Traffic Generator/DDoS Tool
(Network and application level)

Manas Kumar Panda(ADG SAS-IV)
Hari Ramamoorthy(RA)

Recent Incident about DDoS attack:

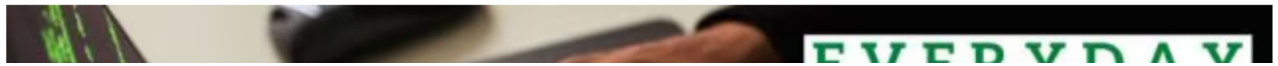
Trump-Musk interview on X reportedly hit by DDoS attack: What it means

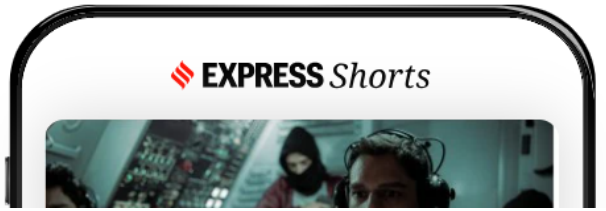
Elon Musk cited a "massive DDoS attack" for the technical glitches during his recent audio interview with Donald Trump on X. We explain how such attacks work.

By: Explained Desk
New Delhi | Updated: August 14, 2024 11:31 IST

4 min read

NewsGuard

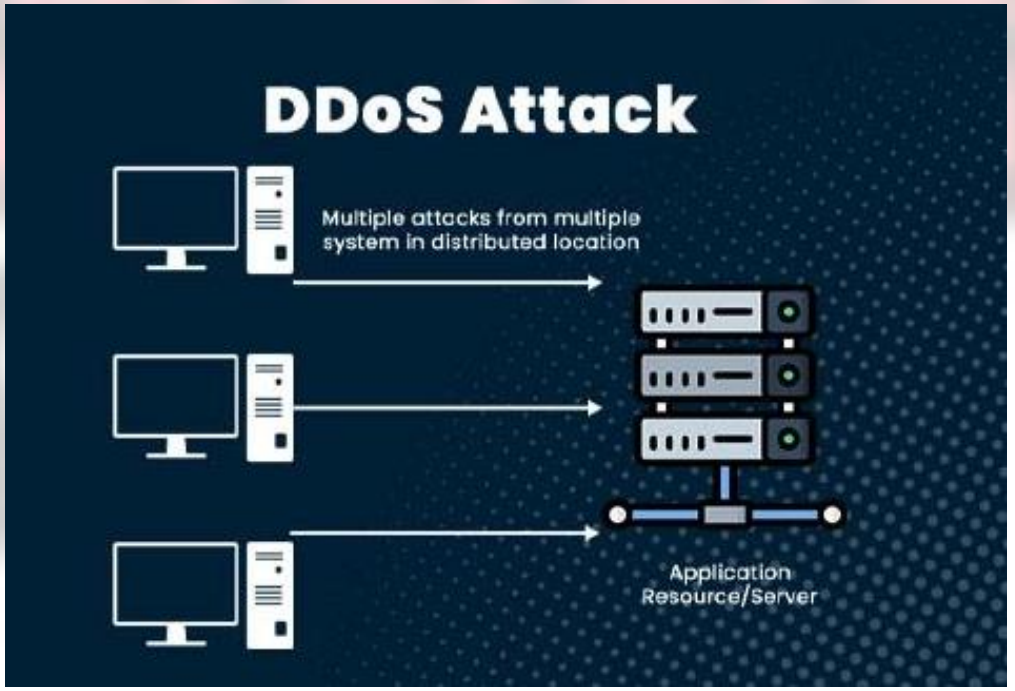




Difference between DoS vs DDoS :

DoS : Denial of Service
Attack the DUT(Services/Protocols) from single device

DDoS : Distributed Denial of Service
Attack the DUT(Services/Protocols) from multiple device



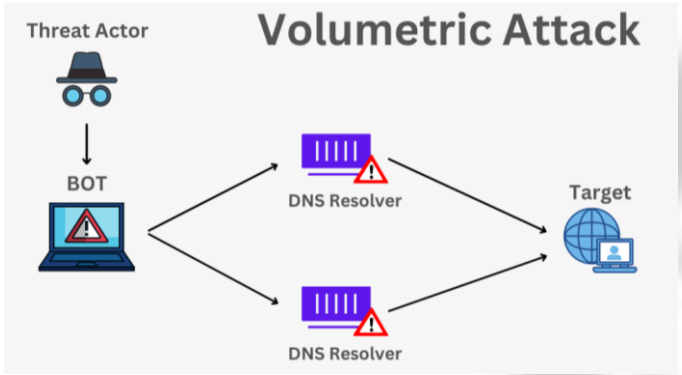
How do you know if an attack is happening?

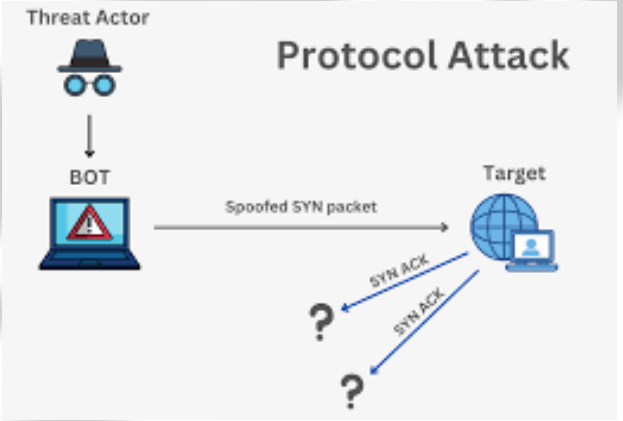
Symptoms
Unusually slow (opening files or accessing websites),
•Unavailability of a particular website, or
•An inability to access any website.

