

Joshua Trzcienski

COMP 435

LAB 6 Report

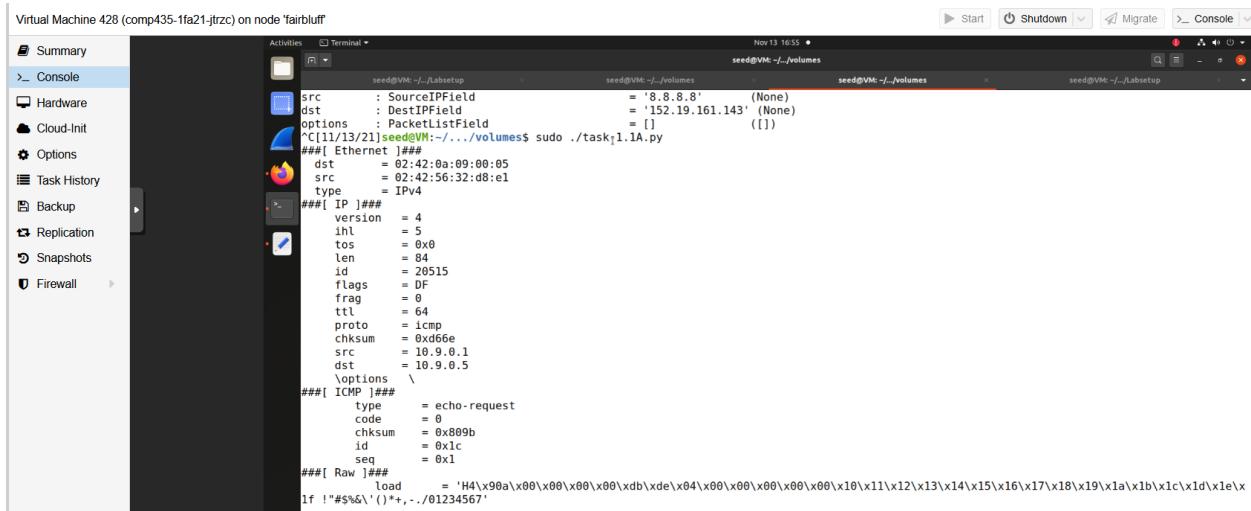
Packet Sniffing and Spoofing

Task 1.1: Sniffing Packets

1.1A

In 1.1A, 10.9.0.5 was pinged once, as seen in Image Two, and in a separate terminal task-1.1A.py was run, and in Image One is the captured packet that it picked up. The captured packet happened when task-1.1A.py was run with root, but when the same program was run without root, it was not able to run, seen in Image Three. The error that was given was Operation Not Permitted, and without running with root privilege, it could not capture any packets.

Image One



A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "seed@VM: ~./Labsetup". The terminal content shows a captured network packet. The packet details are as follows:

