DOCKER SECURITY WORKSHOP

Mohit Gupta

F-Secure

$ DOCKER INFO

MOHIT GUPTA

@_Skybound

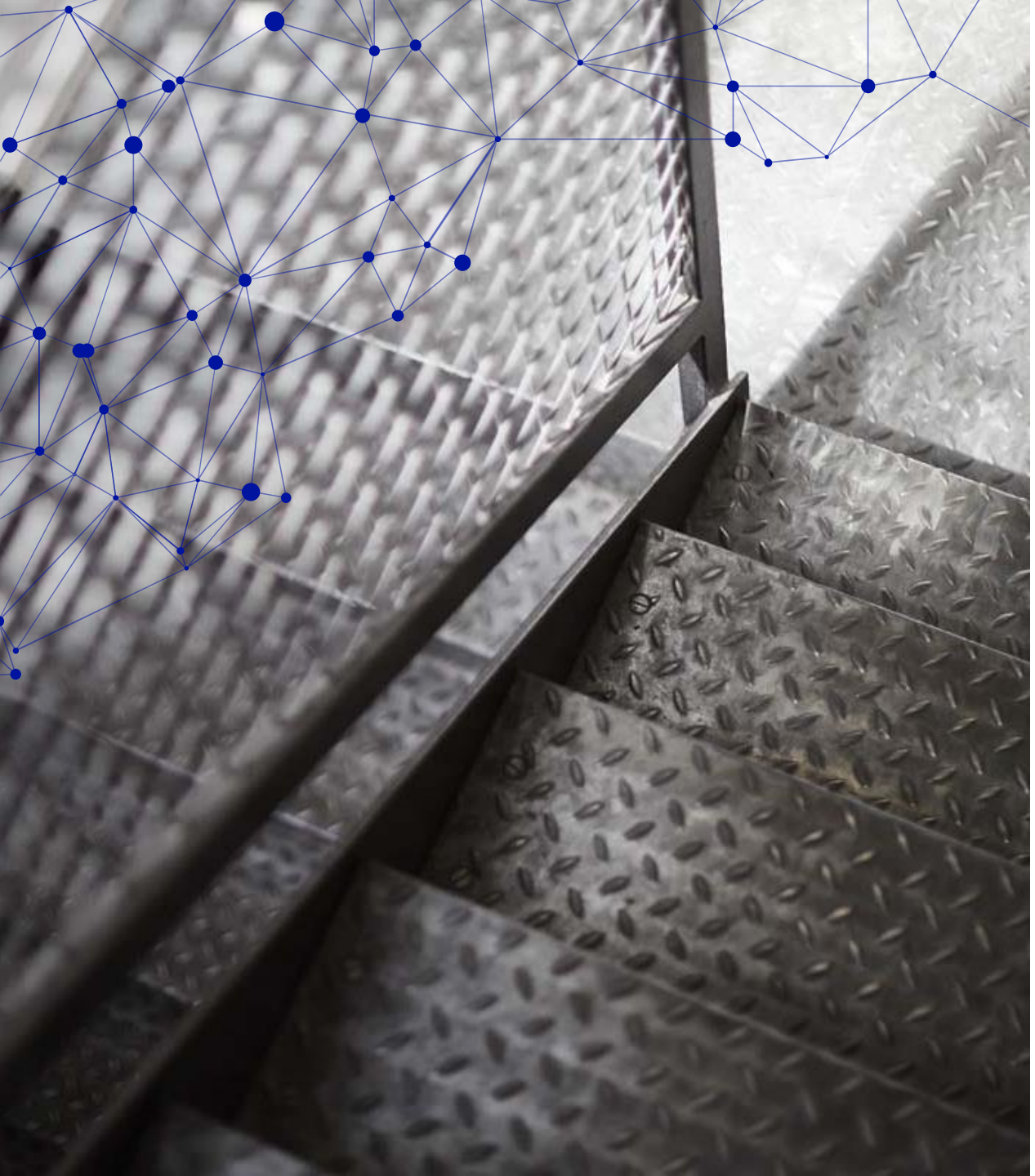
Security Consultant at F-Secure Consulting (formerly MWR InfoSecurity)

