class-vmware-docker-2018-03-12

vmware advanced docker class page

View on GitHub

Verify seccomp

Run `docker info` and verify the seccomp profile in place.

Enable User namespacing

Let's enable the namespacing feature of the kernel by doing:

```
echo "{\"userns-remap\": \"play\"}" | sudo tee /etc/docker/daemon.json
```

Then we restart docker:

```
sudo /etc/init.d/docker restart
```

Now run a container, e.g. nginx:

```
docker run -d nginx
```

Look at the userids from the host perspective:

```
ps aux |grep nginx
```

You can disable usernamespacing by deleting the daemon file and restarting docker:

```
sudo rm /etc/docker/daemon.json
sudo /etc/init.d/docker restart
```

Content Trust

Do a docker pull like:

```
docker pull alpine
```

Enable content trust in your current shell:

```
export DOCKER_CONTENT_TRUST=1
```

Now run another docker pull:

```
docker pull nginx
```

And note how the output changes.

We cannot `docker push` content without repository authentication, so we will skip that step. If you continue later on your own you will be prompted to enter passphrases to protect your signing keys.

You can unset the docker trust variable with this command:

```
unset DOCKER_CONTENT_TRUST
```

Running as non-root

You may recall from lab 3 that you can build a Go program as a static binary and, using multi-stage builds, insert it on the scratch image like so:

```
FROM golang:1.9-alpine

COPY main.go /go/src/github.com/chrishiestand/docker-go-hello-world/

RUN cd /go/src/github.com/chrishiestand/docker-go-hello-world && \
    go get && \
    CGO_ENABLED=0 GOOS=linux go build -a -o /go/bin/hello main.go

FROM scratch
CMD ["/app"]
COPY --from=0 /go/bin/hello /app
```

Now to improve security further, let's make this application not run as root. You will need to use the `USER` dockerfile command. You will also need an `/etc/passwd` file in your new image. You can get one from a number of places, or create one, but it's probably simplest now to copy the file from the

golang build container. To find out what unprivileged users are in that file you can `docker run` the golang build image and `cat /etc/passwd`.

Linux Capabilities

Run this command to see the current linux capabilities:

```
sudo /sbin/capsh --print
```

Now start this container that immediately installs the capsh program:

```
docker run -it debian /bin/sh -c "apt-get update && apt-get install -qq libcap2-bin
```

And then from inside the container run:

```
/sbin/capsh --print
```

Note the difference between the output on the host and inside the container. You can also try starting containers with dropped capabilities:

```
docker run --cap-drop=net_raw  -it debian /bin/sh -c "apt-get update && apt-get inst
```

Note the dropped capability is missing from the output inside the container.
