

# Docker Security Report

## **Index**

|                           |   |
|---------------------------|---|
| Introduction.....         | 3 |
| Objectives.....           | 3 |
| Challenges .....          | 3 |
| Objectives completed..... | 4 |
| Lessons Learnt.....       | 9 |
| Conclusion.....           | 9 |

## **Introduction**

Docker is an open-source project that automates the deployment of Linux applications inside software containers

Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Linux. Docker uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines.

To avoid any vulnerable system call, the objective of this project is to have a control over which system calls should be allowed and which should not be allowed. We came across user profiles that could be used to prevent vulnerable system calls. Using this information we came up with a design and implemented a security feature.

Github Project: <https://github.com/Changy-/Docker-Security>

## **Objectives**

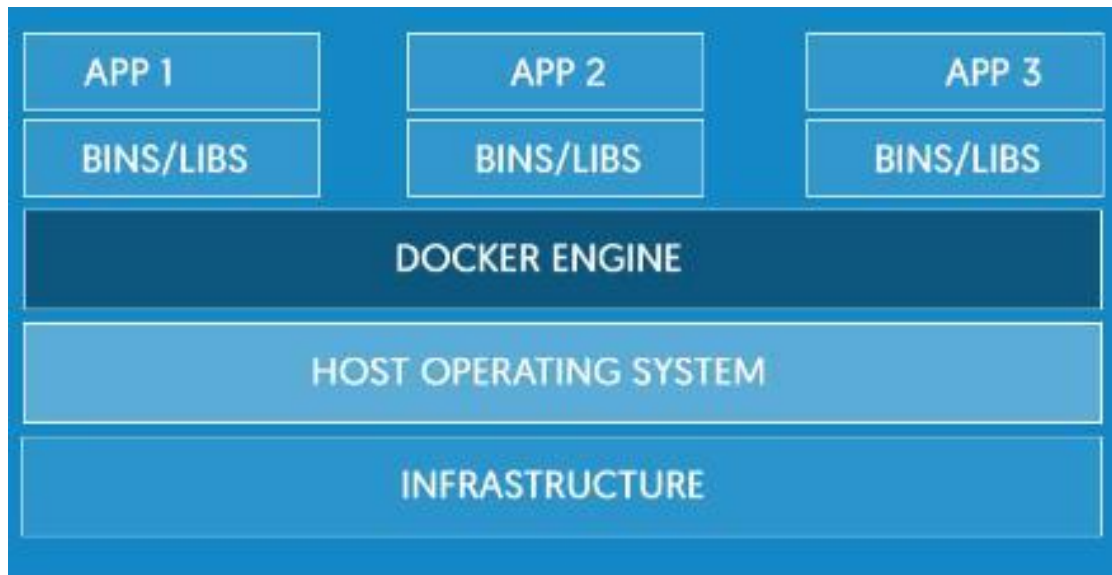
- Analyzing `runc` which is part of Docker Project.
- Perform a SECCOMP lab provided in the task list.
- Learning about SECCOMP profiles to filter system calls to prevent it to go through the OS.
- Limit the number of system calls to the underlying Operating System.
- Learning GO programming language.
- Blocking a list of 44 vulnerable system calls with parameters.
- Blocking CVE-2015-5706.
- Creating a C code to run a vulnerable system call(CVE-2015-5706) in Docker environment and then block it using the SECCOMP filter.

## **Challenges**

- Understand and learn how to block a system call using SECCOMP.
- Programming using GO.
- Writing C code at a very low level to make system calls and to test it.

## Objectives Completed

- Docker is like a middleware between Host Operating system and applications.



