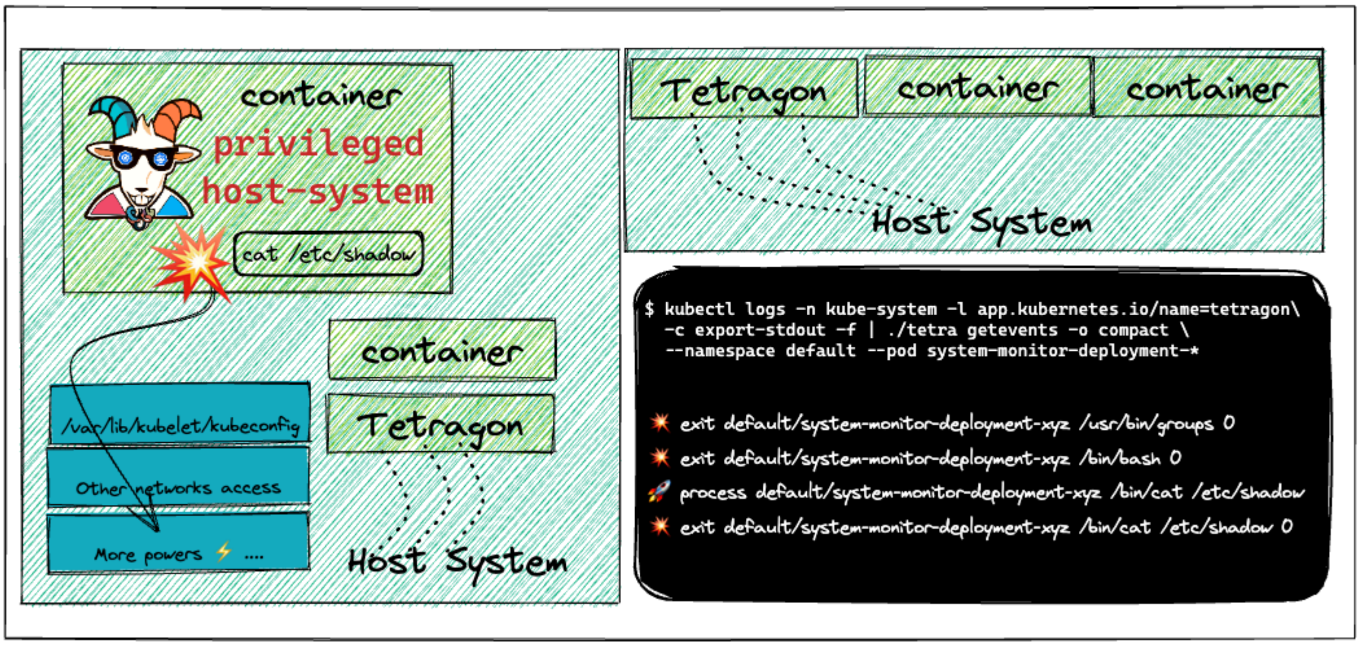
⎈ Cilium Tetragon - eBPF-based Security Observability and Runtime Enforcement

🙌 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 Cilium Tetragon to detect and perform runtime security monitoring using the `tracingpolicy` in action.



```
$ kubectl logs -n kube-system -l app.kubernetes.io/name=tetragon\
  -c export-stdout -f | ./tetra getevents -o compact \
  --namespace default --pod system-monitor-deployment-*


💥 exit default/system-monitor-deployment-xyz /usr/bin/groups 0
💥 exit default/system-monitor-deployment-xyz /bin/bash 0
🚀 process default/system-monitor-deployment-xyz /bin/cat /etc/shadow
💥 exit default/system-monitor-deployment-xyz /bin/cat /etc/shadow 0
```

