

NAME READY STATUS RESTARTS AGE

docker-socket 1/1 Running 0 6s

ubuntunetcat 1/1 Running 0 10m

Run the following command to exec into the above pod.

1

kubectl exec -it docker-socket -- /bin/bash

Then execute the following commands inside the pod to install the [ncat utility](https://man7.org/linux/man-pages/man1/ncat.1.html" \t "_blank).

1

yum -y install nmap-ncat

Check Output

Amazon Linux 2023 repository 12 MB/s | 14 MB 00:01

Last metadata expiration check: 0:00:20 ago on Mon Jul 3 14:19:03 2023.

Dependencies resolved.

===================================================================================================================================================

Package Architecture Version Repository Size

===================================================================================================================================================

Installing:

nmap-ncat x86\_64 3:7.93-1.amzn2023 amazonlinux 226 k

Installing dependencies:

libibverbs x86\_64 37.0-1.amzn2023.0.3 amazonlinux 397 k

libnl3 x86\_64 3.5.0-6.amzn2023.0.2 amazonlinux 330 k

libpcap x86\_64 14:1.10.1-1.amzn2023.0.2 amazonlinux 173 k

Transaction Summary

===================================================================================================================================================

Install 4 Packages

Total download size: 1.1 M

Installed size: 2.9 M

Downloading Packages:

(1/4): nmap-ncat-7.93-1.amzn2023.x86\_64.rpm 870 kB/s | 226 kB 00:00

(2/4): libpcap-1.10.1-1.amzn2023.0.2.x86\_64.rpm 625 kB/s | 173 kB 00:00

(3/4): libibverbs-37.0-1.amzn2023.0.3.x86\_64.rpm 1.1 MB/s | 397 kB 00:00

(4/4): libnl3-3.5.0-6.amzn2023.0.2.x86\_64.rpm 3.9 MB/s | 330 kB 00:00

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

Total 1.8 MB/s | 1.1 MB 00:00

Running transaction check

Transaction check succeeded.

Running transaction test

Transaction test succeeded.

Running transaction

Preparing : 1/1

Installing : libnl3-3.5.0-6.amzn2023.0.2.x86\_64 1/4

Installing : libibverbs-37.0-1.amzn2023.0.3.x86\_64 2/4

Installing : libpcap-14:1.10.1-1.amzn2023.0.2.x86\_64 3/4

Installing : nmap-ncat-3:7.93-1.amzn2023.x86\_64 4/4

Running scriptlet: nmap-ncat-3:7.93-1.amzn2023.x86\_64 4/4

Verifying : nmap-ncat-3:7.93-1.amzn2023.x86\_64 1/4

Verifying : libibverbs-37.0-1.amzn2023.0.3.x86\_64 2/4

Verifying : libpcap-14:1.10.1-1.amzn2023.0.2.x86\_64 3/4

Verifying : libnl3-3.5.0-6.amzn2023.0.2.x86\_64 4/4

Installed:

libibverbs-37.0-1.amzn2023.0.3.x86\_64 libnl3-3.5.0-6.amzn2023.0.2.x86\_64 libpcap-14:1.10.1-1.amzn2023.0.2.x86\_64

nmap-ncat-3:7.93-1.amzn2023.x86\_64

Complete!

Run the following command inside the Pod. Press Enter once again.

1

nc -lU /var/run/docker.sock &

Check Output

[1] 31

bash-5.2# Ncat: bind to /var/run/docker.sock: Address already in use. QUITTING.

[1]+ Exit 2 nc -lU /var/run/docker.sock

Now, run the following command inside the Pod to trigger the Finding.

1

echo SocketAccessd | nc -w5 -U /var/run/docker.sock

Check Output

1

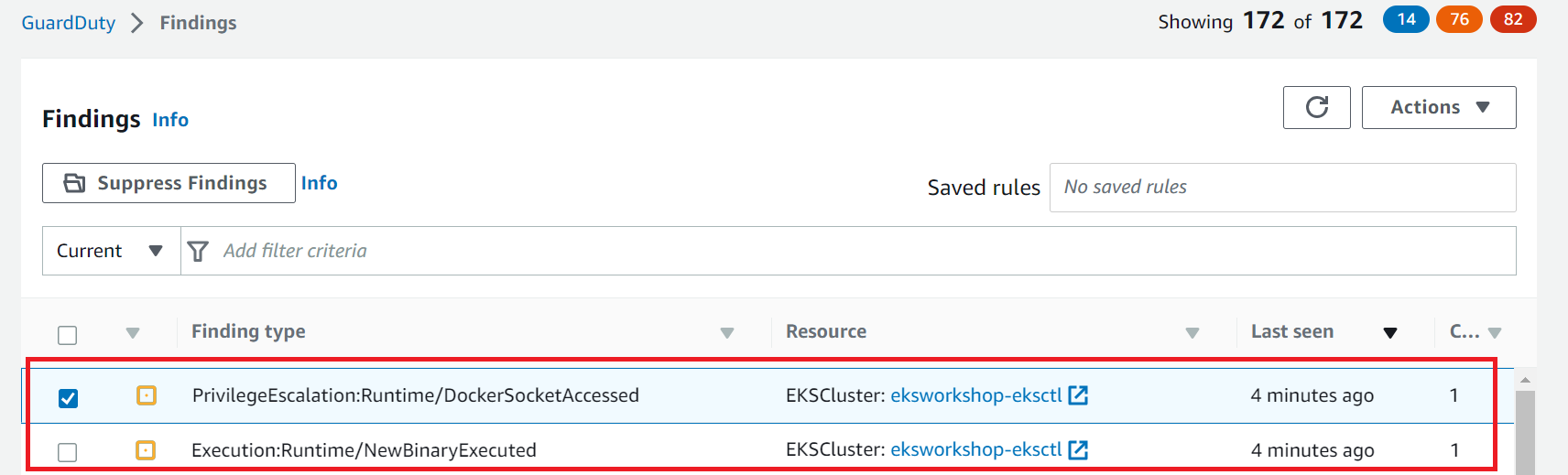
Ncat: Connection refused.