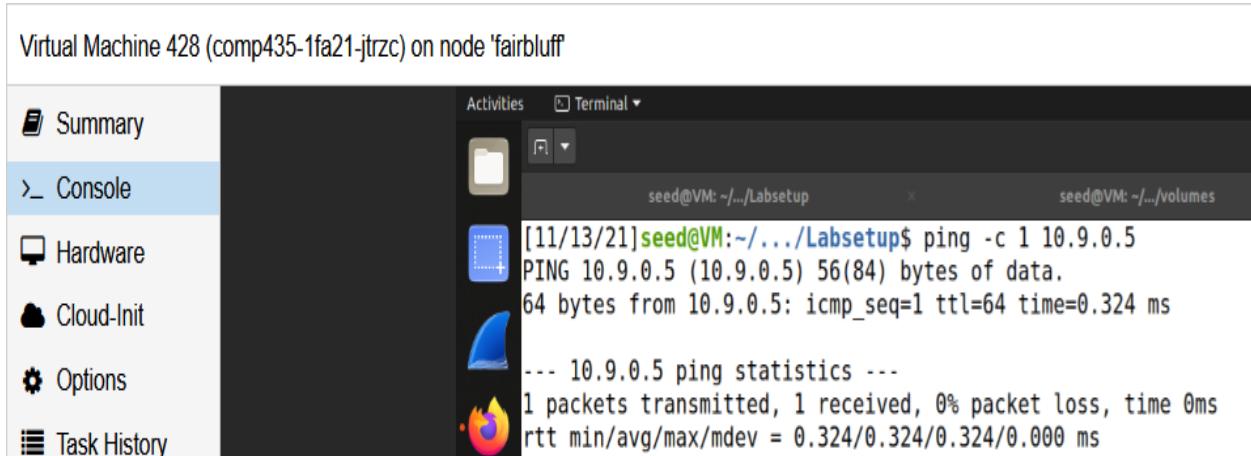
```
Nov 13 16:55 •
seed@VM: ~./volumes
seed@VM: ~./volumes
seed@VM: ~./volumes
seed@VM: ~./volumes

src      : SourceIPField           = '8.8.8.8'   (None)
dst      : DestIPField            = '152.19.161.143' (None)
options  : PacketListField        = []          ([])

#[[ Ethernet
    dst    = '02:42:80:09:00:05'
    src    = '02:42:56:32:d8:e1'
    type   = IPv4

##[ IP ]##
    version = 4
    ihl    = 5
    tos    = 0x0
    len    = 84
    id     = 20515
    flags  = DF
    frag   = 0
    ttl    = 64
    proto  = icmp
    checksum = 0xd66e
    src    = 10.9.0.1
    dst    = 10.9.0.5
    \options \
##[ ICMP ]##
    type   = echo-request
    code   = 0
    checksum = 0x809b
    id     = 0x1c
    seq    = 0x1
##[ Raw ]##
    load   = 'H\x90a\x00\x00\x00\x00\xdb\xde\x04\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !#$%& ('*+, -./01234567'
```

Image Two



A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "seed@VM: ~./Labsetup". The terminal content shows the output of a ping command:

```
[11/13/21]seed@VM:~/.../Labsetup$ ping -c 1 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.324 ms

--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.324/0.324/0.324/0.000 ms
```

Image Three

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

Activities Terminal Nov 13 13:46 seed@VM: ~/volumes

seed@VM:~/volumes\$./task-1.1A.py

```
[11/13/21]seed@VM:~/.volumes$ ./task-1.1A.py
Traceback (most recent call last):
  File "./task-1.1A.py", line 11, in <module>
    pkt = sniff(iface='br-aa5a4d58b935', filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in _run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type)) # noqa: E501
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

1.1B:

In 1.1B, now instead of the packet filtering only being set to ICMP, there are changes being made. The first filter was to capture any TCP packet that came from 10.9.0.1 with a destination port of 23. The filter used can be seen in Image Four, with Image Five showing the result when task-1.1B.py being run at the same time as the telnet command being run simultaneously, which is Image Six. Telnet is used to make a connection to TCP ports, and its default it port 23, so this is why it is used. Part 2 involved capturing all packets that come from or go to the subnet 10.9.0.0/24. The change to the filter can be seen in Image Seven, and one of the responses can be seen in Image Nine. The difference between this filter and 1.1A is that this filter will show more than just ICMP packets, and this one will show more than ICMP, for example Image Nine shows an ARP packet.

Image Four

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

Activities Text Editor Nov 13 14:34 task-1.1B.py

task-1.1B.py

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
##TODO:
# - Copy your iface value from running the ifconfig command below.
# - Modify the filter parameter to
#   (1) Capture any TCP packet the comes from a particular IP and with a destination port number 23.
#   (2) Capture packets that come from or go to the subnet
#       10.9.0.0/24
pkt = sniff(iface='br-aa5a4d58b935', filter='src host 10.9.0.1 and tcp dst port 23', prn=print_pkt)
```

