Docker Security Report

Index

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

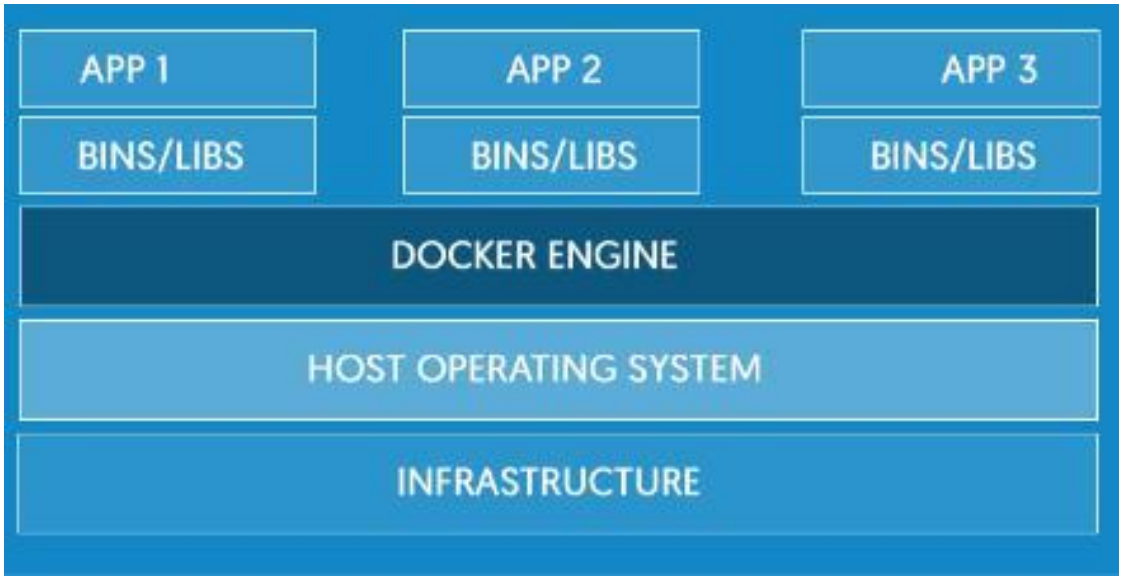
- Analyzing `runc` which is part of Docker Project.
- Perform a SECCOMP lab provided in the task list.
- Learning about SECCOMP profiles to filter system calls to prevent it to go through the OS.
- Limit the number of system calls to the underlying Operating System.
- Learning GO programming language.
- 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.