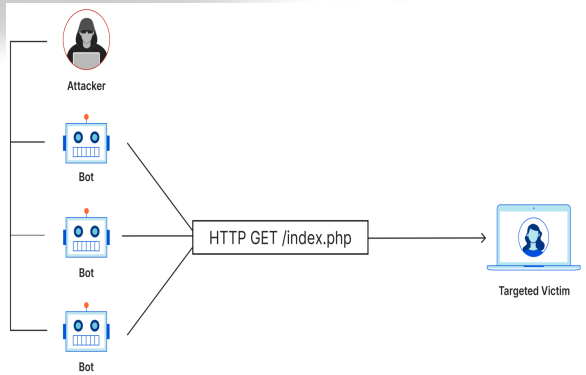
Confirm TEST:via network traffic monitoring and analysis.

Types of DoS/DDoS Attack :

Type	Volume-based	Protocol-based	Application layer
Magnitude measurement	Bits per second (bps)	Packets per second (pps)	Requests per second (rps)
Common examples	• UDP flood • DNS amplification • Misused application	• SYN flood • ICMP flood • Ping of death	• HTTP flood • Slowloris







Context-What ITSAR/3GPP Says?

- Section 4.2.3.3.1 → **System handling during overload situations**

ITSAR*-Network Level and application-level DDoS*

The system shall provide security measures to deal with overload situations which may occur as a result of a denial of service attack or during periods of increased traffic, or reach the congestion threshold

NE shall have protection mechanism against Network level and Application-level DDoS attacks. NE shall provide security measures to deal with overload situations which may occur as a result of a denial of service attack- *(Attack generator)* or during periods of increased traffic.----(*Traffic Generator)*

Testing

***Test case*: Refer to test case in clause 4.2.3.3.3.**

Context-What ITSAR/3GPP Says?

- Section 4.2.3.3.1 ->System handling during excessive overload situations
- **Excessive Overload Protection**
- The system shall act in a predictable way if an overload situation cannot be prevented. A system shall be built in this way that it can react on an overload situation in a controlled way. However it is possible that a situation happens where the security measures are no longer sufficient.
- In such case it shall be ensured that the system cannot reach an undefined and thus potentially insecure state. In an extreme case this means that a controlled system shutdown is preferable to uncontrolled failure of the security functions and thus loss of system protection.
- Simulate a Overload Situation-Push to the limit--- Can be by DDoS or Increased Traffic
- **Test Name:** TC_SYSTEM_HANDLING_OF_OVERLOAD_SITUATIONS
  - NOTE: This test case covers requirements 4.2.3.3.1 and this requirement4.2.3.3.3.

Volume/Protocol based DoS/DDoS Attack :

1. **Random Source Attack :** In this attack, an attacker can send multiple random packets with different source addresses to the target machine which may cause the Distributed denial of service attack. It is difficult to identify the actual source address after an incident occurs.
   Ex : *# hping3 -S -p 80 127.0.0.1 — flood — rand-source*
2. **SMURF Attack :** This is a kind of DDoS attack in which spoofed source address send a large amount of ICMP packets to the target address. It uses a victim address as a source address to send/broadcast the multiple ICMP ping request.
   Ex : *# hping3 — icmp — flood 127.0.0.1 -a 127.0.0.1*
3. **LAND Attack :** This is a kind of DoS (Denial of Service) attack in which a packet is sent to a target machine with the same address (Source Address and destination address the same).
   Ex : *hping3 -S -p 80 127.0.0.1 -a 127.0.0.1*
4. **SYN Flood Attack :** Syn flood is also known as a half-open attack. In this attack, the attacker sends multiple connection requests to perform the distributed denial of service attack.
   Ex : *# hping3 -S -p 80 Target — flood*
5. **TCP Sequence Prediction Attack (ISN Prediction)** : When a packet is sent or received from client to server, usually each packet contains a sequence number which helps to keep tracking of received and acknowledged data packets. Sometimes attackers exploit the sequence number of TCP packets and to commit attacked to perform malicious activities. The aim of this attack is to predict the sequence number used to identify the packets in a TCP connection, which can be used to counterfeit packets. Below is the command to identify the sequence number of TCP Packets
   Ex : *# hping3 -S -p 80 -Q 127.0.0.1*