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

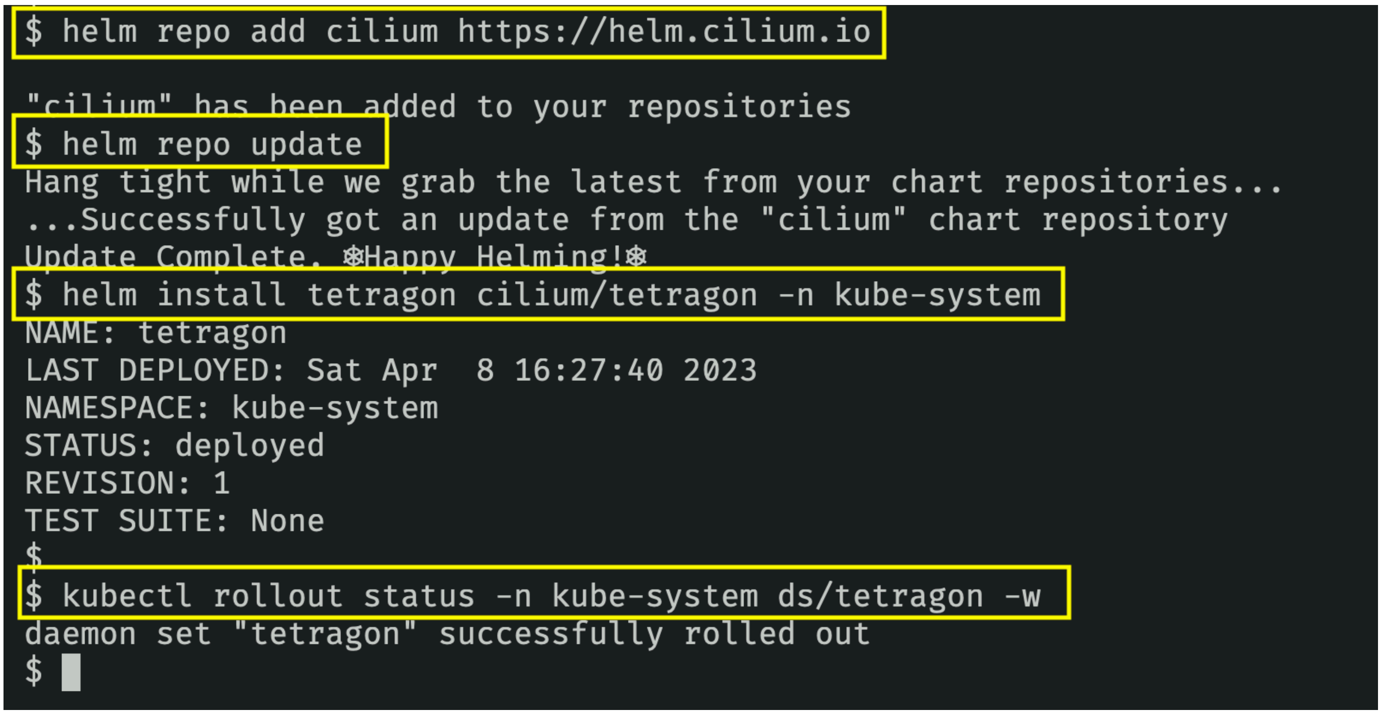
1. Deploying the helm chart into the Kubernetes cluster
2. Performing the log analysis and detection of security events in the Kubernetes Cluster
3. Use, analyze and detect security issues in near real-time using Cilium Tetragon

⚡ 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 detections using `tracingpolicy` based on attacker triggers.

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

> INFO
>
> - To deploy the Cilium Tetragon helm chart run the following commands
>
> ```
> helm repo add cilium https://helm.cilium.io
> ```
>
> ```
> helm repo update
> ```
>
> ```
> helm install tetragon cilium/tetragon -n kube-system
> ```
>
> - You can verify that the Tetragon pods are in running state by running the following command
>
> ```
> kubectl rollout status -n kube-system ds/tetragon -w
> ```



```
$ helm repo add cilium https://helm.cilium.io

"cilium" 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 "cilium" chart repository
Update Complete. ⎈Happy Helming!⎈
$ helm install tetragon cilium/tetragon -n kube-system
NAME: tetragon
LAST DEPLOYED: Sat Apr  8 16:27:40 2023
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
$
$ kubectl rollout status -n kube-system ds/tetragon -w
daemon set "tetragon" successfully rolled out
$ █
```

🎯 Goal

> TIP
>
> Deploy the Cilium Tetragon and detect security events in near real-time using the `tracingpolicy` and events.

☐ Hints & Spoilers

▸ ✨ Want to know more?

🎉 Solution & Walkthrough

🎲 Method 1

> INFO
>
> Cilium's new Tetragon component enables powerful realtime, eBPF-based Security Observability and Runtime Enforcement.

