Docker for Pentester: Image Vulnerability Assessment

September 11, 2020 By Raj Chandel

We are moving from virtualization to containerization and we are all familiar with the container services such as docking or quay.io. You can pick a dock image for a particular application by selecting several choices. As you know, when a developer works with a container, it not only packs the program but is part of the OS, and we do not know whether the connect libraries have been patched or vulnerable.

So, we will show "**how to perform a container audit and vulnerability assessment**" in any infrastructure in this role.

Table of Contents

Prerequisites

At your host machine, Install docker and pull an image, you want to scan.

Clair: Vulnerability

Installation

CoreOS has created an awesome container scan tool called Clair. Clair is an open-source project for the static analysis of vulnerabilities in apps and Docker containers. You can clone the package with the help of git, using the following command

```
git clone https://github.com/arminc/clair-scanner.git
```

```
root@ubuntu:~# git clone https://github.com/arminc/clair-scanner.git
Cloning into 'clair-scanner'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 405 (delta 28), reused 33 (delta 13), pack-reused 346
Receiving objects: 100% (405/405), 105.88 KiB | 337.00 KiB/s, done.
Resolving deltas: 100% (229/229), done.
root@ubuntu:~#
```

The scanner is developed in go language, therefore golang on your local machine over which is docker is running.

```
apt install golang
```

```
root@ubuntu:~# apt install golang
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installe
  libllvm9
Use 'sudo apt autoremove' to remove it.
The following additional packages will be instal
  golang-1.13 golang-1.13-doc golang-1.13-go gol
Suggested packages:
  bzr | brz mercurial subversion
```

Build the library to install all dependencies of the Clair.

```
cd clair-scanner
make build
```

```
root@ubuntu:~/clair-scanner# make build
CGO_ENABLED=0 go build
go: downloading gopkg.in/yaml.v2 v2.0.0-20170812160011
go: downloading github.com/docker/docker v1.13.1
go: downloading github.com/coreos/clair v2.0.7+incompa
go: downloading github.com/olekukonko/tablewriter v0.0
go: downloading github.com/jawher/mow.cli v1.0.2
go: downloading github.com/mbndr/logo v0.0.0-201709221
go: extracting github.com/olekukonko/tablewriter v0.0.
```

```
make cross
```

```
root@ubuntu:~/clair-scanner# make cross
Sending build context to Docker daemon   2.56kB
Step 1/9 : FROM debian:jessie
jessie: Pulling from library/debian
0cd7281e66ed: Pull complete
Digest: sha256:38a0be899b925afb5524130771850d3d8dd00619a6b50171
Status: Downloaded newer image for debian:jessie
 ---> a3590c0e9ff9
Step 2/9 : RUN apt-get update -y && apt-get install --no-instal
 ---> Running in 9135f9f9f65e
Get:1 http://security.debian.org jessie/updates InRelease [44.9
```

As you can see, we have the following file in the bucket list.

```
root@ubuntu:~/clair-scanner# ls
clair.go      dist    docker.go   example-whiteli
clair-scanner docker  example-alpine.yaml  go.mod
```

If in your host machine, you don't have a docker image, you can pull a new image, as we did here to illustrate vulnerability assessment.

```
docker pull ubuntu:16.04
```

```
root@ubuntu:~# docker pull ubuntu:16.04
16.04: Pulling from library/ubuntu
6aa38bd67045: Pull complete
981ae4862c05: Pull complete
5bad8949dcb1: Pull complete
ca9461589e70: Pull complete
Digest: sha256:69bc24edd22c270431d1a9e6dbf57cfc4a77b2da19
Status: Downloaded newer image for ubuntu:16.04
docker.io/library/ubuntu:16.04
```

Now, run the docker image of the Clair that will listen at local port 5432.

```
docker run -d -p 5432:5432 --name db arminc/clair-db:latest
```

```
root@ubuntu:~# docker run -d -p 5432:5432 --name db arminc/clair-db:latest
Unable to find image 'arminc/clair-db:latest' locally
latest: Pulling from arminc/clair-db
c9b1b535fdd9: Pull complete
d1030c456d04: Pull complete
d1d0211bbd9a: Pull complete
07d0560c0a3f: Pull complete
ce7fd4584a5f: Pull complete
63eb0325fe1c: Pull complete
b67486507716: Pull complete
f58de2b85820: Pull complete
ca982626dd56: Pull complete
05a443c21363: Pull complete
Digest: sha256:436b969073a4f4633097fbf32f16a5f8a70bd5882db2e90d3e8abf632899200d
Status: Downloaded newer image for arminc/clair-db:latest
09aa495578dcaac1507d79ed2f80235e0430a0942c5df7c5fb2f19ca63859511
```