F-Secure

PRACTICE VM

- Can be exploited in multiple ways
- Training ground
- Compose files present
- PLAY!
- vagrant / vagrant

F-Secure

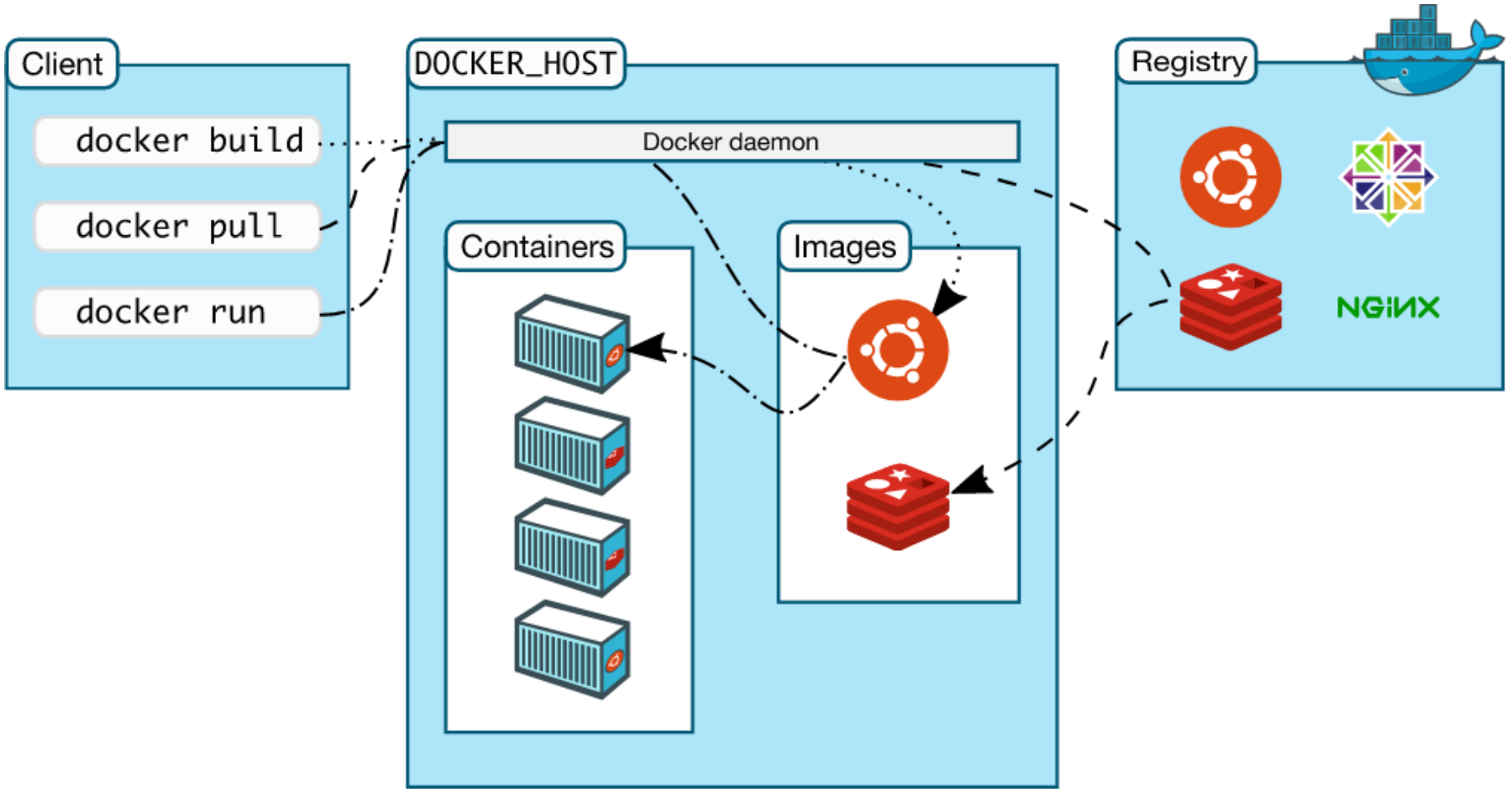


$ DOCKER PS

1. Docker Daemon
2. Registries
3. UID Namespaces
4. Host Volumes
5. Network Stack
6. Linux Capabilities
7. Multi stage builds

F-Secure

① DOCKER DAEMON



https://docs.docker.com/engine/docker-overview/

1 DOCKER DAEMON INTRODUCTION

REST API

- *Resources*
- *HTTP verbs*
  - *GET*
  - *POST*
  - *DELETE*
  - *etc*

UNIX Socket

- *root* user
- *docker* group
- */var/run/docker.sock*

TCP Port

- tcp/2375
- tcp/2376

F-Secure

1 DOCKER DAEMON INTRODUCTION

List running containers

```
$ docker -H tcp://127.0.0.1:2375 ps
```

Create a container with access to host filesystem

```
$ docker -H tcp://127.0.0.1:2375 run --rm -ti -v /:/host ubuntu bash
```

DOCKER DAEMON PRACTICE

1. Access docker daemon via UNIX socket within the container
2. Access docker daemon via TCP socket from within the container
3. Print the hosts /etc/shadow file from within the container

1 DOCKER DAEMON WHAT CAN YOU DO?

Control access to daemon

- docker group
- Limit exposure to containers
- Limit exposure on the network

Authorisation plugin

- Forwards request to plugin
- Plugin validation authorisation

Rootless docker

- Experimental in 19.03
- Caveats

2 REGISTRY

Central Storage

Image

- Manifest
- Config
- Image Layers

Tags

F-Secure

2 REGISTRY

Can view:

- Manifest
- Individual layers
- Images
- Tags

```
$ reg ls -k -f localhost

Repositories for localhost

REPO                    TAGS

random_image                    latest
```

```
$ reg tags -k -f
localhost/random_image

latest
```