Tetragon detects and is able to react to security-significant events, such as

- Process execution events
- System call activity
- I/O activity including network & file access

When used in a Kubernetes environment, Tetragon is Kubernetes-aware - that is, it understands Kubernetes identities such as namespaces, pods and so-on - so that security event detection can be configured in relation to individual workloads.

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

```
kubectl get pods -n kube-system --selector app.kubernetes.io/instance=tetragon
```

- Manually obtaining the logs from the Tetragon using the following command

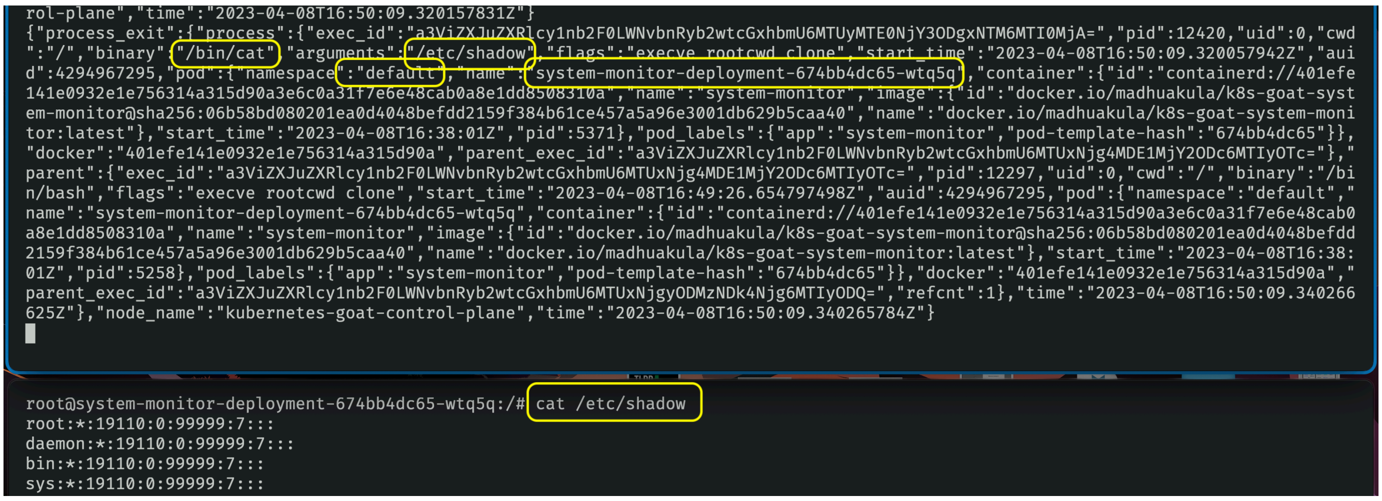
```
kubectl logs -n kube-system -l app.kubernetes.io/name=tetragon -c export-
stdout -f
```

- Now, let's use the system-monitor pod and try privileges escalation to the host system and see if that detects by Tetragon

```
export POD_NAME=$(kubectl get pods -l "app=system-monitor" -o jsonpath="
{.items[0].metadata.name}")
kubectl exec -it $POD_NAME bash
```

