CLOUDNATIVE SECURITYCON

NORTH AMERICA 2023

CLOUDNATIVE SECURITYCON NORTH AMERICA 2023

CSI Container: Can You DFIR It?

Stefano Chierici
Threat Research Lead Manager

Alberto Pellitteri
Security Researcher

#WhoAreWe?

- Stefano Chierici
- Threat Research Lead Manager @Sysdig
- @darryk10
- https://github.com/darryk10
- Falco Rule Reviewer and Contributor

- Alberto Pellitteri
- Security Researcher @Sysdig
- @pellibert1
- https://github.com/AlbertoPellitteri
- Falco Rule Contributor

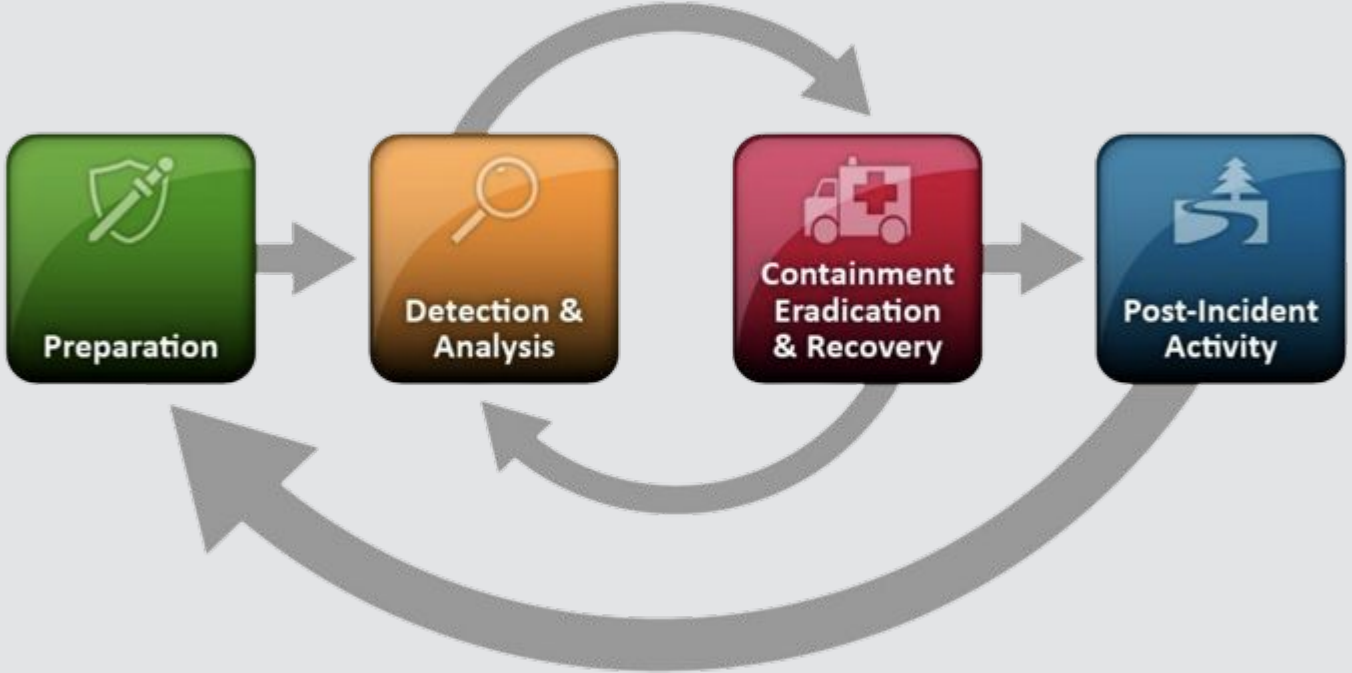
- DFIR = DF + IR
- NIST IR life-cycle
- DFIR in containers
  - Best practices
  - Steps
  - Tools
- Demo
- Main Takeaways

DFIR = DF + IR

Digital Forensics (DF)

Incident Response (IR)

DFIR - NIST IR life cycle



https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf

DFIR - Prepare your plan!

Traditional DFIR - it’s all about tools

CLOUDNATIVE SECURITYCON NORTH AMERICA 2023

Pretty well known!

Traditional DFIR Tools

- Tools
  - EDR/XDR
  - Volatility
  - RedLine
- Websites
  - Virustotal
- Cheatsheets
- Books