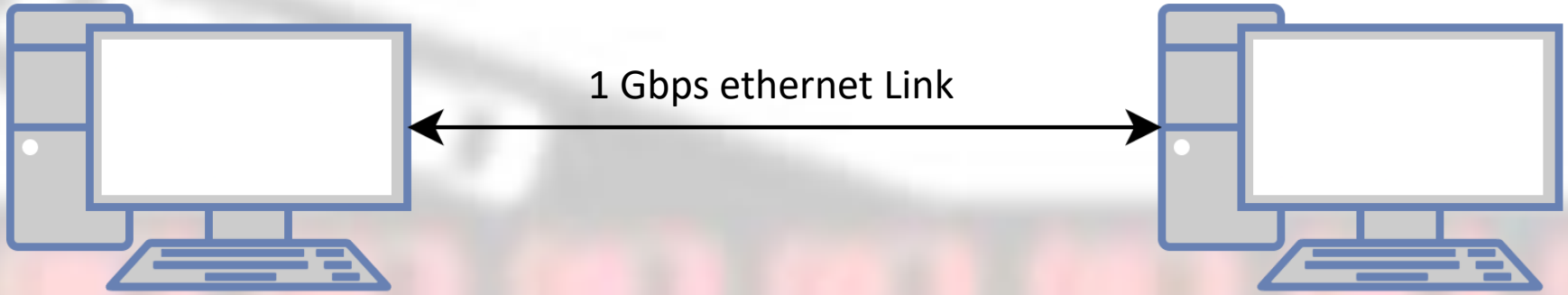
Tools Used : hping3, any commercial tool

What is hping3 ?

- hping3 is a network tool able to send custom ICMP/UDP/TCP packets and to display target replies like ping does with ICMP replies.
- It handles fragmentation and arbitrary packet body and size, and can be used to transfer files under supported protocols.
- Using hping3, you can test
  - ❖ firewall rules,
  - ❖ perform (spoofed) port scanning,
  - ❖ test network performance using different protocols,
  - ❖ do path MTU discovery,
  - ❖ perform traceroute-like actions under different protocols,
  - ❖ fingerprint remote operating systems,
  - ❖ audit TCP/IP stacks.

Test Case I : Desktop environment

Test Setup :



Tester Machine(Client Machine - Hping3)
OS : Ubuntu 22.04
Src.IP : 192.168.129.21/24

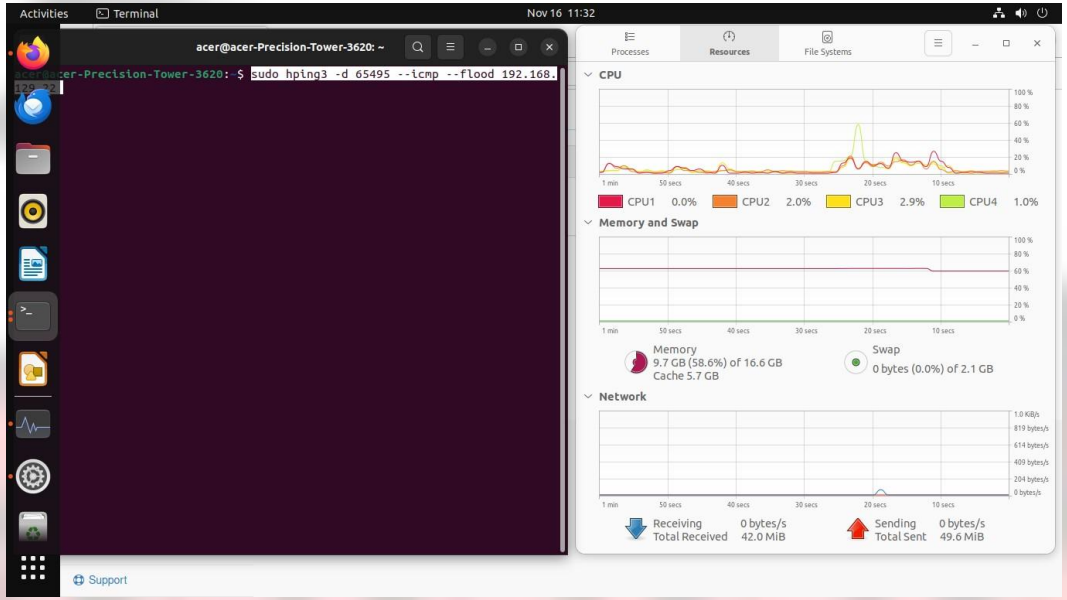
DUT Machine(Server Machine)
OS : Windows 10 Pro
Src.IP : 192.168.129.22/24

Test Plan : Perform Flood based attacks

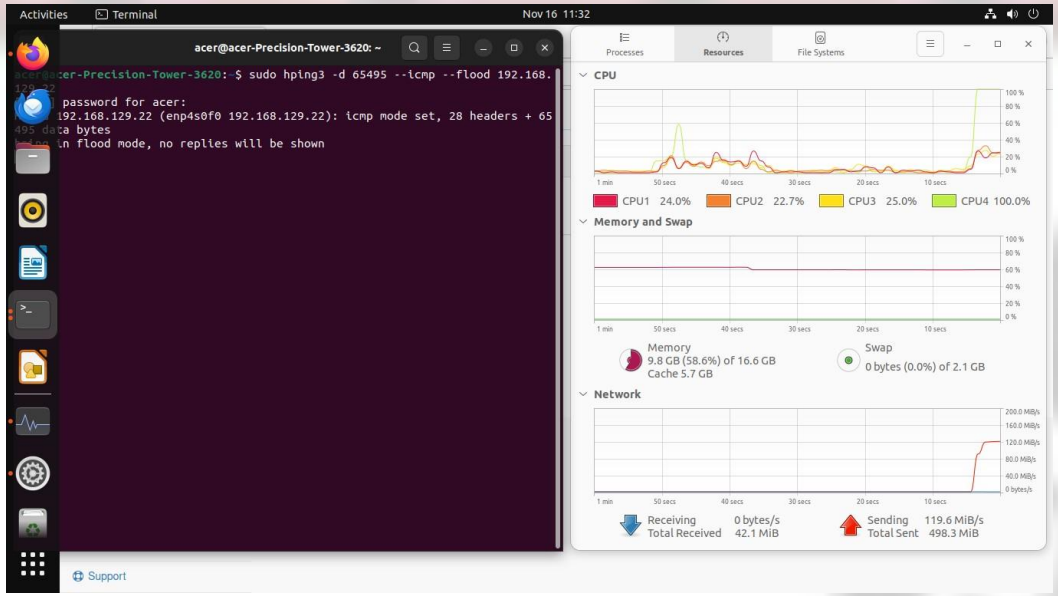
1) ICMP
2) TCP(SYN, ACK, RST, FIN, other flgs also)
3) UDP
4) RAW IP