Image Five

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

The screenshot shows the Oracle VM VirtualBox Manager interface. On the left is a sidebar with icons for Summary, Console, Hardware, Cloud-Init, Options, Task History, Backup, Replication, Snapshots, and Firewall. The 'Console' icon is selected and highlighted in blue. On the right is a terminal window titled 'Activities Terminal'. The terminal output shows a Python script named 'task-1.1B.py' being run. The script prints a detailed network configuration for an interface, including fields like dst, src, type, version, ihl, tos, len, id, flags, frag, ttl, proto, checksum, src, dst, options, sport, and dport. The output is as follows:

```
options = [('NOP', None), ('NOP', None), ('Timestamp', None)]
^C[11/13/21]seed@VM:~/.../volumes$ sudo ./task-1.1B.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:06
src      = 02:42:56:32:d8:e1
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x10
len      = 53
id       = 61177
flags    = DF
frag     = 0
ttl      = 64
proto   = tcp
checksum = 0x37a1
src      = 10.9.0.1
dst      = 10.9.0.6
options  =
###[ TCP ]###
sport    = 40674
dport    = telnet
```

Image Six

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

The screenshot shows the Oracle VM VirtualBox Manager interface, similar to Image Five. The 'Console' icon is selected in the sidebar. On the right is a terminal window titled 'Activities Terminal'. The terminal output shows a telnet session to the VM's IP address 10.9.0.6. The session starts with 'Trying 10.9.0.6...', followed by 'Connected to 10.9.0.6.', 'Escape character is '^]'.', and 'Ubuntu 20.04.1 LTS'. It then prompts for a password with '3fe18ec47b50 login: seed'. After logging in, it displays the welcome message 'Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)'. Below this, it lists documentation, management, and support links. At the bottom, it states that the system has been minimized and provides instructions to restore content using the 'unminimize' command. The terminal also shows the last login information.

```
[11/13/21]seed@VM:~/.../Labsetup$ telnet 10.9.0.6 23
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
3fe18ec47b50 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Nov 13 19:14:31 UTC 2021 from 10.9.0.1 on pts/1
```

Image Seven

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

The screenshot shows a Linux desktop interface with a terminal window open. The terminal window title is "task-1.1B.py". The code inside the terminal is:

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7
8
9##TODO:
10# - Copy your iface value from running the ifconfig command below.
11# - Modify the filter parameter to
12#   (1) Capture any TCP packet that comes from a particular IP and with a destination port number 23.
13#   (2) Capture packets that come from or go to the subnet
14#       10.9.0.0/24
15#       Make sure you take screenshots
16#
17#       (1) Capture any TCP packet that comes from a particular IP and with a destination port number 23.
18#       (2) Capture packets that come from or go to the subnet
19#           10.9.0.0/24
20
21pkt = sniff(iface='br-aa5a4d58b935', filter='dst net 10.9.0.0/24', prn=print_pkt)
```

Image Eight

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

The screenshot shows a Linux desktop interface with a terminal window open. The terminal window title is "seed@VM: ~/.../Labsetup". The output of the ping command is:

```
[11/13/21]seed@VM:~/.../Labsetup$ ping 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=0.414 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.119 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.119/0.266/0.414/0.147 ms
```

Image Nine

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

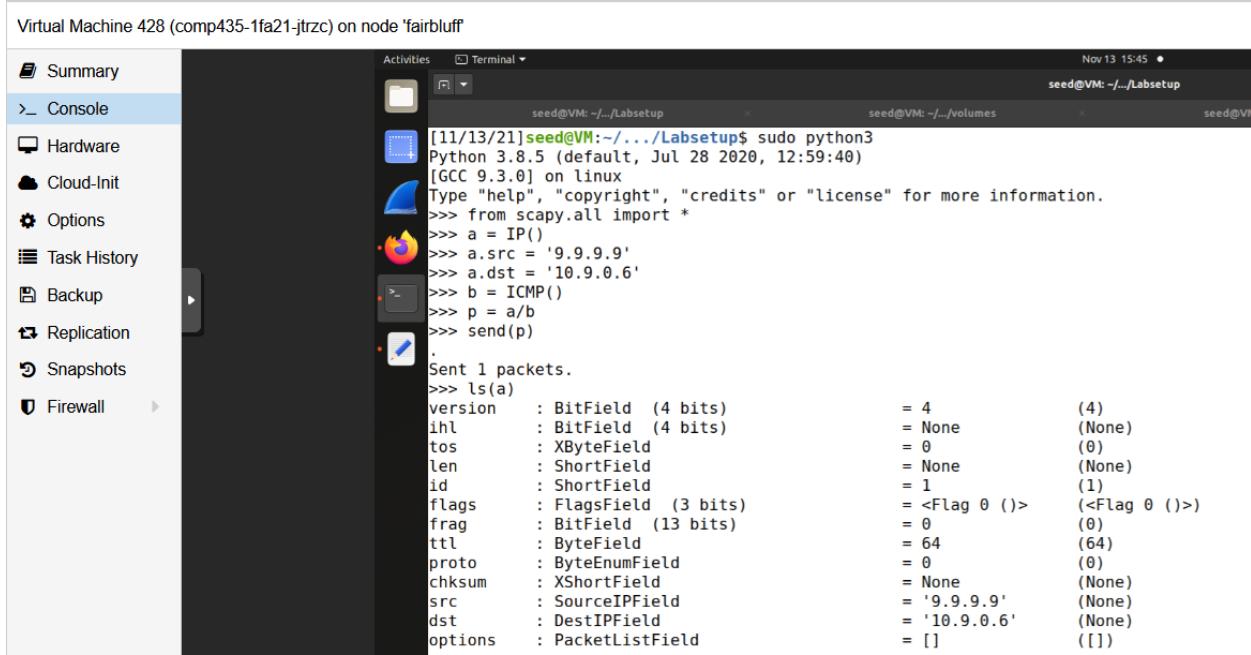
The screenshot shows a Linux desktop interface with a terminal window open. The terminal window title is "seed@VM: ~/.../Labsetup". The output of the scapy command is:

```
hwdst      = 00:00:00:00:00:00
pdst       = 10.9.0.6
###[ Ethernet ]###
dst        = 02:42:56:32:d8:e1
src        = 02:42:0a:09:00:06
type       = ARP
###[ ARP ]###
hwtype     = 0x1
ptype      = IPv4
hwlen      = 6
plen       = 4
op         = who-has
hwsrc     = 02:42:0a:09:00:06
psrc       = 10.9.0.6
hwdst     = 00:00:00:00:00:00
pdst       = 10.9.0.1
```

Task 1.2: Spoofing ICMP packets

Image Ten shows the python code that was used in the interactive python session, and the ls(a) command shown after to show the spoofed packet. Below in Image Eleven it shows task-1.1A.py being run at the same time picking up the spoofed packet.

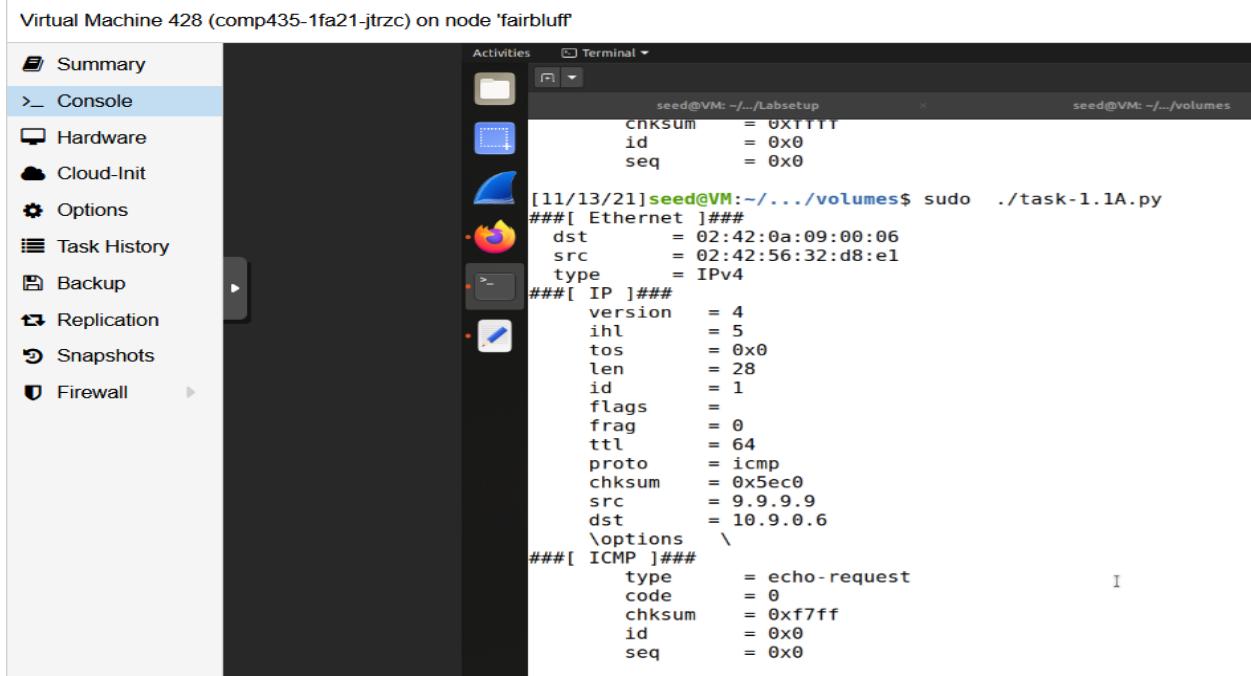
Image Ten



Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

