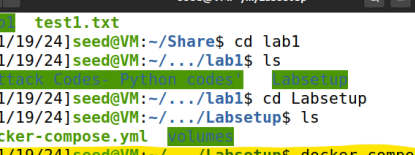


## 1. Setting up Host A and Host B using docker-compose up:



```

seed@VM: ~/..Labsetup
test1.txt
[01/19/24]seed@VM:~/Share$ cd lab1
[01/19/24]seed@VM:~/../Lab1$ ls
Attack-compose  python-compose  abs@VM
[01/19/24]seed@VM:~/../Lab1$ cd Labsetup
[01/19/24]seed@VM:~/../Labsetup$ ls
docker-compose.yml  python
[01/19/24]seed@VM:~/../Labsetup$ docker-compose up
Starting seed-attacker ... done
Starting hostB-10.9.0.6 ... done
Starting hostA-10.9.0.5 ... done
Attaching to seed-attacker, hostA-10.9.0.5, hostB-10.9.0.6
hostA-10.9.0.5 | * Starting internet superserver inet
d
[ OK ]
hostB-10.9.0.6 | * Starting internet superserver inet
d
[ OK ]
```

**Host A has the IP address 10.9.0.5 and Host B has the IP address 10.9.0.6, both hosts are now started.**

## 2. Finding my interface address using ifconfig

```

[01/19/24]seed@VM:~/.../Attack Codes- Python codes$ vi
m task1.1.py
[01/19/24]seed@VM:~/.../Attack Codes- Python codes$ if
config
br-5ac33773c377: flags=4163<UP,BROADCAST,RUNNING,MULTI
CAST> mtu 1500
inet 10.9.0.1 netmask 255.255.255.0 broadcast
t 10.9.0.255
inet6 fe80::42:77ff:fea0:9bdb prefixlen 64 s
copeid 0x20<link>
ether 02:42:77:a0:9b:db txqueuelen 0 (Ethernet)

RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 37 bytes 4636 (4.6 KB)
TX errors 0 dropped 0 overruns 0 carrier 0
collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

```

**This is my interface address and I found it using ifconfig**

### 3. Changing interface address in task1.1 python code

[illegible]

**I changed the interface address to mine in the python task1.1 code**

#### 4. Running task1.1 initially starts sniffing packets

```
Activities Terminal Jan 19 13:07
root@VM: /home/seed/Share/lab1/Attack Codes- Python ...
seed@VM: ~/.../Labsetup root@VM: /home/seed/Share/lab1/A...
vethe104e15: flags=4163<UP,BROADCAST,RUNNING,MULTICAST
> mtu 1500
    inet6 fe80::7ce7:dcff:fee0:84a8 prefixlen 64
    scopeid 0x20<link>
    ether 7e:e7:dc:e0:84:a8 txqueuelen 0 (Ethern
et)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 64 bytes 7770 (7.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0
    collisions 0

[01/19/24]seed@VM:~/.../Attack Codes- Python codes$ su
do su
root@VM: /home/seed/Share/lab1/Attack Codes- Python cod
es# python3 task1.1.py
SNIFFING PACKETS.....
```

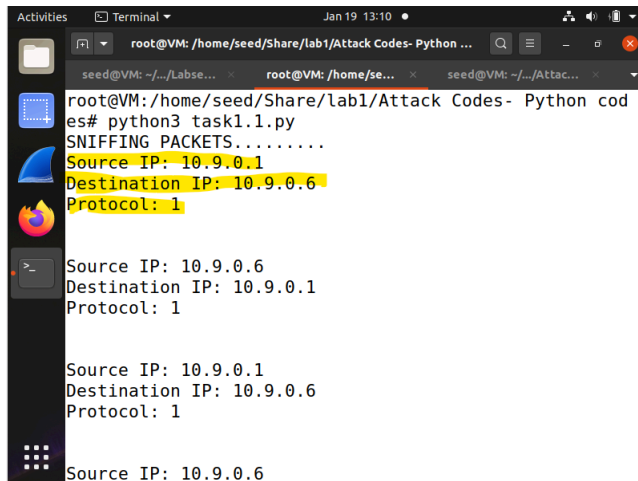
By using sudo su and then running task1.1 I was able to start SNIFFING PACKETS which means that I will be able to see activity such as pings between my LAN and the host

#### 5. Send a ping to 10.9.0.6

```
Activities Terminal Jan 19 13:09
seed@VM: ~/.../Attack Codes- Python codes
seed@VM: ~/.../Labse... root@VM: /home/se... seed@VM: ~/.../Attac...
[01/19/24]seed@VM:~/.../Attack Codes- Python codes$ pi
ng 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.130 m
s
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.255 m
s
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.074 m
s
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.108 m
s
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.258 m
s
64 bytes from 10.9.0.6: icmp_seq=6 ttl=64 time=0.084 m
s
64 bytes from 10.9.0.6: icmp_seq=7 ttl=64 time=0.103 m
s
64 bytes from 10.9.0.6: icmp_seq=8 ttl=64 time=0.122 m
s
```