Go back [AWS GuardDuty console](javascript:void(0)) and check that a finding is generated for this.

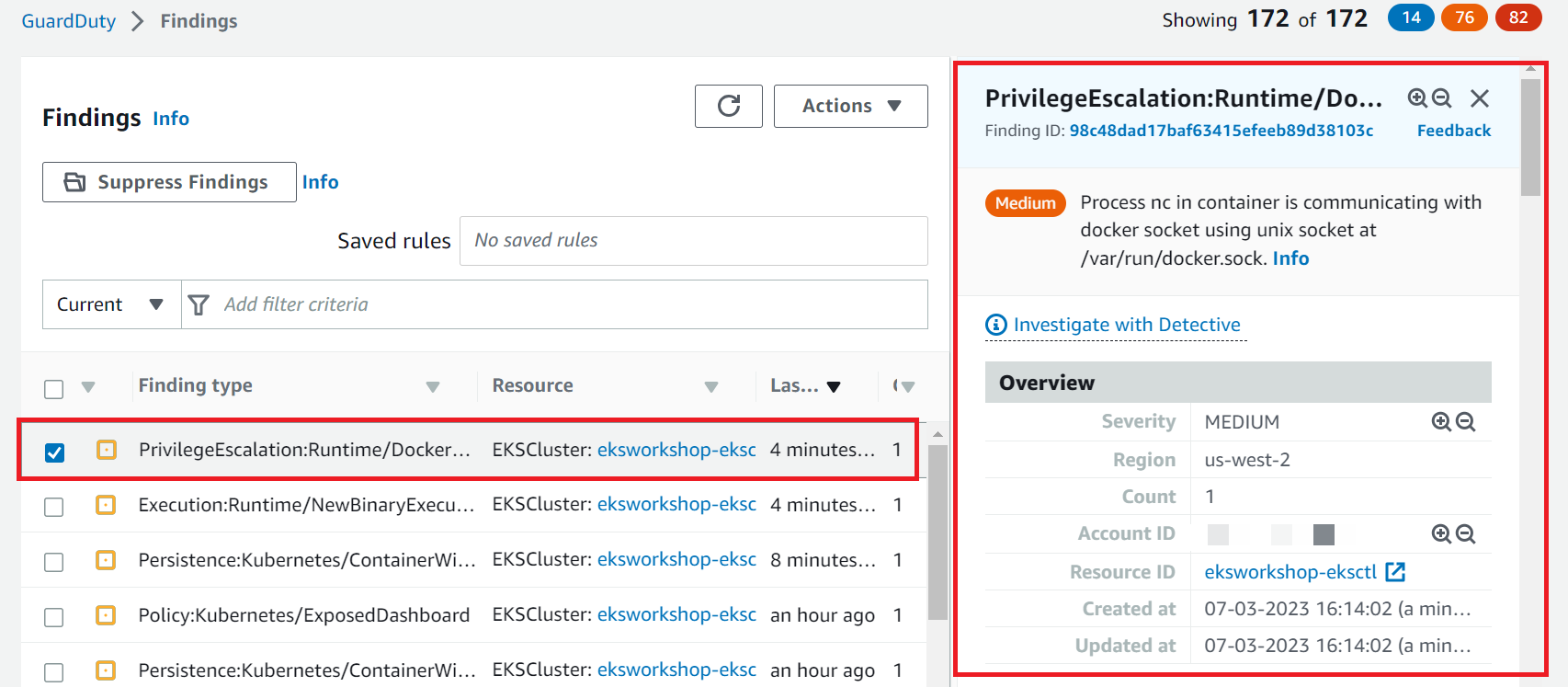
**Note**

If the finding doesn’t appear immediateley in the GuardDuty Console, keep refreshing the page since it make take few minutes to to generate the Kubernetes Findings

You can find the following findings in the Console.



Click on any of the Findings to the details.



Cleanup

Once you have completed this chapter, you can cleanup the files and resources you created by issuing the following commands:

eksctl delete addon --cluster $EKS\_CLUSTER\_NAME --name $GD\_EKS\_ADDON\_NAME

aws ec2 delete-vpc-endpoints --vpc-endpoint-ids $VPC\_ENDPOINT\_ID

aws ec2 delete-security-group --group-id $SECURITY\_GROUP\_ID