

Quiz5_ Packet Sniffing and Spoofing Lab

Docker build and up as last one

```
ialsmadi@VM:~/Downloads/Lab5
ialis...@VM:~/Downloads/Lab5$ ls
Labsetup
Ove...
ialis...@VM:~/Downloads/Lab5$ cd Labsetup/
Env...
Codo...
docke...
compose.yml
volum...
Lab...
ialis...@VM:~/Downloads/Lab5$ sudo docker-compose build
to ...
attacker uses an image, skipping
Lab...
hostA uses an image, skipping
Pro...
hostB uses an image, skipping
Pac...
ialis...@VM:~/Downloads/Lab5$ sudo docker-compose up
Cr...
Creating network "net-10.9.0.0" with the default driver
Cr...
Creating hostA-10.9.0.5 ... done
Cr...
Creating hostB-10.9.0.6 ... done
Cr...
Creating seed-attacker ... done
Attaching to seed-attacker, hostB-10.9.0.6, hostA-10.9.0.5
hostB-10.9.0.6 | * Starting internet superserver inetd
hostA-10.9.0.5 | * Starting internet superserver inetd
[ OK ] [ OK ]

```

content in this file and all the involved Dock
to the website of this lab. If this is the first ti

Then docker ps to get the name of the hosts

CONTAINER ID	IMAGE	COMMAND	STATUS	PORTS	NAMES
75471937ee0	handsonsecurity/seed-ubuntu:large	"bash -c '/etc/init.d/inetd start'"	Up 18 minutes		seed-attacker
df28595892bd	handsonsecurity/seed-ubuntu:large	"bash -c '/etc/init.d/inetd start'"	Up 18 minutes		hostB-10.9.0.6
25af91680eff	handsonsecurity/seed-ubuntu:large	"bash -c '/etc/init.d/inetd start'"	Up 18 minutes		hostA-10.9.0.5

```
ialsmadi@VM:~/Downloads/Lab5/Labsetup$ cd ..
ialis...@VM:~/Downloads/Lab5$ sudo docker exec df28595892bd
\`docker exec\` requires at least 2 arguments.
See \`docker exec --help\`.

Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
ialis...@VM:~/Downloads/Lab5$ sudo docker ps
docker-compose.yml
ialis...@VM:~/Downloads/Lab5$ sudo docker exec handsonsecurity/seed-ubuntu:large
CONTAINER ID IMAGE COMMAND STATUS PORTS NAMES
75471937ee0 handsonsecurity/seed-ubuntu:large "/bin/sh" Up 18 minutes
df28595892bd handsonsecurity/seed-ubuntu:large "bash -c '/etc/init.d/inetd start'" Up 18 minutes
25af91680eff handsonsecurity/seed-ubuntu:large "bash -c '/etc/init.d/inetd start'" Up 18 minutes
ialis...@VM:~/Downloads/Lab5$ sudo docker exec handsonsecurity/seed-ubuntu:large
\`docker exec\` requires at least 2 arguments.
See \`docker exec --help\`.

Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
ialis...@VM:~/Downloads/Lab5$ sudo docker exec handsonsecurity/seed-ubuntu:large
Error: No such container: handsonsecurity/seed-ubuntu:large
ialis...@VM:~/Downloads/Lab5$ 
```

Make sure you note the names at the end

Next, execute an interactive bash shell on the container.

```
ialsmadi@VM:~/Downloads/Lab5/Labsetup$ docker exec -it seed-attacker bash
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: connect: permission denied
ialis...@VM:~/Downloads/Lab5$ sudo docker exec -it seed-attacker bash
root@VM:~# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var volumes
root@VM:~# 
```

```

talsmadi@VM:~$ cd Downloads/Labs/Labsetup/
talsmadi@VM:~/Downloads/Labs/Labsetup$ sudo docker exec it hostB-10.9.0.6 bash
Error: No such container: it
talsmadi@VM:~/Downloads/Labs/Labsetup$ sudo docker exec -it hostB-10.9.0.6 bash
root@df28595892bd:#

```

DRAFT

```

talsmadi@VM:~/Downloads/Labs/Labsetup$ docker exec -it seed-attacker bash
    permission denied while trying to connect to the Docker daemon socket at unix:///run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.40/containers/seed-attacker/json: dial unix /var/run/docker.sock: connect: permission denied
talsmadi@VM:~/Downloads/Labs/Labsetup$ sudo docker exec -it seed-attacker bash
root@VM:#

```

```

// The following example shows how to list all the volumes
$ dockeps
$ ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run
sbin srv sys tmp usr var volumes
$ root@VM:#

```

Then get the bridge ID for the host IP

```

br-095ad24dc462: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.28.0.1 netmask 255.255.0.0 broadcast 172.28.255.255
        ether 02:42:c9:55:d2:54 txqueuelen 0 (Ethernet)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
br-57ceea08d943: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.1 netmask 255.255.255.0 broadcast 10.9.0.255
        ether fe:80::42:5eff:fe4f:f946 txqueuelen 64 (link)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 47 bytes 7246 (7.2 kB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
br-a8019b0cd517: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.29.0.1 netmask 255.255.255.0 broadcast 172.29.255.255
        ether 02:42:95:25:59:6c txqueuelen 1000 (link)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)