```
[11/13/21] seed@VM: ~/.../Labsetup$ sudo python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> a = IP()
>>> a.src = '9.9.9.9'
>>> a.dst = '10.9.0.6'
>>> b = ICMP()
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
>>> ls(a)
version      : BitField (4 bits)          = 4           (4)
ihl         : BitField (4 bits)          = None        (None)
tos         : XByteField               = 0            (0)
len         : ShortField              = None        (None)
id          : ShortField              = 1           (1)
flags        : FlagsField (3 bits)        = <Flag 0 ()> (<Flag 0 ()>)
frag        : BitField (13 bits)         = 0            (0)
ttl          : ByteField                = 64          (64)
proto        : ByteEnumField           = 0           (0)
chksum       : XShortField             = None        (None)
src          : SourceIPField           = '9.9.9.9'   (None)
dst          : DestIPField              = '10.9.0.6'  (None)
options      : PacketListField         = []          ([])
```

Image Eleven



Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

```
Activities Terminal
seed@VM: ~/.../Labsetup
seed@VM: ~/.../volumes

[11/13/21] seed@VM: ~/.../volumes$ sudo ./task-1.1A.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:06
src      = 02:42:56:32:d8:e1
type    = IPv4
###[ IP ]###
version  = 4
ihl     = 5
tos     = 0x0
len     = 28
id      = 1
flags   =
frag    = 0
ttl     = 64
proto   = icmp
chksum  = 0x5ec0
src     = 9.9.9.9
dst     = 10.9.0.6
\options \
###[ ICMP ]###
type    = echo-request
code   = 0
checksum = 0xf7ff
id     = 0x0
seq    = 0x0
```

Task 1.3: Traceroute

Task 1.3 involved figuring out how many routers or hops it would take to get to the wanted destination, in this case google.com. After trial and error, the upper bound, seen in Image Twelve, was set to 17, and the amount of hops it took was 16 to get to the final destination seen in Image Thirteen.

Image Twelve

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

The screenshot shows a desktop environment with a sidebar containing icons for Summary, Console, Hardware, Cloud-Init, Options, Task History, Backup, Replication, Snapshots, and Firewall. The 'Console' icon is selected. In the main window, there is a terminal-like interface with a file browser on the left. A text editor window titled 'task-1.1A.py' is open, displaying the following Python code:

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4 hostname = "google.com"
5
6#TODO
7# Modify the upper_bound variable so that
8# you can see how many routers there are
9# from you to your destination.
10
11upper_bound = 17
12
13for i in range(1, upper_bound):
14    pkt = IP(dst=hostname, ttl=i) / UDP(dport=33434)
15    reply = sr1(pkt, verbose=0)
16    if reply == None:
17        break
18    elif reply.type == 3:
19        print("Done", reply.src)
20    else:
21        print("%d hops away: %i, reply.src)
```