1. ICMP Flood Traffic :

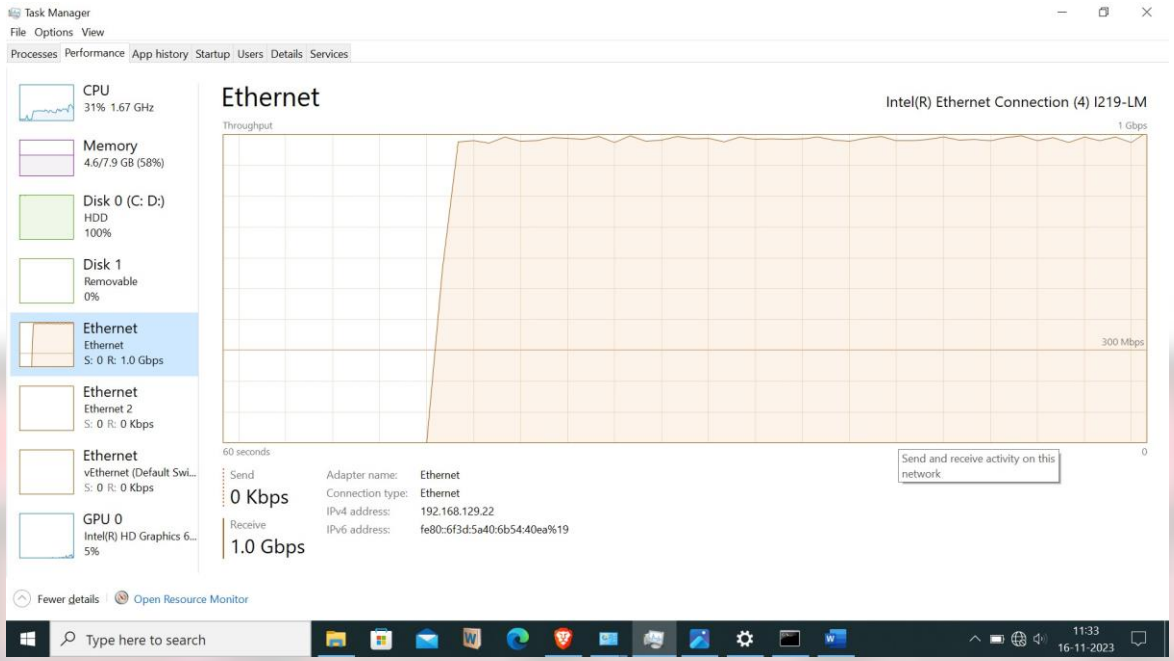
1.1 Before ICMP flood traffic at client side



1.2 After ICMP flood traffic at client side



1.3 After ICMP flood traffic at Server side



Commands Info :

sudo hping3 –d 65495 --icmp –flood 192.168.129.22

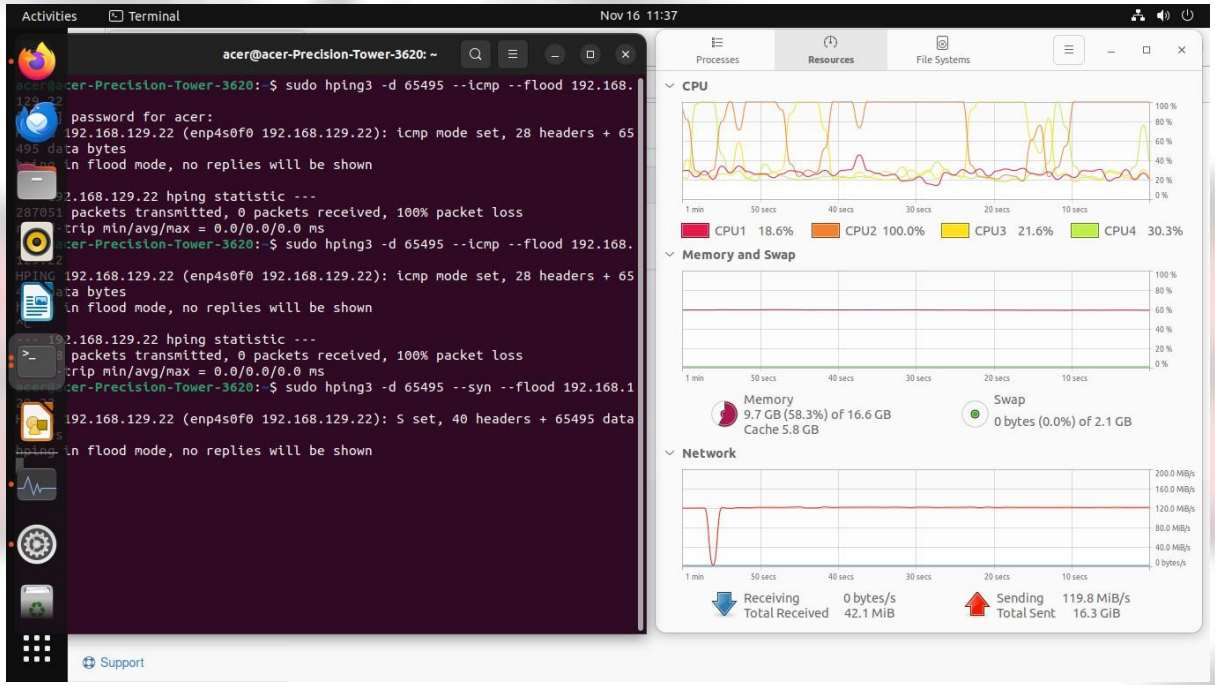
-d -> data size(payload)