DFIR in containers

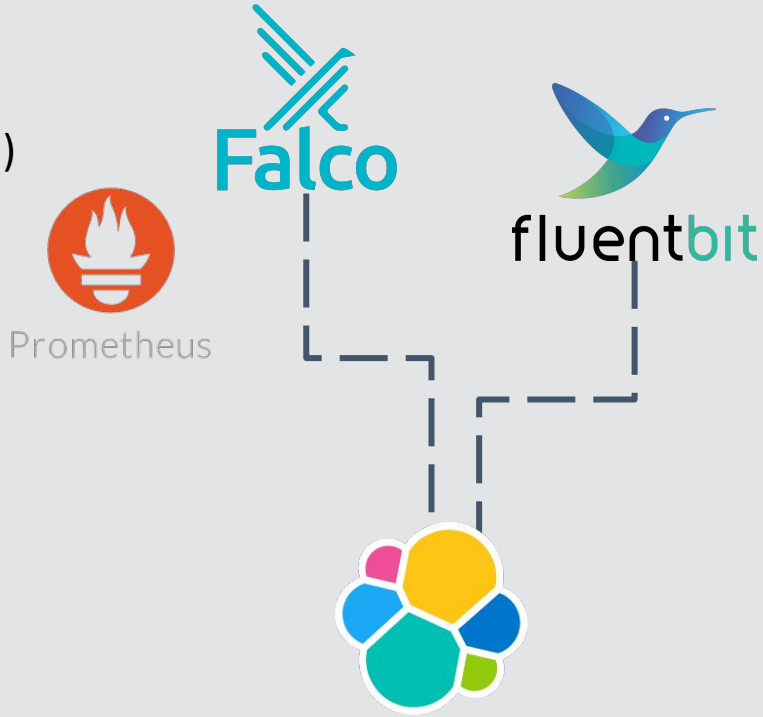
- What about DFIR in containers?
- Steps, approaches, guidelines
- Tools

https://sysdig.com/blog/guide-kubernetes-forensics-dfir/

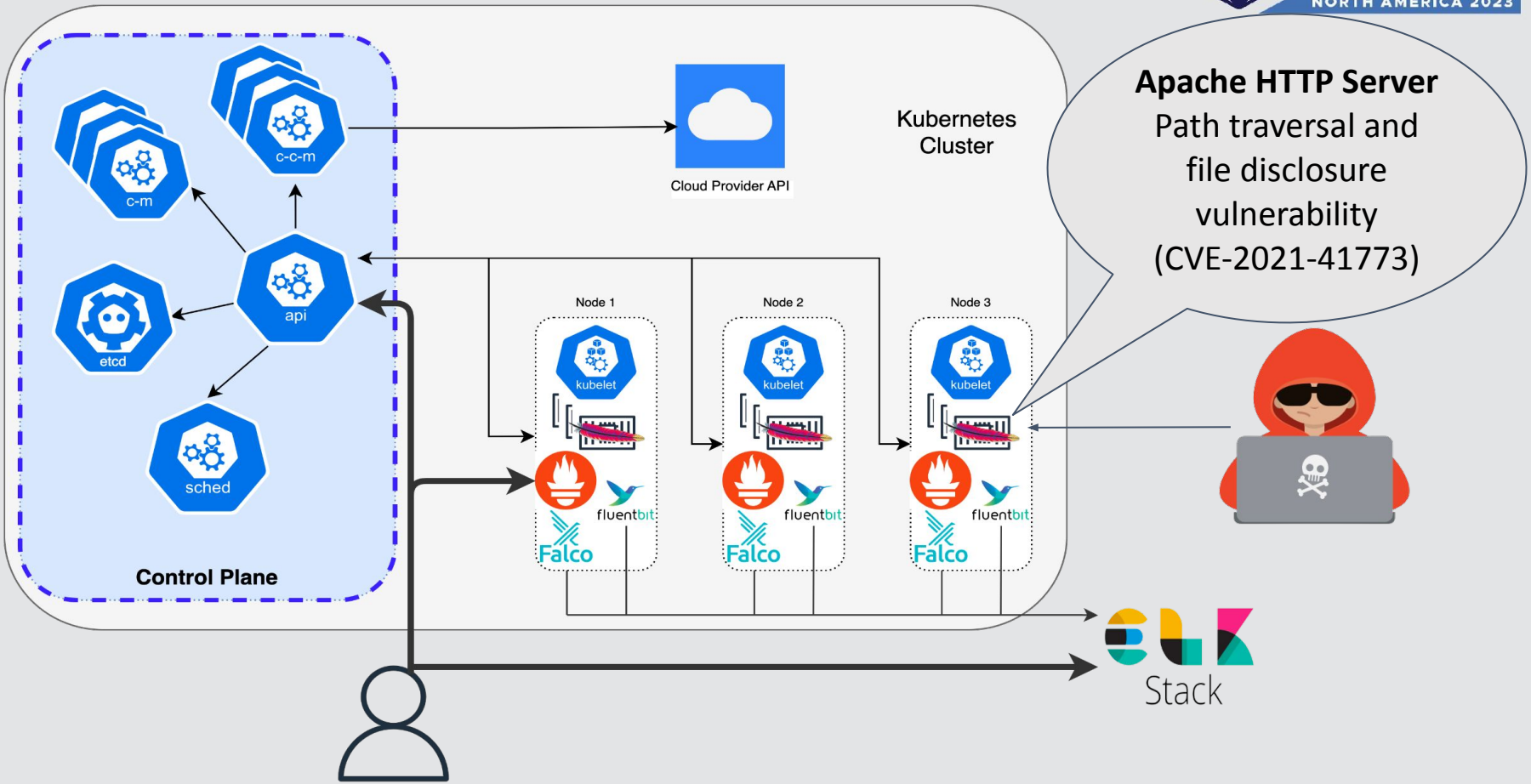
Preparation

Tools:

- Container runtime security:
  - Falco + Falcosidekick (CNCF incubated)
- Monitoring system:
  - Prometheus (CNCF graduated)
- Logging solution:
  - Fluent-bit/Fluentd (CNCF graduated)
- Log management platform:
  - ELK Stack, Opensearch, etc…