I sent a ping from 10.9.0.6 to test if the packet sniffing was working

#### 6. Task1.1 is working, and it is showing the source IP and the destination IP of the sent ping

A terminal window titled 'root@VM: /home/seed/Share/lab1/Attack Codes- Python codes' showing the output of a Python script. The script has finished running, and the output displays sniffed packet information. The first packet shows Source IP: 10.9.0.1, Destination IP: 10.9.0.6, and Protocol: 1. The second packet shows Source IP: 10.9.0.6, Destination IP: 10.9.0.1, and Protocol: 1. The third packet shows Source IP: 10.9.0.1, Destination IP: 10.9.0.6, and Protocol: 1. The fourth packet shows Source IP: 10.9.0.6.

```
root@VM: /home/seed/Share/lab1/Attack Codes- Python codes# python3 task1.1.py
SNIFFING PACKETS.....
Source IP: 10.9.0.1
Destination IP: 10.9.0.6
Protocol: 1

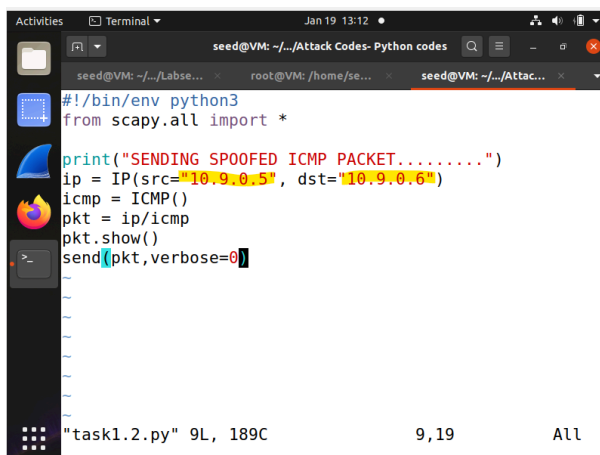
Source IP: 10.9.0.6
Destination IP: 10.9.0.1
Protocol: 1

Source IP: 10.9.0.1
Destination IP: 10.9.0.6
Protocol: 1

Source IP: 10.9.0.6
```

Task 1.1 is complete and working as shown above, the packets were sniffed from the ping and it shows the source IP and destination IP as well.

## 7. Setting up source and destination for task1.2

A terminal window showing the code for task1.2.py. The code imports scapy.all and sets up a spoofed ICMP packet from 10.9.0.5 to 10.9.0.6. The script is saved as task1.2.py with 9 lines and 189 characters.

```
#!/bin/env python3
from scapy.all import *

print("SENDING SPOOFED ICMP PACKET.....")
ip = IP(src="10.9.0.5", dst="10.9.0.6")
icmp = ICMP()
pkt = ip/icmp
pkt.show()
send(pkt, verbose=0)
```

I changed the source and destination IP addresses in the task1.2 code to the two hosts, A and B, I have running.

## 8. Sending the spoofed ICMP packet from 10.9.0.5 to 10.9.0.6

```
root@VM: /home/seed/Share/lab1/Attack Codes- Python ...  
root@VM: ~/.../Labse... root@VM: /home/se... root@VM: /home/se...  
root@VM: /home/seed/Share/lab1/Attack Codes- Python codes$ do su  
root@VM: /home/seed/Share/lab1/Attack Codes- Python codes# python3 task1.2.py  
SENDING SPOOFED ICMP PACKET.....  
###[ IP ]###  
version = 4  
ihl = None  
tos = 0x0  
len = None  
id = 1  
flags =  
frag = 0  
ttl = 64  
proto = icmp  
chksum = None  
src = 10.9.0.5  
dst = 10.9.0.6  
options \  
###[ ICMP ]###
```

I then sent a spoofed ICMP packet from 10.9.0.5 to destination 10.9.0.6 by running task1.2

## 9. Sniffing the spoofed ICMP packet

```
Source IP: 10.9.0.6  
Destination IP: 10.9.0.1  
Protocol: 1  
  
Source IP: 10.9.0.5  
Destination IP: 10.9.0.6  
Protocol: 1  
  
Source IP: 10.9.0.6  
Destination IP: 10.9.0.5  
Protocol: 1
```

I was still sniffing packets and as you can see above, task1.2 was successfully completed because the source IP is 10.9.0.5 and the destination IP is 10.9.0.6 which is exactly what I sent in step 8.

All of the work above is my own

Darion Kwasnitza - 3122890

Assignment 1