Also, run the docker image for postgres to link Clair scan with the help of the following command.

```
docker run -p 6060:6060 --link db:postgres -d --name clair arminc/clair-local-scan
```

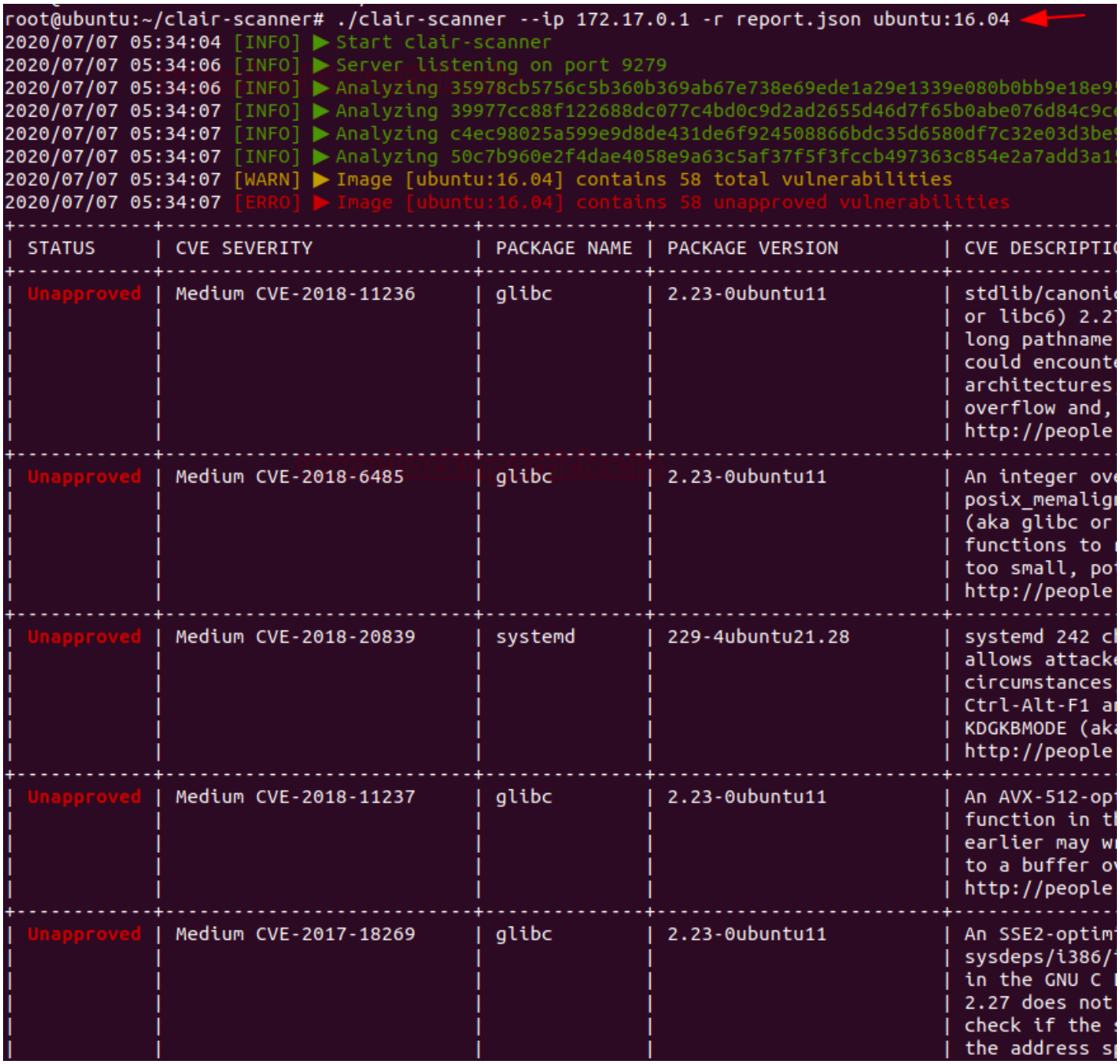
```
root@ubuntu:~# docker run -d -p 6060:6060 --link db:postgres --name clair arminc/clair-local-scan:latest
Unable to find image 'arminc/clair-local-scan:latest' locally
latest: Pulling from arminc/clair-local-scan
89d9c30c1d48: Pull complete
8ef94372a977: Pull complete
1ec62c064901: Pull complete
a47b1e89d194: Pull complete
bf1a3d234800: Pull complete
e86df44ff081: Pull complete
e4ea05d3fe20: Pull complete
db83214ca2c8: Pull complete
763b27721a28: Pull complete
Digest: sha256:482133d9ce156a59572d4b34035a8254312645bd6475d80b4385d0ee473f4ad1
Status: Downloaded newer image for arminc/clair-local-scan:latest
8eedf7ecbb110e631e50d23f8474d46f1ef2cfed0d6c3aa32429b127e6da64c8
root@ubuntu:~#
```

Now, let’s use the Clair for scanning the vulnerability of a container or docker image, with the help of the following command.