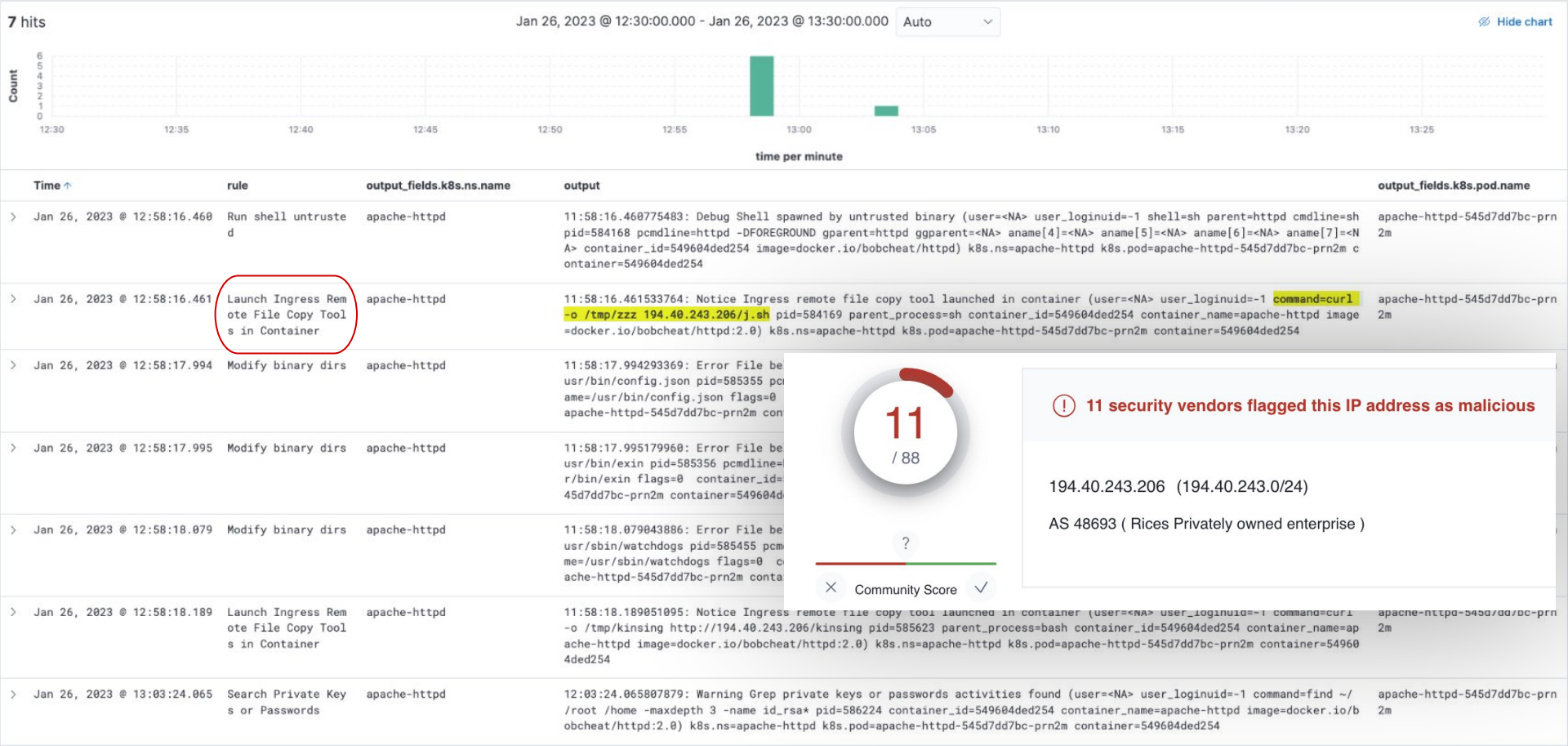


Demo


c-c-m
c-m
api
etcd
sched
Control Plane
Cloud Provider API
Kubernetes Cluster
Node 1
Node 2
Node 3
kubelet
fluentbit
Falco
Apache HTTP Server
Path traversal and file disclosure vulnerability (CVE-2021-41773)
Stack