Image Thirteen

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

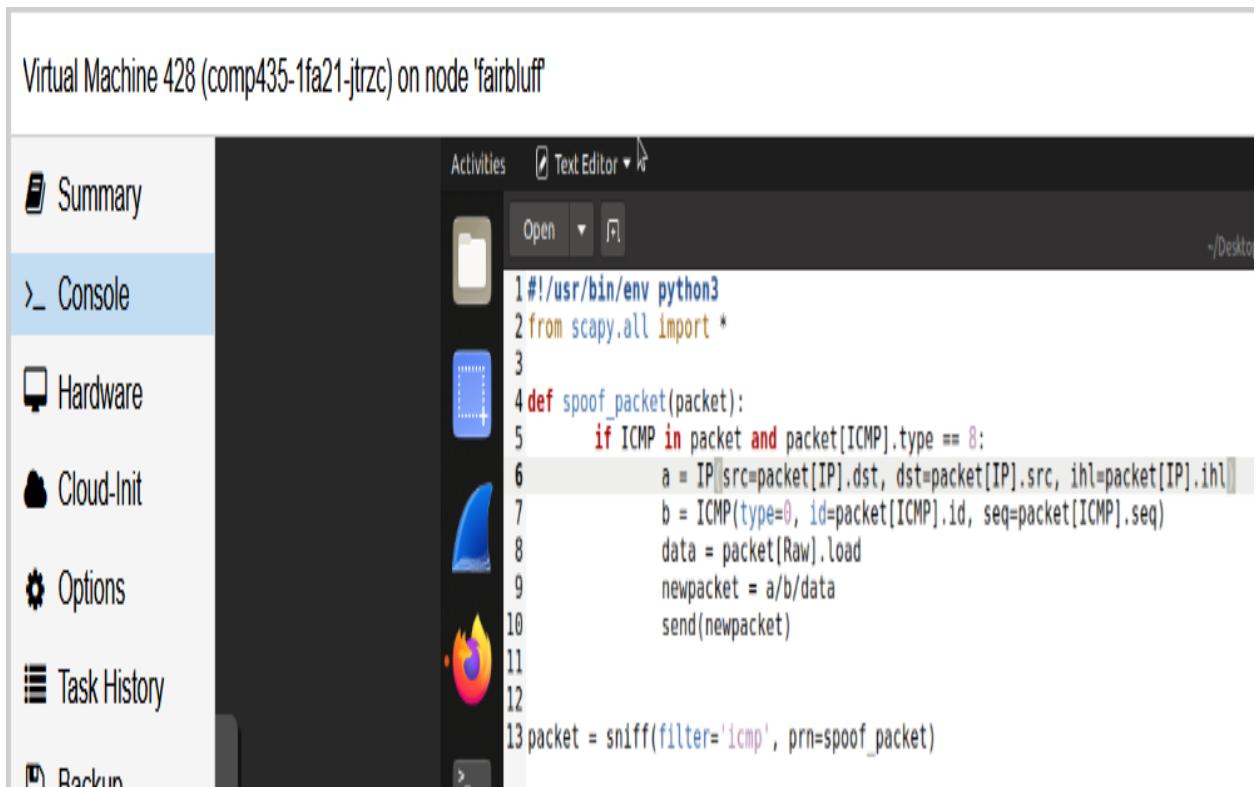
The screenshot shows a desktop environment with a sidebar containing icons for Summary, Console, Hardware, Cloud-Init, Options, Task History, Backup, Replication, Snapshots, and Firewall. The 'Console' icon is selected. In the main window, there is a terminal-like interface with a file browser on the left. The terminal window shows the command 'sudo ./task-1.3.py' being run, followed by the output of a traceroute command:

```
[11/13/21]seed@VM:~/.../Labsetup$ sudo ./task-1.3.py
1 hops away: 152.19.160.1
2 hops away: 152.19.253.53
3 hops away: 152.19.253.89
4 hops away: 152.2.255.65
5 hops away: 128.109.1.89
6 hops away: 128.109.9.21
7 hops away: 198.86.53.233
8 hops away: 108.170.249.163
9 hops away: 209.85.248.247
10 hops away: 216.239.40.131
11 hops away: 142.250.209.128
12 hops away: 142.251.49.36
13 hops away: 142.251.49.16
14 hops away: 108.170.240.97
15 hops away: 142.250.232.77
Done 142.250.73.238
```