Syntax: ./clair-scanner -ip <docker ip> -r output.jason <docker-image>

```
./clair-scanner --ip 172.17.0.1 -r report.json ubuntu:16.04
```

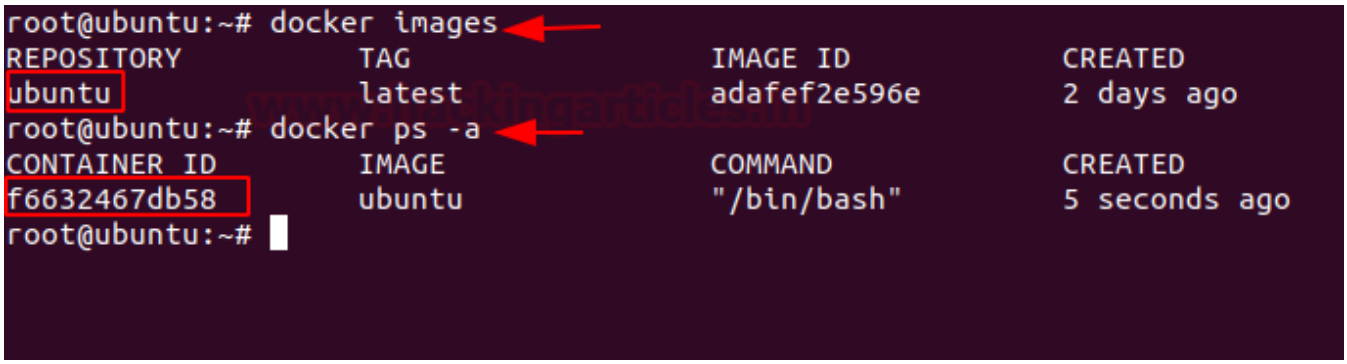
Booom!!!! And we got the scanning output which is showing 50 unapproved vulnerabilities.



Bench-Security: Container Hardening

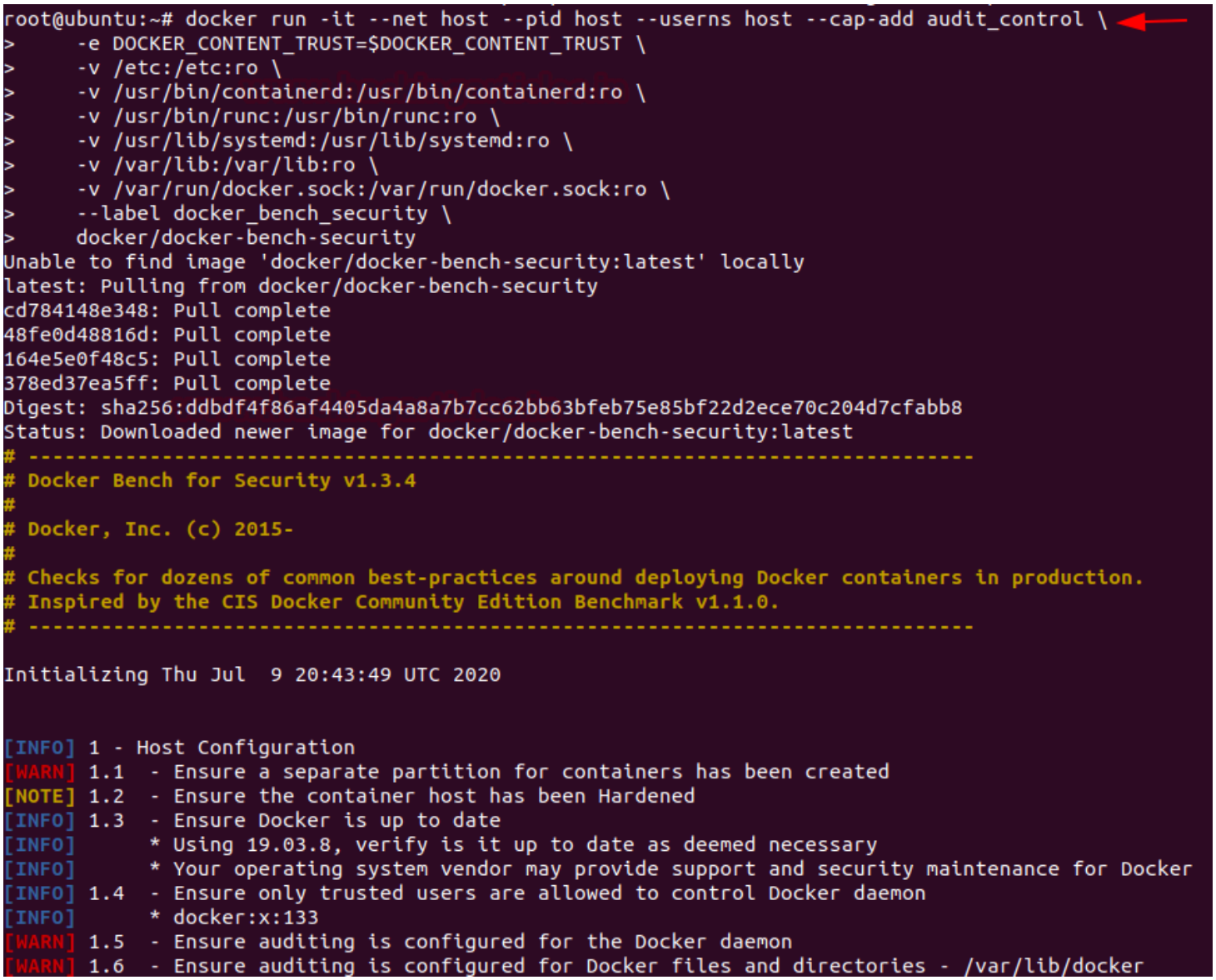
The Docker Bench for Security is a script that checks for dozens of common best-practices around deploying Docker containers in production. The tests are all automated and are inspired by the **CIS Docker Benchmark v1.2.0**.