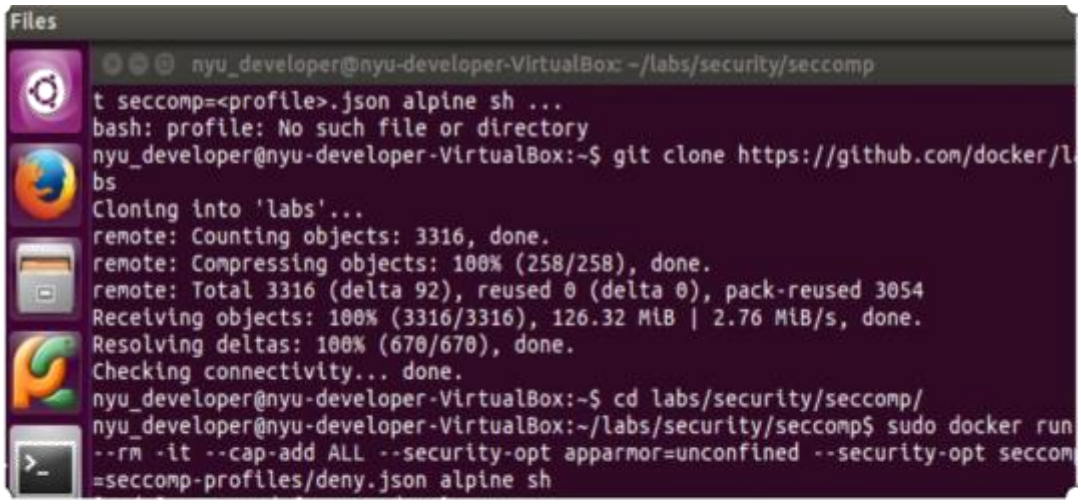
Objectives Completed

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

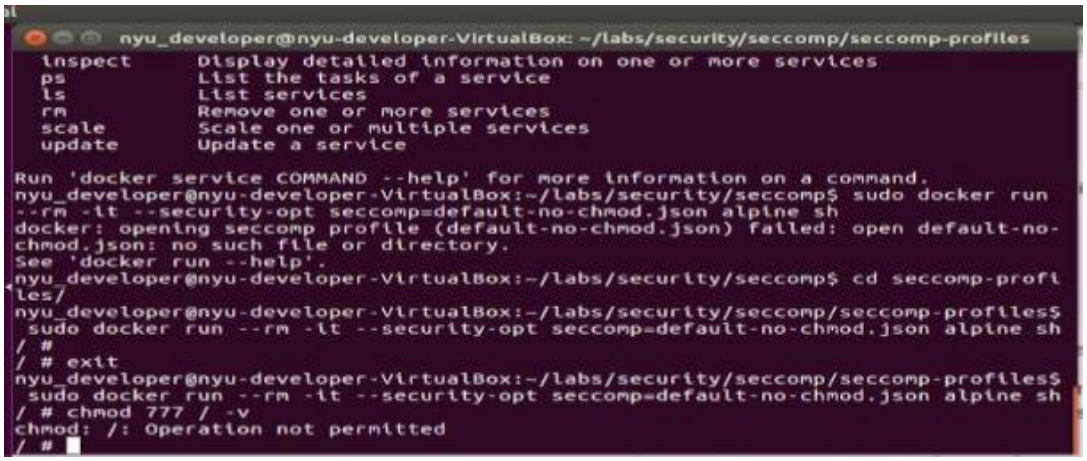


SECCOMP Lab

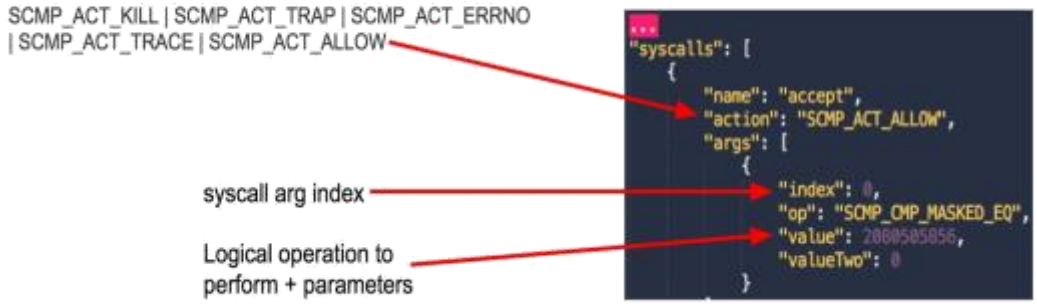
- sudo docker run --rm -it --cap-add ALL --security-opt apparmor=unconfined --security-opt seccomp=seccomp-profiles/deny.json alpine sh



- 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.



More Granular Filters



Blocking of accept all with a specific parameter.

- Blocking of ls command for testing

```
jin@Jin:~/changy/jin/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

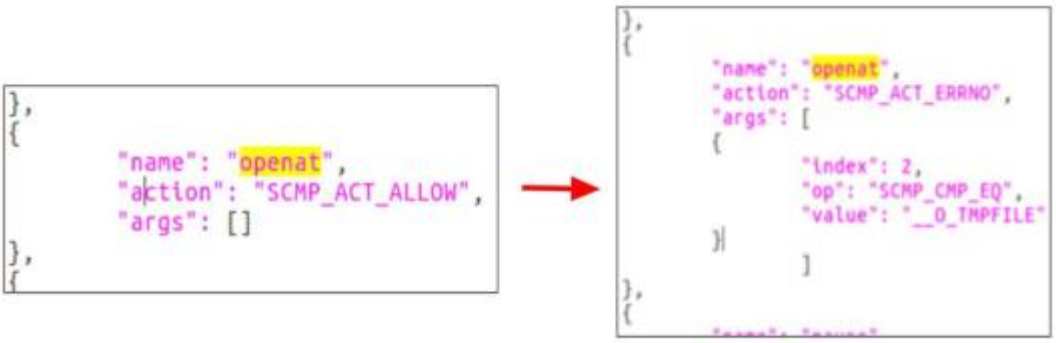
```
OPTIONS:
   --bundle value, -b value  path to the root of the bundle directory, defaults to the current directo
   --console value           specify the pty slave path for use with the container
   --detach, -d              detach from the container's process
   --pid-file value          specify 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)