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

```
cat /etc/shadow
```



```
rol-plane","time":"2023-04-08T16:50:09.320157831Z"}
{"process_exit":{"process":{"exec_id":"a3ViZXJuZXRlcy1nb2F0LWNvbnRyb2wtcGxhbmU6MTUyMTE0NjY3ODgxNTM6MTI0MjA=","pid":12420,"uid":0,"cwd
":"/","binary":"/bin/cat","arguments":"/etc/shadow","flags":"execve rootcwd clone","start_time":"2023-04-08T16:50:09.320057942Z","aui
d":4294967295,"pod":{"namespace":"default","name":"system-monitor-deployment-674bb4dc65-wtq5q","container":{"id":"containerd://401efe
141e0932e1e756314a315d90a3e6c0a31f7e6e48cab0a8e1dd8508310a","name":"system-monitor","image":{"id":"docker.io/madhuakula/k8s-goat-syst
em-monitor@sha256:06b58bd080201ea0d4048befdd2159f384b61ce457a5a96e3001db629b5caa40","name":"docker.io/madhuakula/k8s-goat-system-moni
tor:latest"},"start_time":"2023-04-08T16:38:01Z","pid":5371},"pod_labels":{"app":"system-monitor","pod-template-hash":"674bb4dc65"}},
"docker":"401efe141e0932e1e756314a315d90a","parent_exec_id":"a3ViZXJuZXRlcy1nb2F0LWNvbnRyb2wtcGxhbmU6MTUxNjg4MDE1MjY2ODc6MTIyOTc="},"
parent":{"exec_id":"a3ViZXJuZXRlcy1nb2F0LWNvbnRyb2wtcGxhbmU6MTUxNjg4MDE1MjY2ODc6MTIyOTc=","pid":12297,"uid":0,"cwd":"/","binary":"/bi
n/bash","flags":"execve rootcwd clone","start_time":"2023-04-08T16:49:26.654797498Z","auid":4294967295,"pod":{"namespace":"default","
name":"system-monitor-deployment-674bb4dc65-wtq5q","container":{"id":"containerd://401efe141e0932e1e756314a315d90a3e6c0a31f7e6e48cab0
a8e1dd8508310a","name":"system-monitor","image":{"id":"docker.io/madhuakula/k8s-goat-system-monitor@sha256:06b58bd080201ea0d4048befdd
2159f384b61ce457a5a96e3001db629b5caa40","name":"docker.io/madhuakula/k8s-goat-system-monitor:latest"},"start_time":"2023-04-08T16:38:
01Z","pid":5258},"pod_labels":{"app":"system-monitor","pod-template-hash":"674bb4dc65"}},"docker":"401efe141e0932e1e756314a315d90a","
parent_exec_id":"a3ViZXJuZXRlcy1nb2F0LWNvbnRyb2wtcGxhbmU6MTUxNjgyODMzNDk4Njg6MTIyODQ=","refcnt":1},"time":"2023-04-08T16:50:09.340266
625Z"},"node_name":"kubernetes-goat-control-plane","time":"2023-04-08T16:50:09.340265784Z"}
```

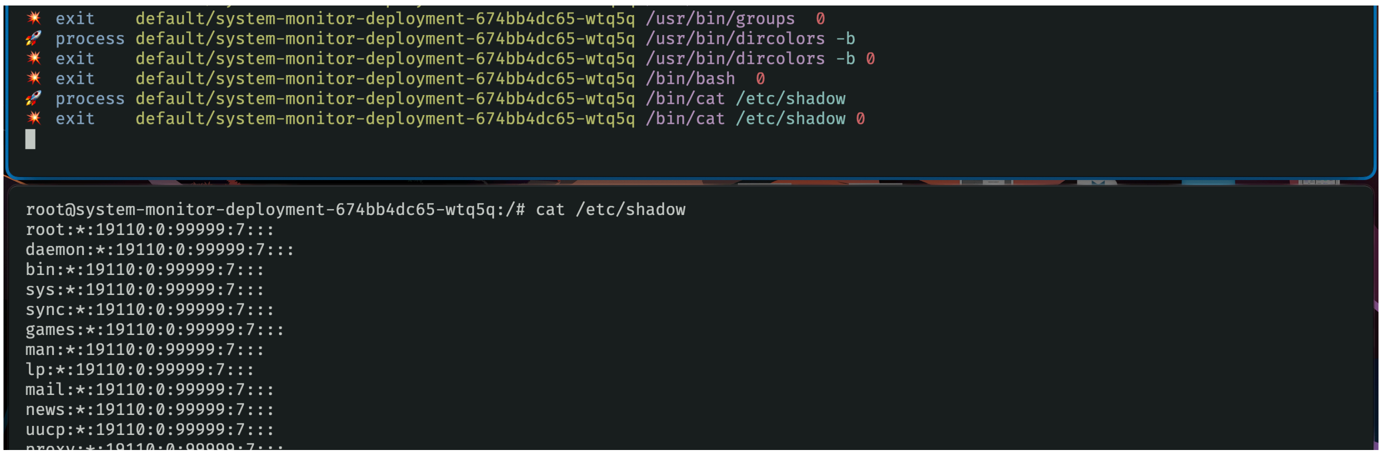
```
root@system-monitor-deployment-674bb4dc65-wtq5q:/# cat /etc/shadow
root:*:19110:0:99999:7:::
daemon:*:19110:0:99999:7:::
bin:*:19110:0:99999:7:::
sys:*:19110:0:99999:7:::
```

- You can also see these events in a nicer way using the official `tetra` cli client in your local system. Refer to the documentation for specific binary as per your **OS and Architecture**

```
wget https://github.com/cilium/tetragon/releases/download/v0.9.0/tetra-linux-amd64.tar.gz
tar -xvzf tetra-linux-amd64.tar.gz
```