Detection and Analysis

CLOUDNATIVE SECURITYCON NORTH AMERICA 2023



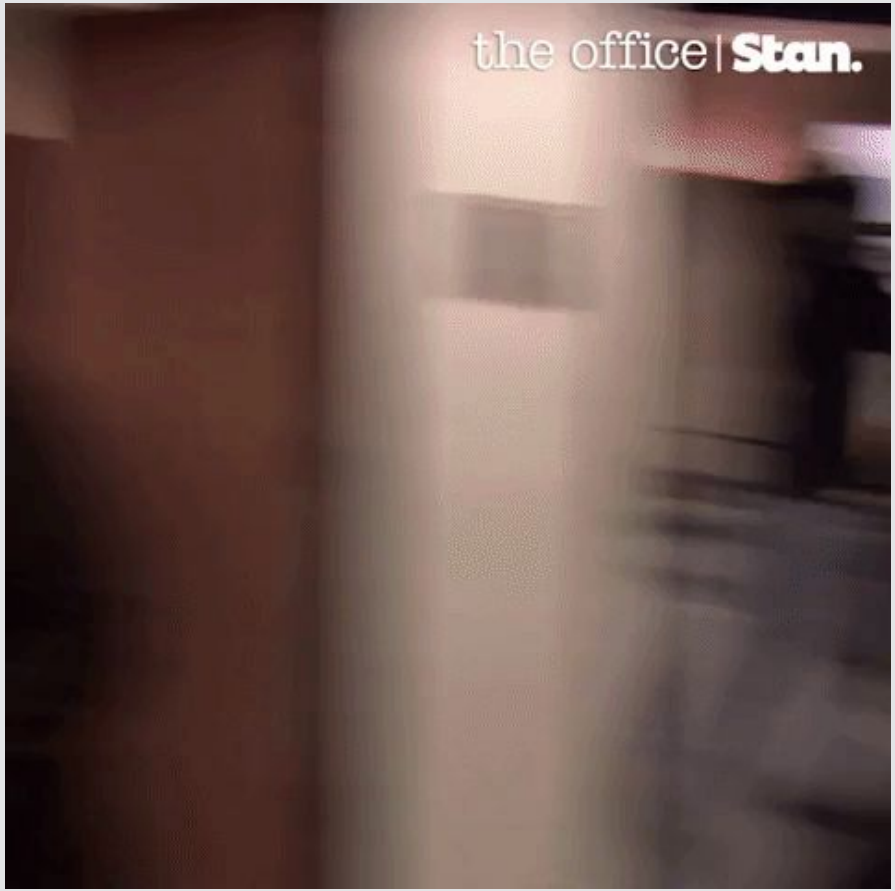
1,488 hits

Jan 26, 2023 @ 12:55:15.000 - Jan 26, 2023 @ 14:00:00.000 Auto

@timestamp per minute

Jan 26, 2023 @ 12:55:35.130	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:55:35.129993328Z stdout F 100.96.3.123 - - [26/Jan/2023:11:55:35 +0000] "GET / HTTP/1.1" 200 45
Jan 26, 2023 @ 12:55:35.170	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:55:35.17032277Z stdout F 100.96.3.123 - - [26/Jan/2023:11:55:35 +0000] "GET / HTTP/1.1" 200 45
Jan 26, 2023 @ 12:55:35.258	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:55:35.258478785Z stdout F 100.96.3.123 - - [26/Jan/2023:11:55:35 +0000] "GET / HTTP/1.1" 200 45
Jan 26, 2023 @ 12:58:16.470	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:16.469707336Z stderr F % Total % Received % Xferd Average Speed Time Time Time Current
Jan 26, 2023 @ 12:58:16.470	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:16.469740562Z stderr F Dload Upload Total Spent Left Speed
Jan 26, 2023 @ 12:58:16.998	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:16.998595139Z stderr F 0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0 16 33483 16 5536 0 0 15638 0 0:00:02 --:--:-- 0:00:02 15594 100 33483 100 33483 0 0 63294 0 --:--:-- --:--:-- --:--:-- 63175
Jan 26, 2023 @ 12:58:17.003	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:17.003517376Z stderr F chattr: Operation not permitted while setting flags on /tmp/
Jan 26, 2023 @ 12:58:17.006	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:17.004852653Z stderr F chattr: Operation not permitted while setting flags on /var/tmp/
Jan 26, 2023 @ 12:58:17.006	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:17.006083782Z stderr F chattr: No such file or directory while trying to stat /var/spool/cron
Jan 26, 2023 @ 12:58:17.009	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:17.007336547Z stderr F chattr: No such file or directory while trying to stat /etc/crontab
Jan 26, 2023 @ 12:58:17.009	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:17.007835859Z stderr F /tmp/zzz: line 8: ufw: command not found
Jan 26, 2023 @ 12:58:17.009	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:17.008699049Z stderr F /tmp/zzz: line 9: iptables: command not found
Jan 26, 2023 @ 12:58:17.009	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:17.009033924Z stderr F /tmp/zzz: line 11: sudo: command not found
Jan 26, 2023 @ 12:58:20.803	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:20.80247946Z stderr F /tmp/zzz: line 775: crontab: command not found
Jan 26, 2023 @ 12:58:20.807	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:20.803517206Z stderr F /tmp/zzz: line 776: crontab: command not found
Jan 26, 2023 @ 12:58:20.807	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:20.803959886Z stderr F /tmp/zzz: line 776: crontab: command not found
Jan 26, 2023 @ 12:58:20.807	apache-httpd-545d7dd7bc-prn2m	2023-01-26T11:58:20.805608415Z stdout F 172.20.50.212 - - [26/Jan/2023:11:58:16 +0000] "POST /cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/bin/sh HTTP/1.1" 200 115

Detection and Analysis



Containment, Eradication, and Recovery

- **Isolate** the attack:
  - Quarantine the impacted pod/container
  - Remove its privileges, revoke credentials, etc..
- Ensure **reliability** and **availability** to your services.
- Assess which **resources** have been **impacted**.
- **Store** the attack's evidences:
  - **Snapshot the worker node volume** where the impacted pod/container was scheduled (manually from you cloud provider console or with cloud-forensics-utils)
  - **Commit** and **push** the infected container
  - (best option) **If possible**, checkpoint the container
- **Fix** the misconfiguration/vulnerability, if possible. Otherwise **mitigate**.