F-Secure

REGISTRY DEMO

REGISTRY PRACTICE

1. Find the flag (a UUID) within each of the docker images within the registry

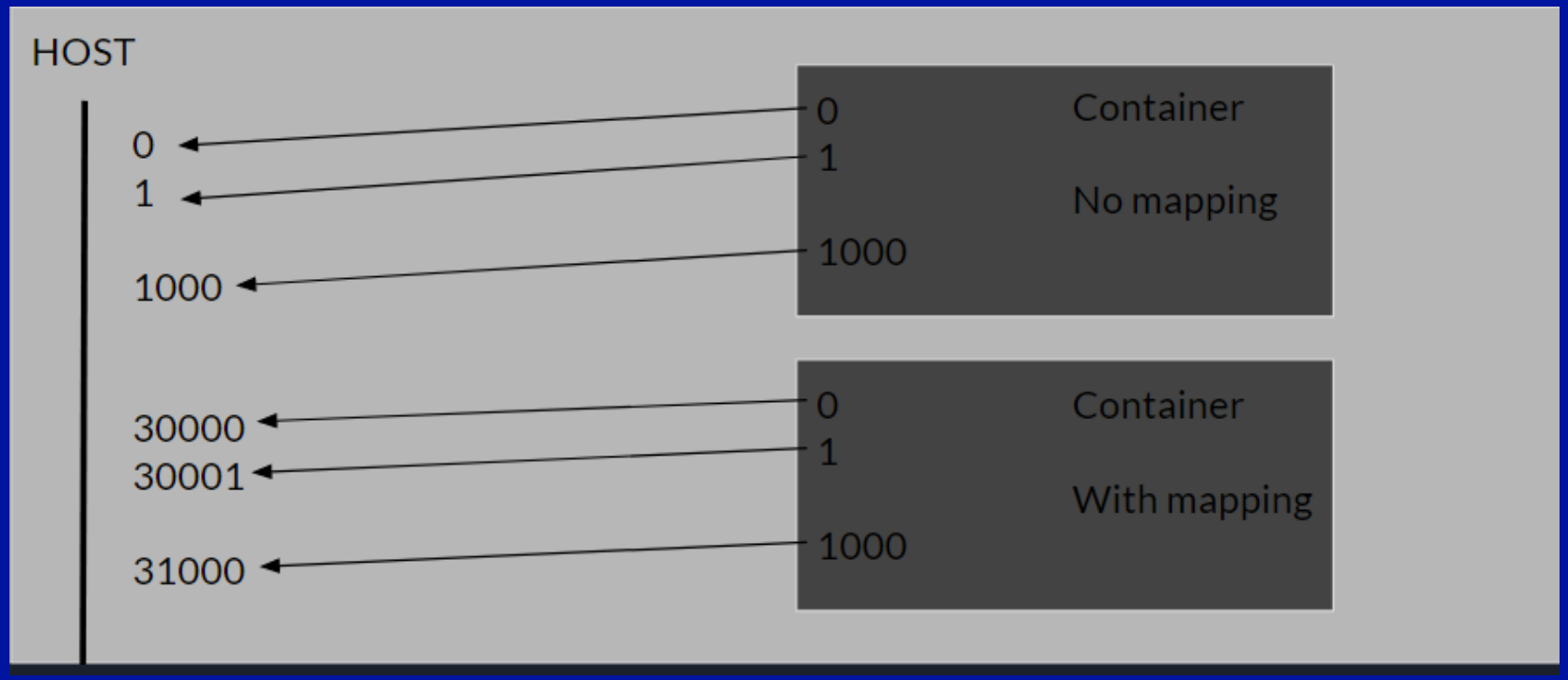
3 UID NAMESPACE INTRODUCTION

Container process UID == Host process UID

```
$ docker run --rm -ti ubuntu bash
root@9150696b8bb4:/# whoami
root
root@9150696b8bb4:/# id
uid=0(root) gid=0(root) groups=0(root)
root@9150696b8bb4:/# sleep 1d
```

```
$ ps aux | grep sleep
root   2076 0.0  0.0 4528  800 pts/0 S+  21:41 0:00 sleep 1d
```

3 UID NAMESPACE



F-Secure

3 UID NAMESPACE CAUTION !

- Volumes
- Sharing namespaces
- Privileged mode
- CLI argument

4 HOST VOLUME INTRODUCTION

- Access to host file system
- UID permissions
- docker inspect
- mount

F-Secure

4 HOST VOLUME WHAT CAN YOU DO?

- Control mounted directories / files
- Read only(?)

5 NETWORKING INTRODUCTION

Network stack

docker0 network

--net CLI argument

- Host
- Other Containers

Overlay networks

F-Secure

NETWORKING DEMO

NETWORKING PRACTICE

1. Communicate between two containers over a network, sharing the same network stack, with localhost communications only