So, as you can see, we have a few docker images on our host.



Let’s start docker audit for container hardening by executing a set of command as shown here.

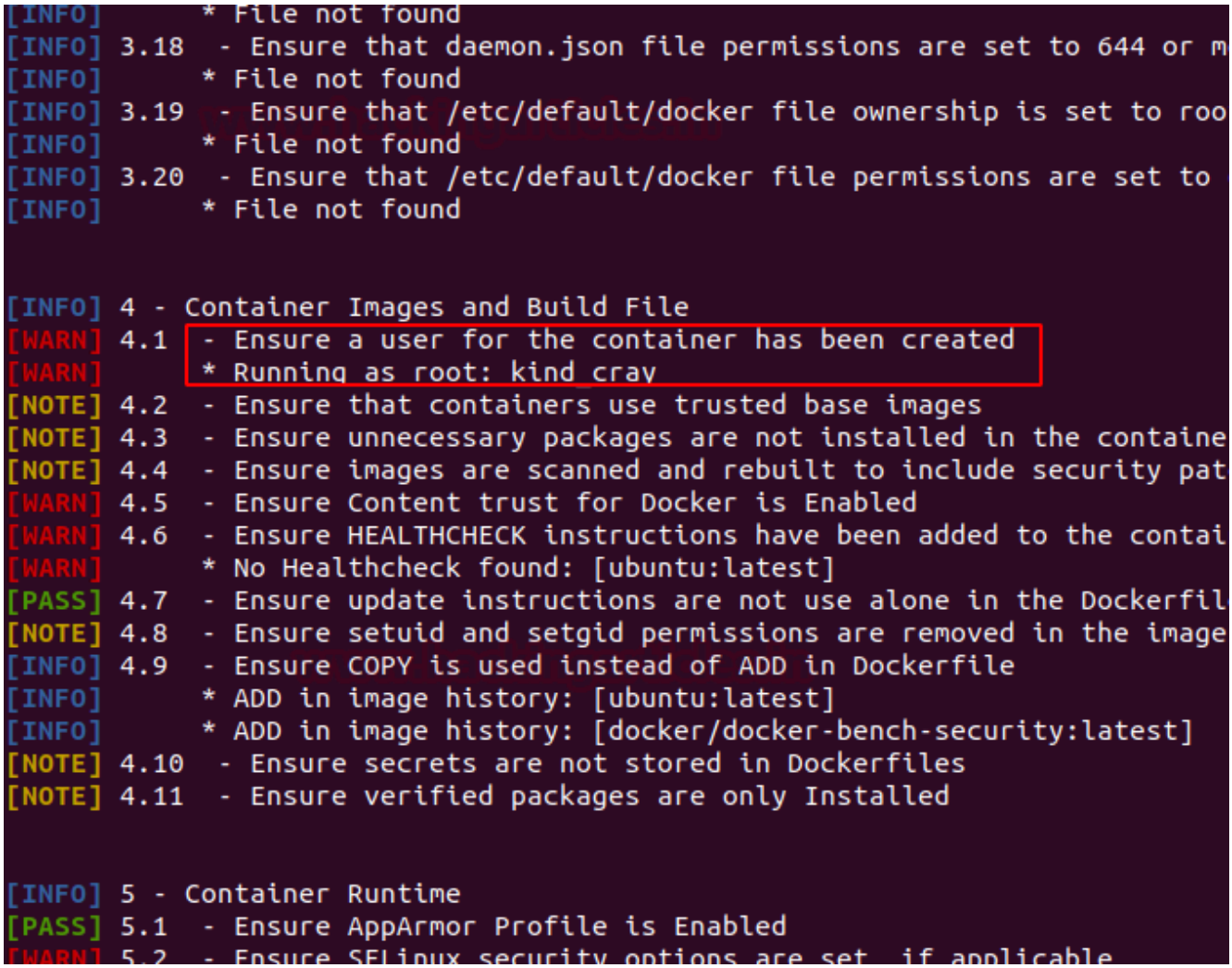
```
docker run -it --net host --pid host --userns host --cap-add audit_control \
    -e DOCKER_CONTENT_TRUST=$DOCKER_CONTENT_TRUST \
    -v /etc:/etc:ro \
    -v /usr/bin/containerd:/usr/bin/containerd:ro \
    -v /usr/bin/runc:/usr/bin/runc:ro \
    -v /usr/lib/systemd:/usr/lib/systemd:ro \
    -v /var/lib:/var/lib:ro \
    -v /var/run/docker.sock:/var/run/docker.sock:ro \
    --label docker_bench_security \
    docker/docker-bench-security
```