What will we use?

- **docker/ctr/crictl/nerdctl/podman**: to interact with the involved container engine
- **kubectl**: to communicate with the kube-apiserver
- **docker-explorer/container-explorer** (by Google): open source projects that can do offline forensic analysis on a snapshotted volume.
- **container-diff** (by Google): is a tool for analyzing and comparing container images. It allows you to detect any changes within an image.
- **cloud-forensics-utils**: an open source project that provides tools to be used by forensics teams to collect evidence from cloud platforms. Currently, Google Cloud Platform, Microsoft Azure, and Amazon Web Services are supported.

Demo

Let’s begin isolating the impacted pod...

```
~ kubectl get po -n apache-httpd
NAME                            READY   STATUS    RESTARTS   AGE
apache-httpd-545d7dd7bc-cl4sn   1/1     Running   0          169m
apache-httpd-545d7dd7bc-cxdmb   1/1     Running   0          15h
apache-httpd-545d7dd7bc-dxq9q   1/1     Running   0          15h
apache-httpd-545d7dd7bc-hpjwz   1/1     Running   0          15h
apache-httpd-545d7dd7bc-j6kts   1/1     Running   0          15h
apache-httpd-545d7dd7bc-mq2gp   1/1     Running   0          15h
apache-httpd-545d7dd7bc-prn2m   1/1     Running   0          15h
apache-httpd-545d7dd7bc-qm98g   1/1     Running   0          38h
apache-httpd-545d7dd7bc-qz6kd   1/1     Running   0          15h
apache-httpd-545d7dd7bc-sbgqv   1/1     Running   0          15h

~ kubectl label po -n apache-httpd apache-httpd-545d7dd7bc-prn2m status=affected
pod/apache-httpd-545d7dd7bc-prn2m labeled

~ cat deny-all-affected.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-affected
spec:
  podSelector:
    matchLabels:
      status: affected
  policyTypes:
  - Ingress
  - Egress

~ kubectl create -f deny-all-affected.yaml -n apache-httpd
networkpolicy.networking.k8s.io/deny-all-affected created
```

Demo

…and cordoning the node.

```
~ kubectl get po -n apache-httpd -o wide | grep apache-httpd-545d7dd7bc-prn2m
apache-httpd-545d7dd7bc-prn2m   1/1     Running   0          15h    100.96.1.143   i-0b2aa6818f893983b

~ kubectl cordon i-0b2aa6818f893983b
node/i-0b2aa6818f893983b cordoned

~ kubectl get nodes
NAME                  STATUS                     ROLES           AGE     VERSION
i-00f4664cb11fb1097   Ready                      node            14h     v1.25.0
i-09fb6a9e4014386d8   Ready                      node            14h     v1.25.0
i-0b27eb437eede62c4   Ready                      control-plane   3d20h   v1.25.0
i-0b2aa6818f893983b   Ready,SchedulingDisabled   node            3d20h   v1.25.0

~
```

Demo - Live approach

Directly interact with the container engine…

```
ubuntu@i-0b2aa6818f893983b:~$
ubuntu@i-0b2aa6818f893983b:~$ sudo crictl ps | grep apache-httpd-545d7dd7bc-prn2m
549604ded2540      ec8402e48fcbf       15 hours ago        Running             apache-httpd                      0                   c5464b47cf655       apache-http
d-545d7dd7bc-prn2m
ubuntu@i-0b2aa6818f893983b:~$ sudo crictl logs 549604ded2540
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 100.96.1.143. Set the 'ServerName' directive globally to suppress this mess
age
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 100.96.1.143. Set the 'ServerName' directive globally to suppress this mess
age
[Wed Jan 25 21:39:37.338520 2023] [mpm_event:notice] [pid 1:tid 140207943607424] AH00489: Apache/2.4.49 (Unix) configured -- resuming normal operations
[Wed Jan 25 21:39:37.338684 2023] [core:notice] [pid 1:tid 140207943607424] AH00094: Command line: 'httpd -D FOREGROUND'
172.20.40.173 - - [26/Jan/2023:09:40:45 +0000] "POST / HTTP/1.1" 200 45
100.96.3.123 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
100.96.4.165 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
100.96.3.123 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
100.96.4.165 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
100.96.3.123 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
172.20.50.212 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
100.96.4.165 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
100.96.4.165 - - [26/Jan/2023:11:17:46 +0000] "GET / HTTP/1.1" 200 45
```

```
ubuntu@i-0b2aa6818f893983b:~$ sudo nerdctl -n k8s.io top 549604ded254
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                570998              570758              0                   Jan25               ?                   00:00:02            httpd -DFOREGROUND
daemon              571045              570998              0                   Jan25               ?                   00:00:00            httpd -DFOREGROUND
daemon              571046              570998              0                   Jan25               ?                   00:00:01            httpd -DFOREGROUND
daemon              571047              570998              0                   Jan25               ?                   00:00:02            httpd -DFOREGROUND
daemon              571048              570998              0                   Jan25               ?                   00:00:03            httpd -DFOREGROUND
daemon              583905              570998              0                   11:32               ?                   00:00:03            httpd -DFOREGROUND
daemon              585636              570998              0                   11:58               ?                   00:00:00            /tmp/kinsing
ubuntu@i-0b2aa6818f893983b:~$
```