```

DRAFT

In my case its br-57ceea08d943

Docker network command shows the same value

```

talsmadi@VM:~/Downloads/Labs/Labsetup$ sudo docker network ls
NETWORK ID     NAME      DRIVER      SCOPE
2dec11a0f59c   bridge    bridge      local
b3581338a28d   host      host      local
095ad24dc462   internet-nano_default  bridge      local
f5f75196813d   internet-nano_net_151_net0  bridge      local
b05e8996cb5c   internet-nano_net_152_net0  bridge      local
dic4cc4cf97a   internet-nano_net_153_net0  bridge      local
f3e5c7ae4edb   internet-nano_net_ix_ix100  bridge      local
a8019b0cd517   map_default  bridge      local
57ceea08d943   net-10.9.0.0  bridge      local
77aceccbbe26   none      null      local

```

----- Now we are ready for lab tasks -----

Lets complete only the first 4 tasks 1.1, 1.2, 1.3, 1.4

I will show you code for task1.py (Make sure you change interface based on yours)

```

#!/bin/env python
from scapy.all import *
print("SNIFFING PACKETS.....")
def print_pkt(pkt):
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)
    print("Protocol:", pkt[IP].proto)
    print("\n")
pkt = sniff(iface=br-57ceea08d943, filter='ip', prn=print_pkt)

```

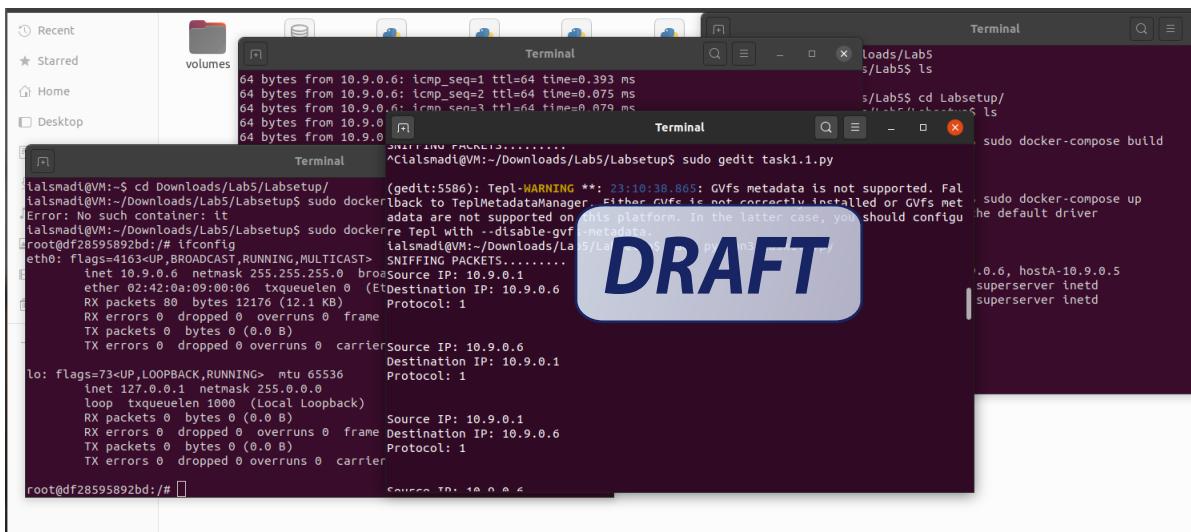
Lets run code of task 1.1

```
[  ] Terminal [  ] _ x
f5f75196813d      internet-nano_net_151_net0  bridge  local
b06e899bc5c      internet-nano_net_152_net0  bridge  local
d1c4cc6f97a      internet-nano_net_153_net0  bridge  local
f3e5c7aeed4b     internet-nano_net_ix_ix100 bridge  local
a8019b0cd517     map_default                bridge  local
57ceea08d943     net-10.9.0.0               bridge  local
77aceccbe26      none                      null    local
ialsmadi@VM:~/Downloads/Lab5$ ls
docker-compose.yml task1.1.py task1.2.py task1.3.py task1.4.py volumes
ialsmadi@VM:~/Downloads/Lab5$ sudo gedit task1.1

(gedit:5472): Tepl-WARNING **: 23:06:46.292: GVfs metadata is not supported. Fall back to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should config re Tepl with --disable-gvfs-metadata.
ialsmadi@VM:~/Downloads/Lab5$ sudo gedit task1.1.py

(gedit:5536): Tepl-WARNING **: 23:09:10.463: GVfs metadata is not supported. Fall back to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should config re Tepl with --disable-gvfs-metadata.
ialsmadi@VM:~/Downloads/Lab5$ sudo python3 task1.1.py
SNIFFING PACKETS.....
```

I am creating some traffic between the different machines so that my little sniffer can pick that



Please similarly, complete tasks, 1.2, 1.3 and 1.4