The output results as **Info**, **Warning**, **Pass** and **Notes** for each of the configuration recommendations as mention below:

1. Host Configuration
2. Docker Daemon Configuration
3. Docker Daemon Configuration Files
4. Container Images and Build Files
5. Container Runtime
6. Docker Security Operations

Let me explain this in a better way: You can observe in the highlighted session that it has created alert against root privilege for running the docker image.



To fix such type of misconfiguration, stop the running process for docker and then again, run the docker image with low privilege user access as shown below.

```
docker stop $(docker ps -aq)
docker rm $(docker ps -aq)
docker run -itd --user 1001:1001 ubuntu
docker exec -it <container-id> /bin/bash
```

```
root@ubuntu:~# docker stop $(docker ps -aq)
b4787ed862a3
f6632467db58
root@ubuntu:~# docker rm $(docker ps -aq)
b4787ed862a3
f6632467db58
root@ubuntu:~# docker run -itd --user 1001:1001 ubuntu
dce6a6c791c7627260f5b971a5303287e98d73d524266ada5dbdcb8e78ebc73b
root@ubuntu:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
dce6a6c791c7        ubuntu              "/bin/bash"         17 seconds ago
root@ubuntu:~# docker exec -it dce6a6c791c7 /bin/bash
groups: cannot find name for group ID 1001
I have no name!@dce6a6c791c7:/$ id
uid=1001 gid=1001 groups=1001
I have no name!@dce6a6c791c7:/$
```

If the loophole is closed, use the bench again for cross-validation and this time ensure you have passed the warning.

As you can see, this time we got the Green sign that shows we got the loopholes patched.

```
[INFO]       * File not found
[INFO] 3.20  - Ensure that /etc/default/docker file permissions are set to
[INFO]       * File not found


[INFO] 4 - Container Images and Build File
[PASS] 4.1   - Ensure a user for the container has been created
[NOTE] 4.2   - Ensure that containers use trusted base images
[NOTE] 4.3   - Ensure unnecessary packages are not installed in the contain
[NOTE] 4.4   - Ensure images are scanned and rebuilt to include security pa
[WARN] 4.5   - Ensure Content trust for Docker is Enabled
[WARN] 4.6   - Ensure HEALTHCHECK instructions have been added to the conta
[WARN]       * No Healthcheck found: [ubuntu:latest]
[PASS] 4.7   - Ensure update instructions are not use alone in the Dockerfi
[NOTE] 4.8   - Ensure setuid and setgid permissions are removed in the imag
```