Demo - Live approach


```
ubuntu@i-0b2aa6818f893983b:~$
ubuntu@i-0b2aa6818f893983b:~$ sudo  ctr -n k8s.io c ls | grep 549604ded2540
549604ded2540eed5837e646ede46b94d3c11ca40c2b74de2a9ba7c8d19fa319    docker.io/bobcheat/httpd:2.0
                            io.containerd.runc.v2
ubuntu@i-0b2aa6818f893983b:~$ sudo ctr -n k8s.io snapshot diff 549604ded2540eed5837e646ede46b94d3c11ca40c2b74de2a9ba7c8d19fa319 | tar -tz
etc/
etc/hosts
run/
run/lock/
run/lock/linux.lock
run/secrets/
run/secrets/kubernetes.io/
run/secrets/kubernetes.io/serviceaccount/
tmp/
tmp/.ICEd-unix/
tmp/.ICEd-unix/uuid
tmp/kdevtmpfsi
tmp/kinsing
tmp/zzz
usr/
usr/local/
usr/local/apache2/
usr/local/apache2/logs/
usr/local/apache2/logs/httpd.pid
var/
var/tmp/
var/tmp/.ICEd-unix/
ubuntu@i-0b2aa6818f893983b:~$
```

Demo - Live approach

Commit and push/save the compromised container from the **impacted worker's node**.

```
ubuntu@i-0b2aa6818f893983b:~$
ubuntu@i-0b2aa6818f893983b:~$
ubuntu@i-0b2aa6818f893983b:~$ sudo nerdctl -n k8s.io commit 549604ded254 docker.io/bobcheat/httpd:compromised
WARN[0000] Image lacks label "nerdctl/platform", assuming the platform to be "linux/amd64"
sha256:77b5ce84cbaa6cf742f295814945d02d096b75dcddac813f424f19c94666ca27
ubuntu@i-0b2aa6818f893983b:~$ sudo nerdctl -n k8s.io images | grep compromised
bobcheat/httpd                                      compromised                     e64fc6f001fc    21 seconds ago    linux/amd64    180.2 MiB     72.3 MiB
ubuntu@i-0b2aa6818f893983b:~$ sudo nerdctl -n k8s.io push docker.io/bobcheat/httpd:compromised
INFO[0000] pushing as a reduced-platform image (application/vnd.docker.distribution.manifest.v2+json, sha256:e64fc6f001fcfd487740c4f822b7343e41185db6c3e4b179fbfe1d68cb5
c1e9e)
manifest-sha256:e64fc6f001fcfd487740c4f822b7343e41185db6c3e4b179fbfe1d68cb5c1e9e: done           |++++++++++++++++++++++++++++++++++++++++|
config-sha256:77b5ce84cbaa6cf742f295814945d02d096b75dcddac813f424f19c94666ca27:   done           |++++++++++++++++++++++++++++++++++++++++|
elapsed: 2.6 s                                                                    total:  11.6 K (4.5 KiB/s)
ubuntu@i-0b2aa6818f893983b:~$
ubuntu@i-0b2aa6818f893983b:~$ sudo nerdctl -n k8s.io save docker.io/bobcheat/httpd:compromised --output /home/ubuntu/compromised.tar.gz
ubuntu@i-0b2aa6818f893983b:~$
```

Live approach in K8s - Ephemeral containers

In Kubernetes (≥v1.25) you can also use **ephemeral containers**.

Ephemeral containers are special type of containers that you can launch into already running pods to troubleshoot/inspect them. Doing so, you can inspect the main container's state and run arbitrary commands.

These special containers allow you to view processes in other containers leveraging **process namespace sharing**.

Useful:

- If you want to inspect the application state or the **attack's evidences** within the impacted pod;
- If the container to be inspected was run from a **distroless image** (has no shell or debugging tools).