Task 1.4: Sniffing then Spoofing

The first part of the code is actually using scapy to sniff the packet and save it, similar to examples above. The packet was named packet, and can be seen in line 13 where it is declared. The packet is then defined in a function called spoof_packet where the actually spoofing will be done. The next line is an if statement, and the purpose of this is to see if this is an ICMP echo request, which is why it is == 8 because the value of the ICMP echo request is 8. If the packet that was sniffed is an ICMP echo request, go into the if statement where if will then set up a spoofed packet to be sent out automatically. Line 6 is the start of the creation of the spoofed packet. The three headers to be set for the IP field are the src, dst, and ihl which are source, destination, and the size. Because it is an echo reply, the source and destination are flipped from the original packet, but the ihl which is the size should stay the same. The next line is the ICMP field, and the type is set to 0 because that is the value for an echo reply which is what is being spoofed. The id and sequence number are going to be the same because this is an echo reply, so these can be directly taken from the original request packet. After this, the data needs to be copied from the request packet, and it all needs to be included in the reply packet so it is going to be added to the end of the packet which can be seen in line 8. It is all added together like in previous examples and the newpacket which is the spoofed packet is sent out. The results from pinging the three IPs can be seen in Image Fifteen.

Image Fourteen



Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

Activities Text Editor ↗

Open ↗ ~/Desktop

Summary

Console

Hardware

Cloud-Init

Options

Task History

Recent

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def spoof_packet(packet):
5    if ICMP in packet and packet[ICMP].type == 8:
6        a = IP(src=packet[IP].dst, dst=packet[IP].src, ihl=packet[IP].ihl]
7        b = ICMP(type=0, id=packet[ICMP].id, seq=packet[ICMP].seq)
8        data = packet[Raw].load
9        newpacket = a/b/data
10       send(newpacket)
11
12
13 packet = sniff(filter='icmp', prn=spoof_packet)
```

Image Fifteen

Virtual Machine 428 (comp435-1fa21-jtrzc) on node 'fairbluff'

The screenshot shows the Oracle VM VirtualBox Manager interface. On the left, a sidebar lists various management options: Summary, Console (which is selected), Hardware, Cloud-Init, Options, Task History, Backup, Replication, Snapshots, and Firewall. The main area is a terminal window titled 'seed@VM: ~.../Labsetup'. The terminal output shows several ping commands being run from the VM's perspective:

```
3 packets transmitted, 0 received, 100% packet loss, time 2030ms
[11/13/21]seed@VM:~/.../Labsetup$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=54.8 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=27.6 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=22.1 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=39.7 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=18.9 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=26.4 ms
^C
--- 1.2.3.4 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5006ms
rtt min/avg/max/mdev = 18.912/31.593/54.805/12.243 ms
[11/13/21]seed@VM:~/.../Labsetup$ ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.1 icmp_seq=1 Destination Host Unreachable
From 10.9.0.1 icmp_seq=2 Destination Host Unreachable
From 10.9.0.1 icmp_seq=3 Destination Host Unreachable
From 10.9.0.1 icmp_seq=4 Destination Host Unreachable
From 10.9.0.1 icmp_seq=8 Destination Host Unreachable
From 10.9.0.1 icmp_seq=9 Destination Host Unreachable
From 10.9.0.1 icmp_seq=11 Destination Host Unreachable
From 10.9.0.1 icmp_seq=12 Destination Host Unreachable
^CFrom 10.9.0.1 icmp_seq=13 Destination Host Unreachable
From 10.9.0.1 icmp_seq=14 Destination Host Unreachable
From 10.9.0.1 icmp_seq=15 Destination Host Unreachable
From 10.9.0.1 icmp_seq=16 Destination Host Unreachable
^C
--- 10.9.0.99 ping statistics ---
19 packets transmitted, 0 received, +12 errors, 100% packet loss, time 18413ms
pipe 4
[11/13/21]seed@VM:~/.../Labsetup$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=11.4 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=37.8 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=11.2 ms
```