--icmp -> use ICMP protocol

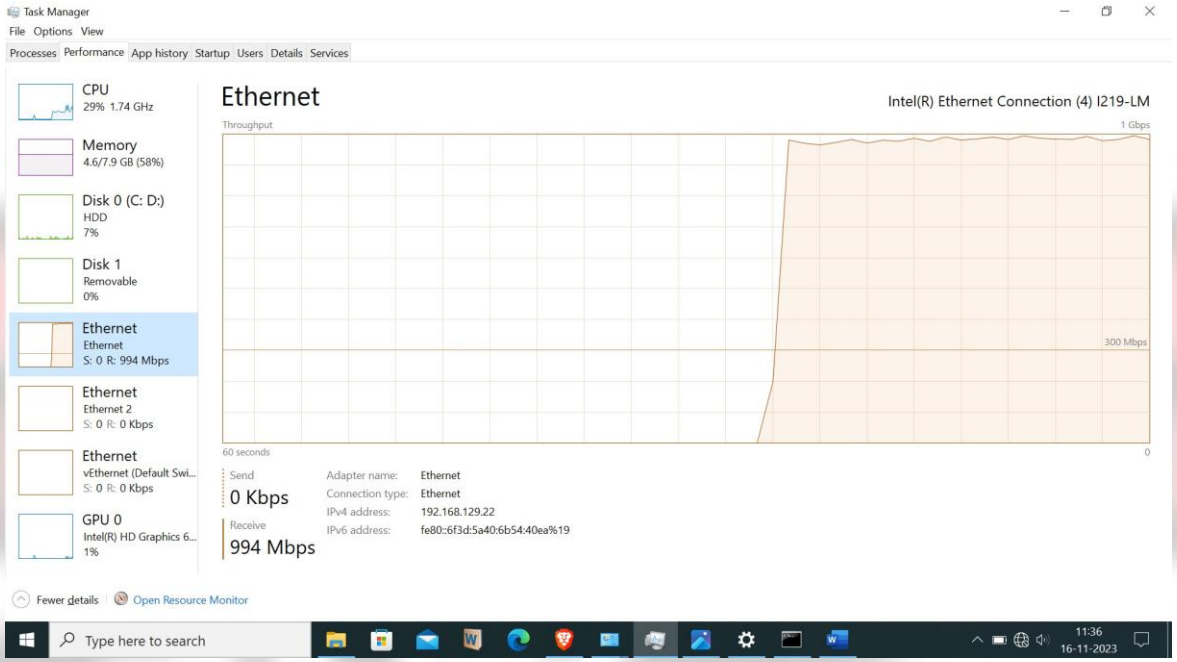
--flood -> Don’t wait for reply. Keep send packets as much possible

2. TCP SYN Flood Traffic :

2.1 SYN flood traffic at client side



2.2 SYN flood traffic at Server side



Commands Info :

sudo hping3 –d 65495 --syn --flood 192.168.129.22

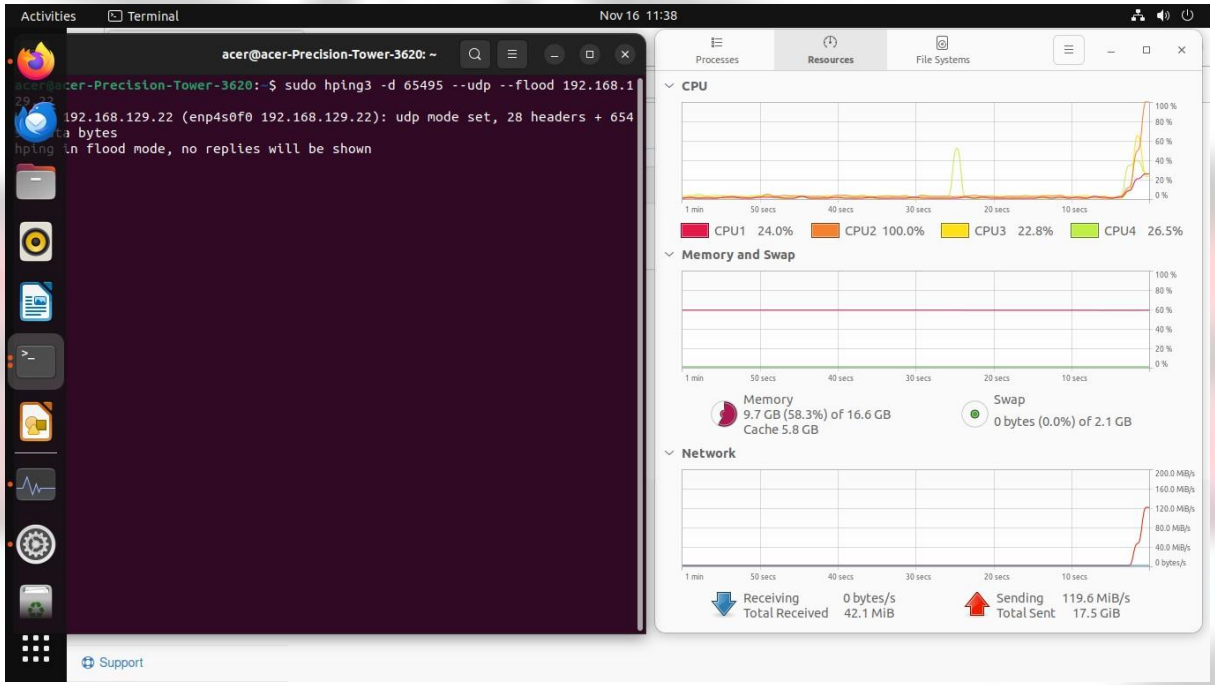
-d -> data size(payload)