2. Intercept communications between the netstack receiver and sender containers using a well placed net-utils container

6 CAPABILITIES INTRODUCTION

- Kernel level privileges
- Split super user privileges
- --cap-add
- --cap-drop

F-Secure

6 CAPABILITIES

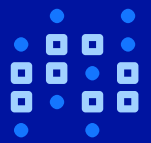
Capability Key	Capability Description
SETPCAP	Modify process capabilities.
MKNOD	Create special files using mknod(2).
AUDIT_WRITE	Write records to kernel auditing log.
CHOWN	Make arbitrary changes to file UIDs and GIDs (see chown(2)).
NET_RAW	Use RAW and PACKET sockets.
DAC_OVERRIDE	Bypass file read, write, and execute permission checks.
FOWNER	Bypass permission checks on operations that normally require the file system UID of the process to match the UID of the file.
FSETID	Don't clear set-user-ID and set-group-ID permission bits when a file is modified.
KILL	Bypass permission checks for sending signals.
SETGID	Make arbitrary manipulations of process GIDs and supplementary GID list.
SETUID	Make arbitrary manipulations of process UIDs.
NET_BIND_SERVICE	Bind a socket to internet domain privileged ports (port numbers less than 1024).
SYS_CHROOT	Use chroot(2), change root directory.
SETFCAP	Set file capabilities.