Demo - Offline approach

Pull/import the previously compromised image within the **analyzer machine**.

```
ubuntu@ip-172-31-8-157:~$
ubuntu@ip-172-31-8-157:~$ sudo ctr image pull docker.io/bobcheat/httpd:compromised
docker.io/bobcheat/httpd:compromised:                                     resolved       |++++++++++++++++++++++++++++++++++++++|
manifest-sha256:e64fc6f001fcfd487740c4f822b7343e41185db6c3e4b179fbfe1d68cb5c1e9e: done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:0827b74117da10c400a8a41c2e650dda3f544df711daf005dce881af60877955:    done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:3739a2cbcb779ebc7638d0b1e241caa33beecd6d4f26ead251a9ac90cd72bbdf:    done           |++++++++++++++++++++++++++++++++++++++|
config-sha256:77b5ce84cbaa6cf742f295814945d02d096b75dcddac813f424f19c94666ca27:   done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:35a526fdcc7dca451994edc22a1770f5a9b36f85433dd53e48a92c49e378d67c:    done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:07aded7c29c6011dfdf02fc98e087c941d3c2661c4e73d134c6491e25231d16c:    done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:59fed288cd328965c61a1e58fed96f7ac63508030afd5999170f9e79e257b91f:    done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:05bb40c8f14889e7e53f646b971630940b47458d620a2d23489728b9864c6cc9:    done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:93aa46a38550269925ad04c3b043e5881a13d8474ee29a6fb6c4e25af5a6f26c:    done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:38ec434c79e07fdc6692be9a59d83c13f2b8e5cc3a8e83a163c4f5add420e769:    done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:ec8e2fb5feb929035dd68e60185d861ea3b3185a2cf55d2a3306b1ff2f21e5d0:    done           |++++++++++++++++++++++++++++++++++++++|
elapsed: 2.6 s                                                            total:  72.3 M (27.8 MiB/s)
unpacking linux/amd64 sha256:e64fc6f001fcfd487740c4f822b7343e41185db6c3e4b179fbfe1d68cb5c1e9e...
done: 3.618321034s
ubuntu@ip-172-31-8-157:~$ sudo ctr run -t docker.io/bobcheat/httpd:compromised compromised /bin/bash
root@ip-172-31-8-157:/usr/local/apache2# ls
bin  build  cgi-bin  conf  error  htdocs  icons  include  logs  modules
root@ip-172-31-8-157:/usr/local/apache2# ls /tmp
kdevtmpfsi  kinsing  zzz
root@ip-172-31-8-157:/usr/local/apache2#
```

Demo - Offline approach

Mount the snapshotted volume to the **analyzer machine**.

```
ubuntu@ip-172-31-8-157:~$
ubuntu@ip-172-31-8-157:~$ lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0      7:0    0 24.4M  1 loop /snap/amazon-ssm-agent/6312
loop1      7:1    0 55.6M  1 loop /snap/core18/2667
loop2      7:2    0 63.3M  1 loop /snap/core20/1778
loop3      7:3    0 91.9M  1 loop /snap/lxd/24061
loop4      7:4    0 49.6M  1 loop /snap/snapd/17883
xvda     202:0    0   16G  0 disk
├─xvda1  202:1    0 15.9G  0 part /
├─xvda14 202:14   0    4M  0 part
└─xvda15 202:15   0  106M  0 part
xvdf     202:80   0  100G  0 disk
├─xvdf1  202:81   0 99.9G  0 part
├─xvdf14 202:94   0    4M  0 part
└─xvdf15 202:95   0  106M  0 part
ubuntu@ip-172-31-8-157:~$ sudo mkdir /data
ubuntu@ip-172-31-8-157:~$ sudo mount /dev/xvdf1 /data
sudoubuntu@ip-172-31-8-157:~$ sudo ls /data
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc  root  run  sbin  snap  srv  sys  tmp  usr  var
ubuntu@ip-172-31-8-157:~$
```

Now you can perform forensics on the worker node's volume, using also **container-explorer** to analyze the previously running containers.

ubuntu@ip-172-31-8-157:~$

Container Checkpoint - Podman

How it started:

```
ubuntu@ip-172-31-11-43:~$ sudo podman exec -it tomcat /bin/bash
root@05b43a30c650:/usr/local/tomcat# ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  1.3  2.0 3586324 84304 pts/0   Ssl+ 10:51   0:02 /opt/java/openjdk/bin/java -Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties -Djava.uti
root        1507  0.0  0.4 710096 17424 ?        Sl   10:52   0:00 /etc/kinsing
root        1670  0.0  0.0      0     0 ?        Z    10:53   0:00 [pkill] <defunct>
root        1672  0.0  0.0      0     0 ?        Z    10:54   0:00 [pkill] <defunct>
root        1678  0.0  0.0      0     0 ?        Z    10:54   0:00 [kdevtmpfsi] <defunct>
root        1679  186 30.7 2585864 1233308 ?     Ssl  10:54   0:22 /tmp/kdevtmpfsi
root        1690  0.0  0.0   7632  3976 pts/1    Ss   10:54   0:00 /bin/bash
root        1693  0.0  0.0  10068  1564 pts/1    R+   10:54   0:00 ps -aux
root@05b43a30c650:/usr/local/tomcat#
ubuntu@ip-172-31-11-43:~$ sudo podman container checkpoint --export tomcat-checkpoint.tar.gz --tcp-established tomcat
tomcat
```

How it’s going:

```
ubuntu@ip-172-31-11-43:~$ sudo podman container restore -i tomcat-checkpoint.tar.gz --tcp-established
05b43a30c650e3e22b99d63d328bb9f7b5cb2224cf54ce39aab20382ae95813e
ubuntu@ip-172-31-11-43:~$ sudo podman ps
CONTAINER ID  IMAGE                                COMMAND          CREATED             STATUS                 PORTS                   NAMES
05b43a30c650  docker.io/[redacted]/tomcat:latest  catalina.sh run  About a minute ago  Up About a minute ago  0.0.0.0:8080->8080/tcp  tomcat
ubuntu@ip-172-31-11-43:~$ sudo podman exec -it tomcat /bin/bash
root@05b43a30c650:/usr/local/tomcat# ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.3  1.5 3586324 61420 pts/0   Ssl+ 11:49   0:00 /opt/java/openjdk/bin/java -Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties -Djava.uti
root          42  0.0  0.1   7632  4032 pts/1    Ss   11:50   0:00 /bin/bash
root          46  0.0  0.0  10068  3324 pts/1    R+   11:50   0:00 ps -aux
root        1507  0.0  0.3 710352 14420 ?        Sl   11:49   0:00 /etc/kinsing
root        1670  0.0  0.0      0     0 ?        Z    11:49   0:00 [pkill] <defunct>
root        1672  0.0  0.0      0     0 ?        Z    11:49   0:00 [pkill] <defunct>
root        1678  0.0  0.0      0     0 ?        Z    11:49   0:00 [kdevtmpfsi] <defunct>
root        1679 71.8 59.7 2789048 2397280 ?     Ssl  11:49   0:40 /tmp/kdevtmpfsi
root@05b43a30c650:/usr/local/tomcat#
```