## SECCOMP Lab

- `sudo docker run --rm -it --cap-add ALL --security-opt apparmor=unconfined --security-opt seccomp=seccomp-profiles/deny.json alpine sh`

A screenshot of a gedit text editor window. The title bar reads 'deny.json (~/.labs/security/seccomp/seccomp-profiles) - gedit'. The editor contains a JSON configuration for a seccomp profile. The visible text is as follows:

```
[
  "defaultAction": "SCMP_ACT_ERRNO",
  "architectures": [
    "SCMP_ARCH_X86_64",
    "SCMP_ARCH_X86",
    "SCMP_ARCH_X32"
  ],
  "syscalls": [
  ]
}
```

```
Files
nyu_developer@nyu-developer-VirtualBox: ~/labs/security/seccomp
t seccomp=<profile>.json alpine sh ...
bash: profile: No such file or directory
nyu_developer@nyu-developer-VirtualBox:~$ git clone https://github.com/docker/l
bs
Cloning into 'labs'...
remote: Counting objects: 3316, done.
remote: Compressing objects: 100% (258/258), done.
remote: Total 3316 (delta 92), reused 0 (delta 0), pack-reused 3054
Receiving objects: 100% (3316/3316), 126.32 MiB | 2.76 MiB/s, done.
Resolving deltas: 100% (670/670), done.
Checking connectivity... done.
nyu_developer@nyu-developer-VirtualBox:~$ cd labs/security/seccomp/
nyu_developer@nyu-developer-VirtualBox:~/labs/security/seccomp$ sudo docker run
--rm -it --cap-add ALL --security-opt apparmor=unconfined --security-opt seccom
=seccomp-profiles/deny.json alpine sh
```

```
nyu_developer@nyu-developer-VirtualBox: ~/labs/security/seccomp
Command 'qgit' from package 'qgit' (universe)
Command 'luit' from package 'x11-utils' (main)
Command 'quizz' from package 'bsdgames' (universe)
quit: command not found
nyu_developer@nyu-developer-VirtualBox:~/labs/security/seccomp$ strace -c -f -S
name whoami 2>&1 1>/dev/null | tail -n +3 | head -n -2 | awk '{print $(NF)}'
access
arch_prctl
brk
close
connect
execve
fstat
geteuid
loctl
lseek
mmap
mprotect
munmap
open
read
socket
write
nyu_developer@nyu-developer-VirtualBox:~/labs/security/seccomp$
```

- We learnt how to configure White list or Black list of the system calls
- To block system calls with specific parameters the profile has Granular filters
- Blocking chmod system call.

```
nyu_developer@nyu-developer-VirtualBox: ~/labs/security/seccomp/seccomp-profiles
inspect    Display detailed information on one or more services
ps         List the tasks of a service
ls         List services
rm         Remove one or more services
scale     Scale one or multiple services
update     Update a service

Run 'docker service COMMAND --help' for more information on a command.
nyu_developer@nyu-developer-VirtualBox:~/labs/security/seccomp$ sudo docker run
--rm -it --security-opt seccomp=default-no-chmod.json alpine sh
docker: opening seccomp profile (default-no-chmod.json) failed: open default-no-
chmod.json: no such file or directory.
See 'docker run --help'.
nyu_developer@nyu-developer-VirtualBox:~/labs/security/seccomp$ cd seccomp-profi
les/
nyu_developer@nyu-developer-VirtualBox:~/labs/security/seccomp/seccomp-profiles$
sudo docker run --rm -it --security-opt seccomp=default-no-chmod.json alpine sh
/ #
/ # exit
nyu_developer@nyu-developer-VirtualBox:~/labs/security/seccomp/seccomp-profiles$
sudo docker run --rm -it --security-opt seccomp=default-no-chmod.json alpine sh
/ # chmod 777 / -v
chmod: /: Operation not permitted
/ #
```

## More Granular Filters

SCMP\_ACT\_KILL | SCMP\_ACT\_TRAP | SCMP\_ACT\_ERRNO  
| SCMP\_ACT\_TRACE | SCMP\_ACT\_ALLOW

syscall arg index

Logical operation to  
perform + parameters

```
...
"syscalls": [
  {
    "name": "accept",
    "action": "SCMP_ACT_ALLOW",
    "args": [
      {
        "index": 0,
        "op": "SCMP_CMP_MASKED_EQ",
        "value": 2000505056,
        "valueTwo": 0
      }
    ]
  }
]
```

Blocking of accept all with a specific parameter.