- Now you can run the following command to pass the output of the Tetragon events to `tetra` cli locally to see in a nicer way

```
kubectl logs -n kube-system -l app.kubernetes.io/name=tetragon -c export-stdout -f | ./tetra getevents -o compact --namespace default --pod system-monitor-deployment-*
```



```
💥 exit    default/system-monitor-deployment-674bb4dc65-wtq5q /usr/bin/groups  0
🚀 process default/system-monitor-deployment-674bb4dc65-wtq5q /usr/bin/dircolors -b
💥 exit    default/system-monitor-deployment-674bb4dc65-wtq5q /usr/bin/dircolors -b 0
💥 exit    default/system-monitor-deployment-674bb4dc65-wtq5q /bin/bash  0
🚀 process default/system-monitor-deployment-674bb4dc65-wtq5q /bin/cat /etc/shadow
💥 exit    default/system-monitor-deployment-674bb4dc65-wtq5q /bin/cat /etc/shadow 0
```

```
root@system-monitor-deployment-674bb4dc65-wtq5q:/# cat /etc/shadow
root:*:19110:0:99999:7:::
daemon:*:19110:0:99999:7:::
bin:*:19110:0:99999:7:::
sys:*:19110:0:99999:7:::
sync:*:19110:0:99999:7:::
games:*:19110:0:99999:7:::
man:*:19110:0:99999:7:::
lp:*:19110:0:99999:7:::
mail:*:19110:0:99999:7:::
news:*:19110:0:99999:7:::
uucp:*:19110:0:99999:7:::
```

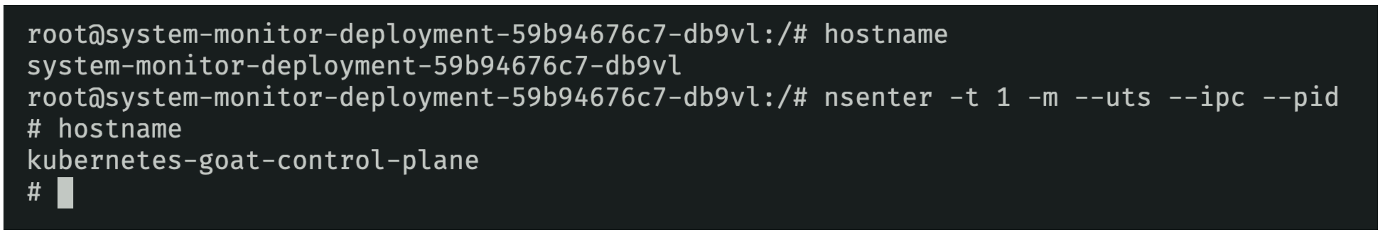
TIP

`tetra` cli provides the context awareness of the Kubernetes, namespaces and other details like processes, etc. When querying you can even limit them to processes, namespace, pod, even regex supported.

- Let's take a spin for detecting the privilege escalation attacks using the Tetragon. You can perform the container escape using the `system-monitor` pod to gain host system access by running the following commands

```
# Get into the system-monitor pod
export POD_NAME=$(kubectl get pods -l "app=system-monitor" -o jsonpath="{.items[0].metadata.name}")
kubectl exec -it $POD_NAME bash
```