--syn -> use TCP SYN protocol

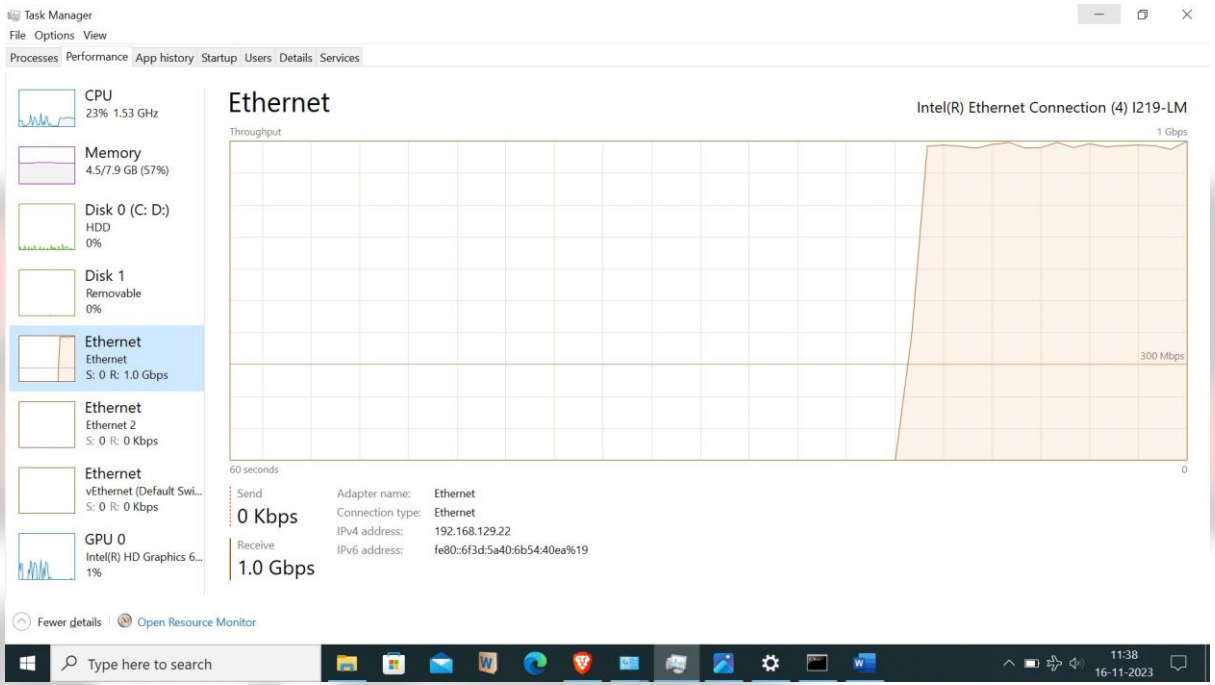
--flood -> Don’t wait for reply. Keep send packets as much possible

3. UDP Flood Traffic :

3.1 UDP flood traffic at client side



3.2 UDP flood traffic at Server side



Commands Info :