jin@Jin:~/changy/jin/docker/runc/runc$ runc run -security mycontainerid
```

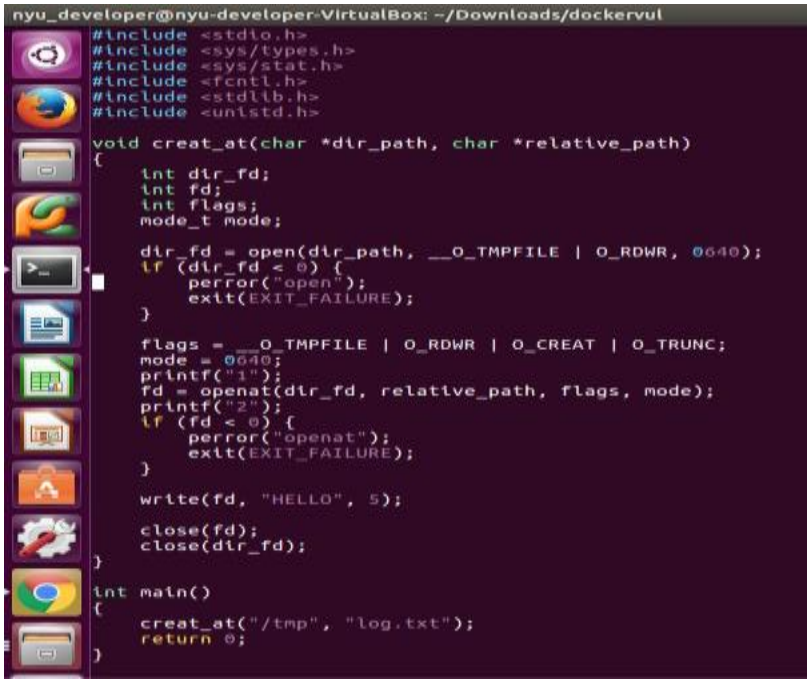
```
jin@Jin:~/changy/jin/docker/runc/runc/mycontainer$ sudo runc run mycontainerid
/ # ls
ls: ./proc: Operation not permitted
ls: ./dev: Operation not permitted
ls: ./etc: Operation not permitted
ls: ./bin: Operation not permitted
ls: ./var: Operation not permitted
ls: ./sys: Operation not permitted
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

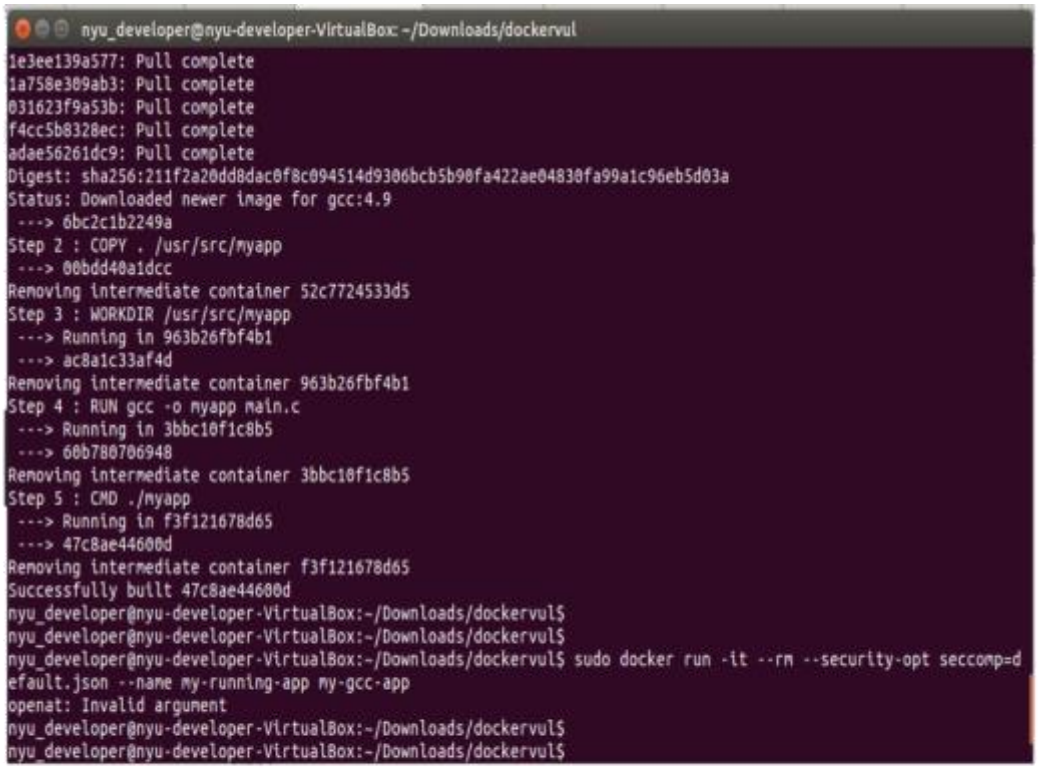
- This was a successful attempt to block a system call
- Next is blocking of openat with specific argument __O_TMPFILE
- Following is the system call to be blocked.