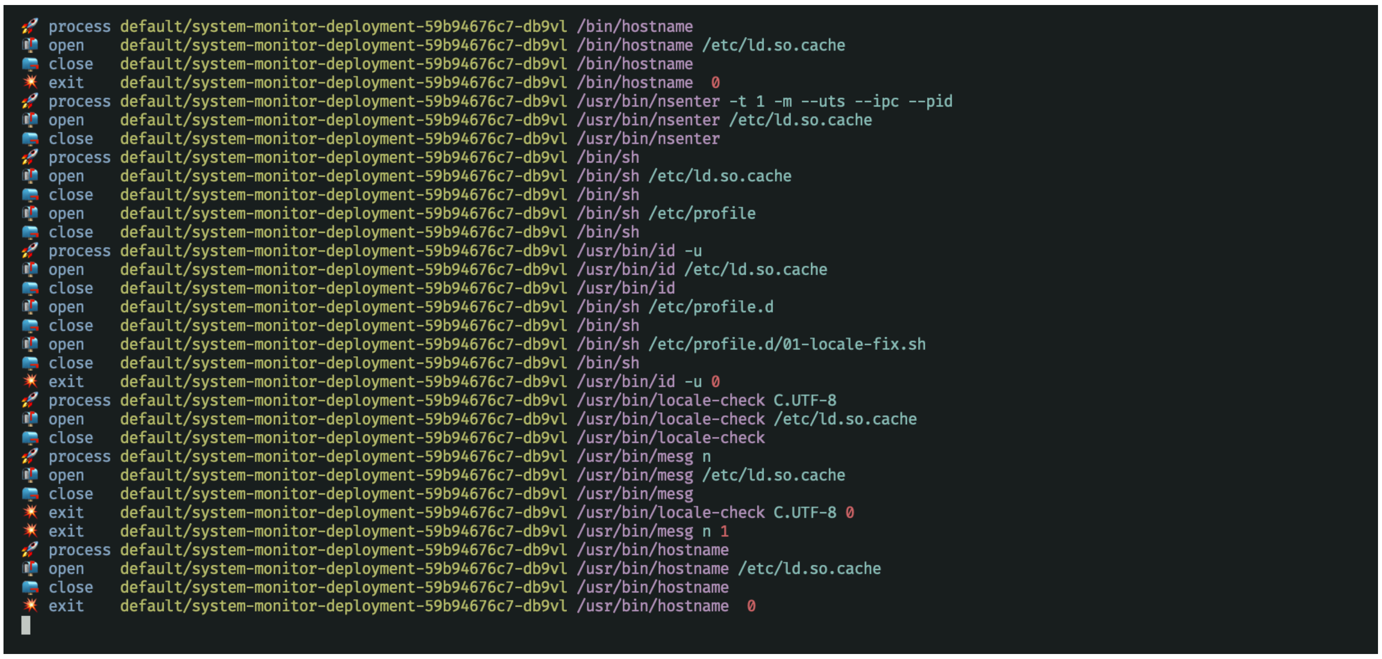
```
# Exploit to gain the host system access using nsenter
nsenter -t 1 -m --uts --ipc --pid
```



```
root@system-monitor-deployment-59b94676c7-db9vl:/# hostname
system-monitor-deployment-59b94676c7-db9vl
root@system-monitor-deployment-59b94676c7-db9vl:/# nsenter -t 1 -m --uts --ipc --pid
# hostname
kubernetes-goat-control-plane
# 
```

- As you can see following, Tetragon in action detecting these attacks in near real-time

```
kubectl logs -n kube-system -l app.kubernetes.io/name=tetragon -c export-stdout -f | ./tetra getevents -o compact --namespace default --pod system-monitor-deployment-*
```



```
🚀 process default/system-monitor-deployment-59b94676c7-db9vl /bin/hostname
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /bin/hostname /etc/ld.so.cache
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /bin/hostname
💥 exit    default/system-monitor-deployment-59b94676c7-db9vl /bin/hostname  0
🚀 process default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/nsenter -t 1 -m --uts --ipc --pid
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/nsenter /etc/ld.so.cache
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/nsenter
🚀 process default/system-monitor-deployment-59b94676c7-db9vl /bin/sh
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /bin/sh /etc/ld.so.cache
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /bin/sh
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /bin/sh /etc/profile
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /bin/sh
🚀 process default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/id -u
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/id /etc/ld.so.cache
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/id
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /bin/sh /etc/profile.d
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /bin/sh
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /bin/sh /etc/profile.d/01-locale-fix.sh
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /bin/sh
💥 exit    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/id -u 0
🚀 process default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/locale-check C.UTF-8
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/locale-check /etc/ld.so.cache
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/locale-check
🚀 process default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/mesg n
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/mesg /etc/ld.so.cache
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/mesg
💥 exit    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/locale-check C.UTF-8 0
💥 exit    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/mesg n 1
🚀 process default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/hostname
📬 open    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/hostname /etc/ld.so.cache
📪 close   default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/hostname
💥 exit    default/system-monitor-deployment-59b94676c7-db9vl /usr/bin/hostname  0
```

- Hooray 🥳 , now we can see that Tetragon detected the attack and notified using the events

🔖 References

- https://github.com/cilium/tetragon
- https://tetragon.cilium.io

✏️ Edit this page