- Blocking of ls command for testing

```
j1n@J1n:~/changy/j1n/docker/runc/runc/mycontainer$ sudo runc run mycontainerid
/ # ls
bin  dev  etc  home  proc  root  sys  tmp  usr  var
/ #
/ #
```

- Adding security option to Docker options

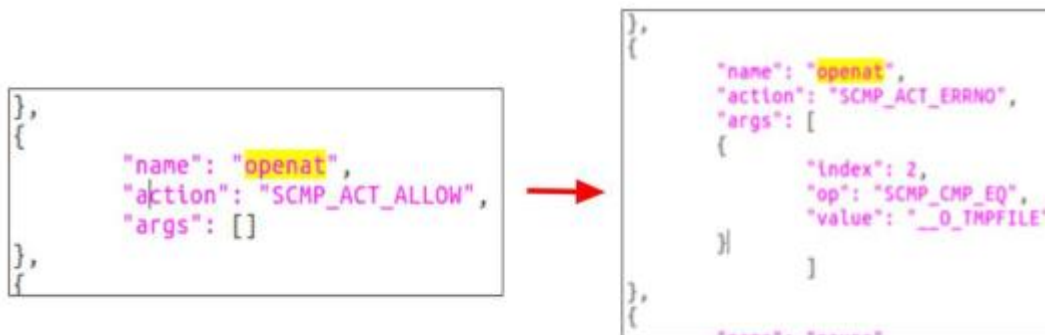
```
OPTIONS:
  --bundle value, -b value    path to the root of the bundle directory, defaults to the current directory
  --console value            specify the pty slave path for use with the container
  --detach, -d               detach from the container's process
  --pid-file value           specify the file to write the process id to
  --no-subreaper              disable the use of the subreaper used to reap reparented processes
  --no-pivot                  do not use pivot root to jail process inside rootfs. This should be used
  --no-new-keyring            do not create a new session keyring for the container. This will cause t
  --security                  provide security enhancement container(added by Team Chupacabra)

j1n@J1n:~/changy/j1n/docker/runc/runc$ runc run -security mycontainerid
```

```
j1n@J1n:~/changy/j1n/docker/runc/runc/mycontainer$ sudo runc run mycontainerid
/ # ls
ls: ./proc: Operation not permitted
ls: ./dev: Operation not permitted
ls: ./etc: Operation not permitted
ls: ./bin: Operation not permitted
ls: ./var: Operation not permitted
ls: ./sys: Operation not permitted
ls: ./usr: Operation not permitted
ls: ./home: Operation not permitted
ls: ./tmp: Operation not permitted
ls: ./root: Operation not permitted
/ #
/ #
/ #
```

- Operation not permitted due to security options configurations

- This was a successful attempt to block a system call
- Next is blocking of `openat` with specific argument `__O_TMPFILE`
- Following is the system call to be blocked.



- To trigger the vulnerability we wrote a code to run the vulnerability in the Docker environment.
- We used a C code which had a function to make the system call `openat`

```

nyu_developer@nyu-developer-VirtualBox: ~/Downloads/dockervul
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

void creat_at(char *dir_path, char *relative_path)
{
    int dir_fd;
    int fd;
    int flags;
    mode_t mode;

    dir_fd = open(dir_path, __O_TMPFILE | O_RDWR, 0640);
    if (dir_fd < 0) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    flags = __O_TMPFILE | O_RDWR | O_CREAT | O_TRUNC;
    mode = 0640;
    printf("1");
    fd = openat(dir_fd, relative_path, flags, mode);
    printf("2");
    if (fd < 0) {
        perror("openat");
        exit(EXIT_FAILURE);
    }

    write(fd, "HELLO", 5);

    close(fd);
    close(dir_fd);
}

int main()
{
    creat_at("/tmp", "log.txt");
    return 0;
}