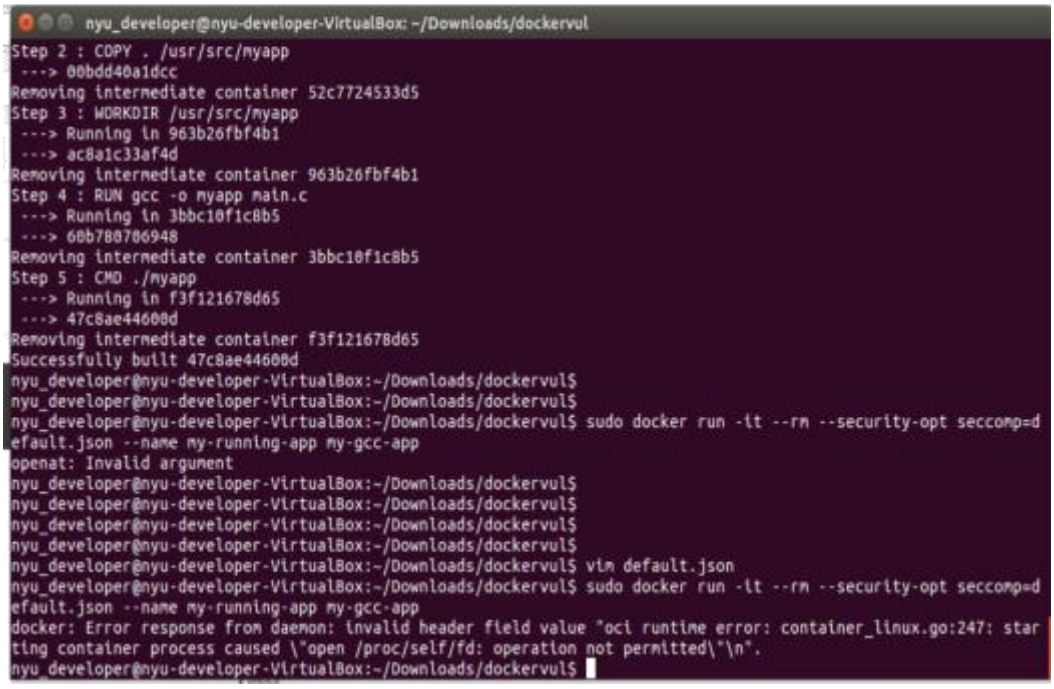


- To trigger the vulnerability we wrote a code to run the vulnerability in the Docker environment.
- We used a C code which had a function to make the system call openat

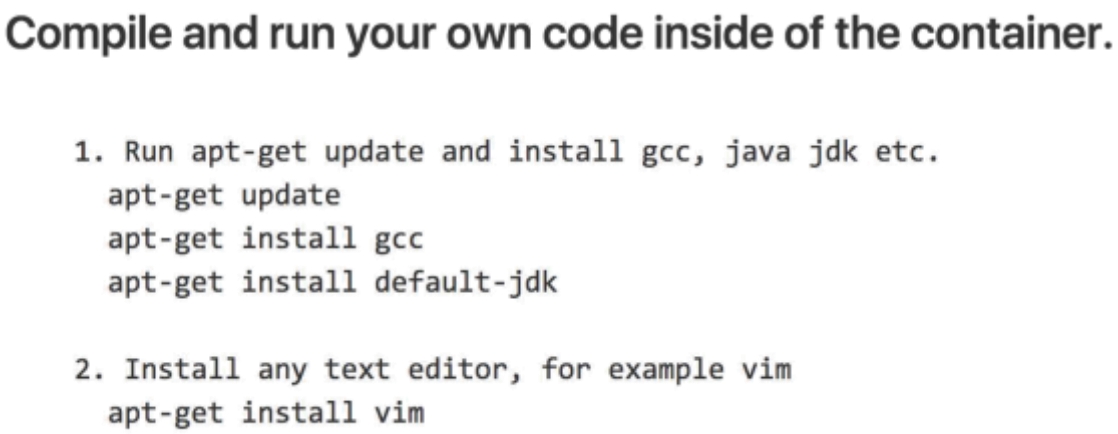


- Then we executed the code twice first without blocking it and then with blocking it.
- First we observed that the code executed the function and later, we found that the code was executed.
- This was done using the SECCOMP profile.





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



- 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.