Seccomp in Docker

Docker can also allow you to filter system calls (syscalls) with seccomp. This has two purposes:

- Restrict what a compromised program can do
- Reduce the kernel's attack surface

Seccomp is available through other tools as well. Docker makes this easier, but it's not very easy. If you find this process too time-intensive, we recommend you stick with the allowlist of syscalls provided by Docker whenever the container isn't "privileged."

Creating seccomp Profiles

Jess Frazelle has led much of the seccomp work. Her blog post :

https://github.com/jessfraz/blog/blob/master/content/post/how-to-use-new-docker-seccomp-profiles.md

While the slides and exercise use her shell script, the one noted in this blog post is more featureful and intended for longer-term use.

https://prefetch.net/blog/2017/11/27/securing-systemd-services-with-seccomp-profiles/

Also, Slim Toolkit (nee Docker-Slim) can minify container images and make system call lists.

https://slimtoolkit.org/

Step 1: Dockerfile

```
FROM centos:7
RUN yum –y update
RUN yum –y install strace
RUN yum –y install vsftpd
RUN cat /etc/vsftpd/vsftpd.conf | sed \
‘s/#write_enable=YES/write_enable=YES/g’ | sed \
‘s/#anon_upload_enable=YES/anon_upload_enable=YES/g’ \
>/etc/vsftpd/vsftpd.conf.2
RUN echo “anon_umask=000” >>/etc/vsftpd/vsftpd.conf.2
RUN mv /etc/vsftpd/vsftpd.conf.2 /etc/vsftpd/vsftpd.conf
RUN chmod go+rw /var/ftp/pub/
EXPOSE 21/tcp
ENTRYPOINT [“/usr/bin/strace”]
CMD [“-ff”,”vsftpd”]
```

Step 2: Build Docker Image

Build a Docker image from that Dockerfile:

```
# docker build -t generate-strace-vsftpd .
```

Start a container based on the image.

```
# docker run -d  --security-opt seccomp=unconfined --name test-vsftpd \
generate-strace-vsftpd
```

Use docker-inspect to get the container's IP address.

```
# docker inspect test-vsftpd
```

Step 3: Exercise vsftpd

```
# ftp 172.17.0.2
…
Name (172.17.0.2:root): anonymous
Password: jay@harden-linux.com
230 Login successful.
…
ftp> cd /pub
250 Directory successfully changed.
ftp> ls
226 Directory send OK.
ftp> put Dockerfile
…
ftp> lcd ..
…
ftp> get Dockerfile
ftp> exit
```

Step 4: Parse Logs to Profile

Capture the strace output into vsftpd-strace.log:

```
# docker logs vsftpd > vsftpd-strace.log 2>&1
```

Convert the strace output to a syscall profile:

```
# seccomp-profile-generator.sh vsftpd-strace.log >/root/seccomp.json
```

Try the new seccomp profile.

```
# docker run -d  --security-opt seccomp=/root/seccomp.json \
--name try-vsftpd-seccompd generate-strace-vsftpd
```

Step 5: Manual Steps

Unfortunately, the automatically-generated set of syscalls isn't always complete. You can iterate through errors until you can find a set of syscalls that's complete.

Also, you can start with Docker's built-in allowlist.

Read more about what it blocks here:

https://docs.docker.com/engine/security/seccomp/#significant-syscalls-blocked-by-the-default-profile

Exercise: RickMorty

We'll use a simulated "trojan horse" vulnerability to break into this virtual machine. Then we'll use seccomp to confine the trojan horse program to its expected functionality.

Please:

Open the Firefox browser on the class machine to:
http://localhost:10000/exercises/rickmorty-seccomp

Running the Pods Containers with a Seccomp Profile

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-seccomp-profile
spec:
  containers:
  - name: mycontainer
    image: myimage
    securityContext:
      runAsUser: 1000
      runAsGroup: 1000
      seccompProfile:
        type: Localhost
        localhostProfile: profile-allow.json
      capabilities:
        drop: [ "all" ]
        add: ["CAP_NET_RAW"]
```