sudo hping3 –d 65495 --udp --flood 192.168.129.22

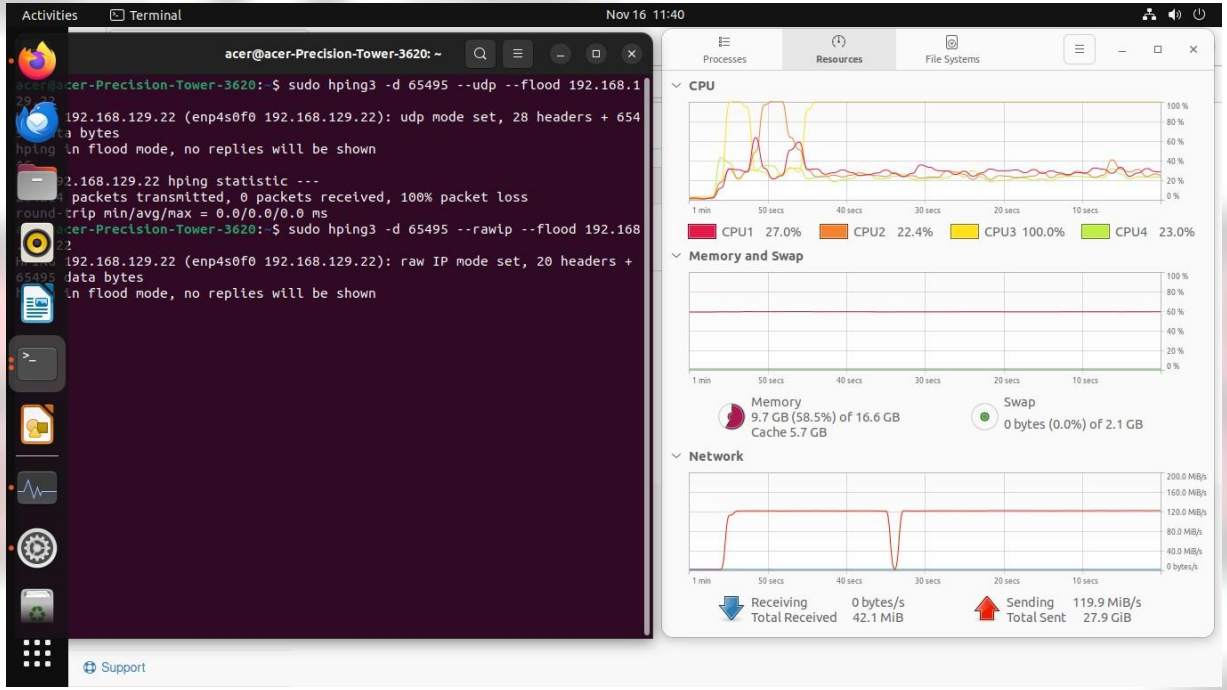
-d -> data size(payload)

--syn -> use UDP protocol

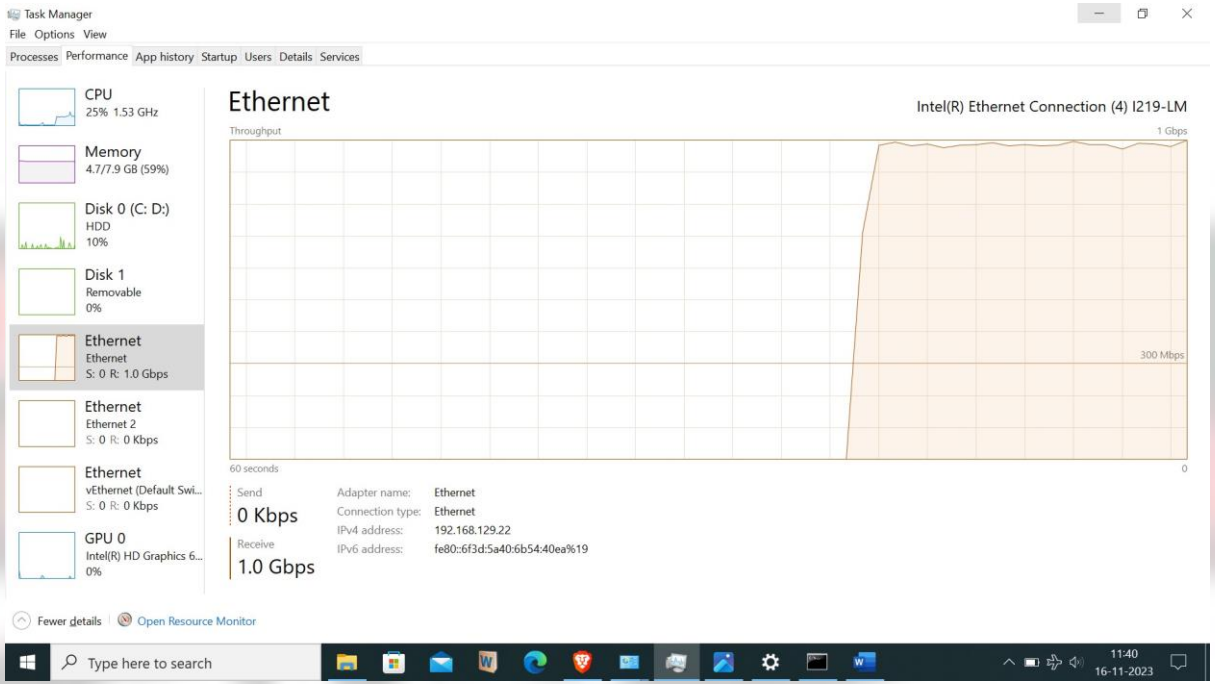
--flood -> Don’t wait for reply. Keep send packets as much possible

4. RAW IP Flood Traffic :

4.1 Raw IP flood traffic at client side



4.2 Raw IP flood traffic at Server side



Commands Info :