https://sysdig.com/blog/forensic-container-checkpointing-dfir-kubernetes/

Eradication - Has the attack spread elsewhere?

- Before moving on with the next step, make sure that the attack has not spread elsewhere:
  - Check whether the affected pod was designed and deployed with **sensitive mounts** or **excessive privileges or capabilities**. If so, there may have been pod escaping or access to host privileged information.

```
[CRITICAL] Container Security Context User Group ID
    · apache-httpd -> Container has no configured security context
        Set securityContext to run the container in a more secure context.
```

```
[CRITICAL] Container Security Context Privileged
    · apache-httpd -> The container is privileged
        Set securityContext.privileged to false. Privileged containers can access all devices on the
        host, and grants almost the same access as non-containerized processes on the host.
```

Eradication - Has the attack spread elsewhere?

- Verify the **Kubernetes service account** bound to the impacted pod, if any. It may have granted unintended access to other resources in the cluster.

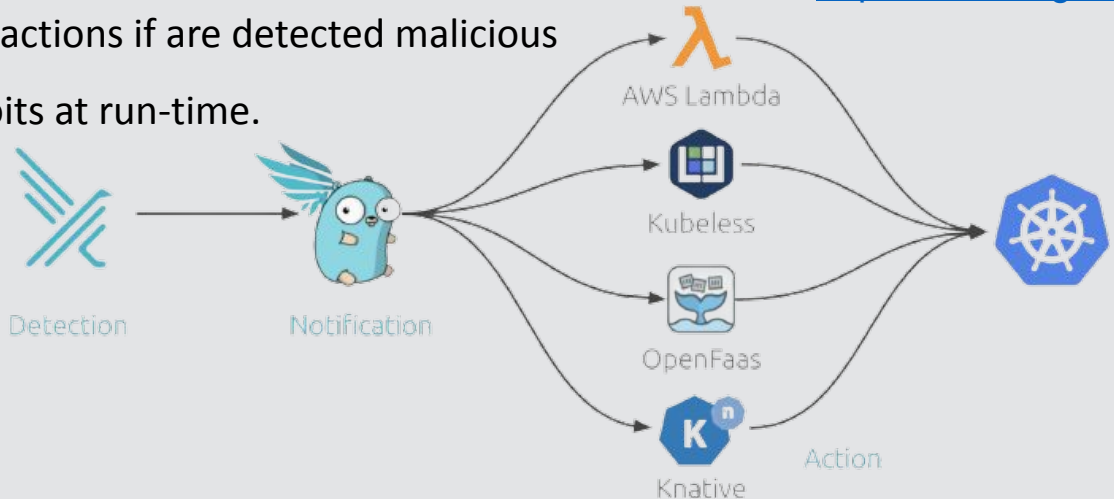
- Monitor the **Kubernetes audit logs** with runtime tools such as Falco, to detect any unwanted actions in the cluster:
  - Creation of new clusterroles/pods in the cluster
  - Stealing secrets…

- Inspect **Cloud logs** to monitor any lateral movement attempts. Sometimes impacted pods may leverage cloud metadata to create other cloud resources, access sensitive data, or cause damage to your cloud infrastructure. Also in this to detect alerts at runtime you can use Falco and its plugins.

Recovery

- Restore systems to normal and working operations:
  - Fix the vulnerability/misconfiguration, if possible.
  - Apply mitigations if there are no patches:
    - Delete the impacted workload
    - Restart the container
    - Run playbook of actions if are detected malicious executions/exploits at run-time.

https://falco.org/blog/



Post-Incident Activity - Continuous Improvement

When a security breach occurs, every company must embrace it as an **opportunity**.

It can represent a new way to protect the resources exposed, adopt new security approaches, and test the environments.
Giving it the proper attention, you can stop in time and prevent more disastrous cyber events in the future.

Main Takeaways

- **Do not be caught unprepared! Prepare your incident response plan!**
- Be prepared to respond to container breaches ASAP!
- Not all the container breaches are limited to the container itself. There might be more to investigate!
- Make sure to have set all the logs and to **fully monitor** your assets.
- Know your tools and be prepared to use them.
- Periodically **simulate** breaches and verify how your team respond to them.
- Hire/engage outside entities if needed.

CLOUDNATIVE
SECURITYCON
NORTH AMERICA 2023

Thank you!

Stefano Chierici
Threat Research Lead Manager
Twitter: @darryk
https://github.com/darryk10

Alberto Pellitteri
Security Researcher
Twitter: @pellibert1
https://github.com/AlbertoPellitteri