```

- Then we executed the code twice first without blocking it and then with blocking it.
- First we observed that the code executed the function and later, we found that the code was executed.
- This was done using the SECCOMP profile.



```

nyu_developer@nyu-developer-VirtualBox: ~/Downloads/dockervul
1e3ee139a577: Pull complete
1a758e309ab3: Pull complete
031623f9a53b: Pull complete
f4cc5b8328ec: Pull complete
adae56261dc9: Pull complete
Digest: sha256:211f2a20dd8dac0f8c094514d9306bcb5b90fa422ae04830fa99a1c96eb5d03a
Status: Downloaded newer image for gcc:4.9
---> 6bc2c1b2249a
Step 2 : COPY . /usr/src/myapp
---> 00bdd40a1dcc
Removing intermediate container 52c7724533d5
Step 3 : WORKDIR /usr/src/myapp
---> Running in 963b26fbf4b1
---> ac8a1c33af4d
Removing intermediate container 963b26fbf4b1
Step 4 : RUN gcc -o myapp main.c
---> Running in 3bbc10f1c8b5
---> 60b780706948
Removing intermediate container 3bbc10f1c8b5
Step 5 : CMD ./myapp
---> Running in f3f121678d65
---> 47c8ae44600d
Removing intermediate container f3f121678d65
Successfully built 47c8ae44600d
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$ sudo docker run -it --rm --security-opt seccomp=d
efault.json --name my-running-app my-gcc-app
openat: Invalid argument
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$

```

```

nyu_developer@nyu-developer-VirtualBox: ~/Downloads/dockervul
Step 2 : COPY . /usr/src/myapp
---> 00bdd40a1dcc
Removing intermediate container 52c7724533d5
Step 3 : WORKDIR /usr/src/myapp
---> Running in 963b26fbf4b1
---> ac8a1c33af4d
Removing intermediate container 963b26fbf4b1
Step 4 : RUN gcc -o myapp main.c
---> Running in 3bbc10f1c8b5
---> 60b780706948
Removing intermediate container 3bbc10f1c8b5
Step 5 : CMD ./myapp
---> Running in f3f121678d65
---> 47c8ae44600d
Removing intermediate container f3f121678d65
Successfully built 47c8ae44600d
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$ sudo docker run -lt --rm --security-opt seccomp=d
efault.json --name my-running-app my-gcc-app
openat: Invalid argument
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$ vln default.json
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$ sudo docker run -lt --rm --security-opt seccomp=d
efault.json --name my-running-app my-gcc-app
docker: Error response from daemon: invalid header field value "oci runtime error: container_linux.go:247: star
ting container process caused \"open /proc/self/fd: operation not permitted\"\\n".
nyu_developer@nyu-developer-VirtualBox:~/Downloads/dockervul$

```

- Using the SECCOMP profile we added the list of 44 system calls to the black list that were vulnerable.



## Compile and run your own code inside of the container.

1. Run apt-get update and install gcc, java jdk etc.

```
apt-get update
```

```
apt-get install gcc
```

```
apt-get install default-jdk
```

2. Install any text editor, for example vim

```
apt-get install vim
```

- `sudo docker run -it --security-opt seccomp=team_chupacabra_security.json ubuntu sh`

### Lessons Learnt

- Understanding Linux level code and system calls and compiling the Linux kernel
- Learning of GO programming
- My individual contribution was developing the SECCOMP profile to block the openat system call with a O\_TMPFILE as a parameter and helping developing the C file to execute a system call with the specified parameter and executing it in the Docker environment.
- Learning CVE-2015-5706 vulnerability and 44 system calls which were vulnerable and why they are vulnerable

### Conclusion

Learnt Go programming language basics. The implementation of Docker and role of SECCOMP filters in blocking vulnerable system calls. Successfully blocked CVE-2015-5706 by designing a SECCOMP profile for the same. Wrote a C program to trigger vulnerabilities and test the SECCOMP filter. Learnt other system calls that are vulnerable.