https://docs.docker.com/engine/reference/run/

6 CAPABILITIES INTRODUCTION

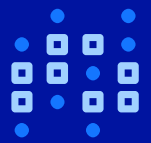
CAP_NET_RAW

- Raw packets
- Enabled by default



CAP_SYS_MODULE

- Load kernel modules
- Not enabled by default



CAPABILITIES PRACTICE

1. Within the capabilities container write, compile and load a kernel module to execute commands upon the host

6 CAPABILITIES WHAT CAN YOU DO?

- Drop all capabilities
- Add required
- Avoid privileged

F-Secure

7 MULTI STAGE BUILDS INTRODUCTION

- Keep images minimal
- Reduce attack surface

- Builder images
- Final image

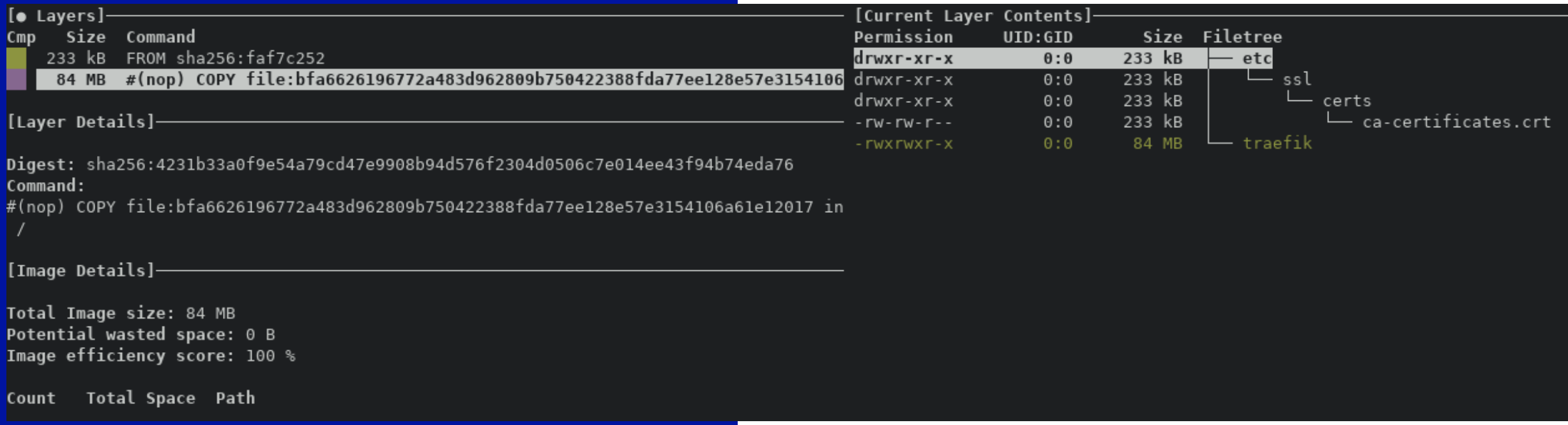
7 MULTI STAGE BUILDS INTRODUCTION

```
FROM golang AS builder
WORKDIR /go/src/github.com/example/example
COPY main.go .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .
```

```
FROM alpine
COPY --from=builder /go/src/github.com/example/example/main .
CMD [“./main”]
```

7 MULTI STAGE BUILDS INTRODUCTION

- scratch root container
- Statically compiled binary
- No other binaries to execute



F-Secure

QUESTIONS