sudo hping3 –d 65495 --rawip --flood 192.168.129.22

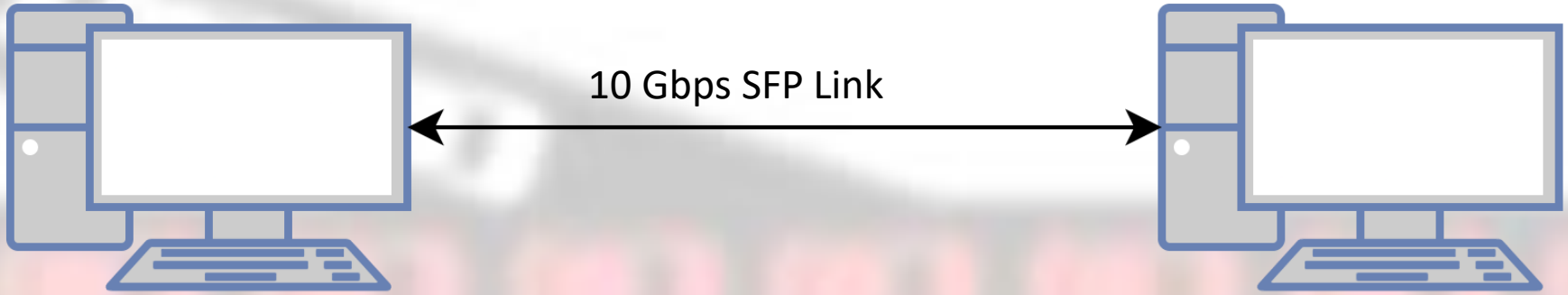
-d -> data size(payload)

--syn -> use IP(RAW) protocol

--flood -> Don’t wait for reply. Keep send packets as much possible

Test Case II : Server environment

Test Setup :



Hybervisor 1(Client Machine - Hping3)
OS : Ubuntu 20.04
IP : 192.168.20.5/24

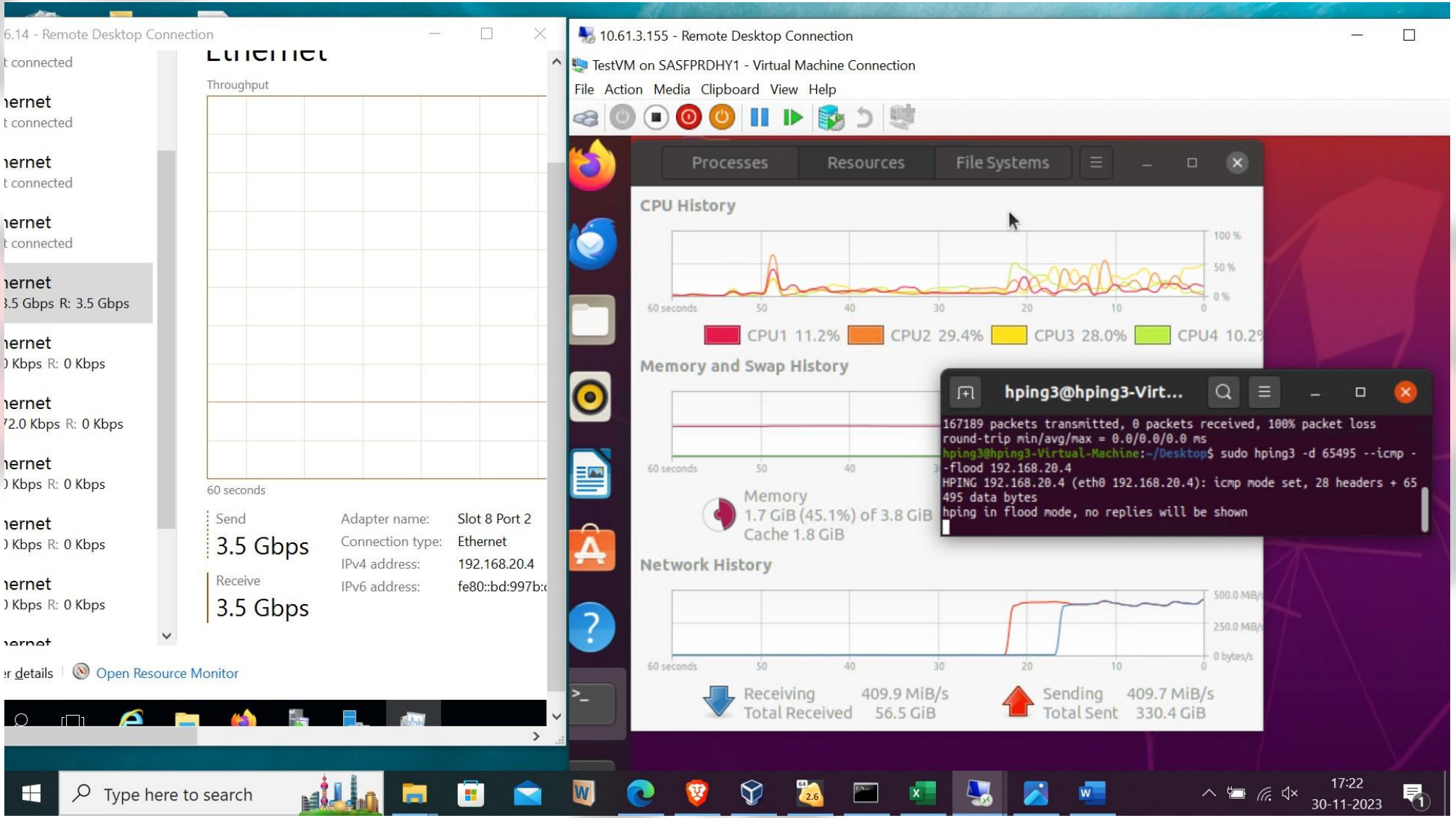
Hybervisor 2 (Server Machine)
OS : Windows 10 Pro
IP : 192.168.20.4/24

Test Plan : Perform Flood based attacks

1) ICMP
2) TCP(SYN, ACK, RST, FIN, other flgs also)
3) UDP
4) RAW IP

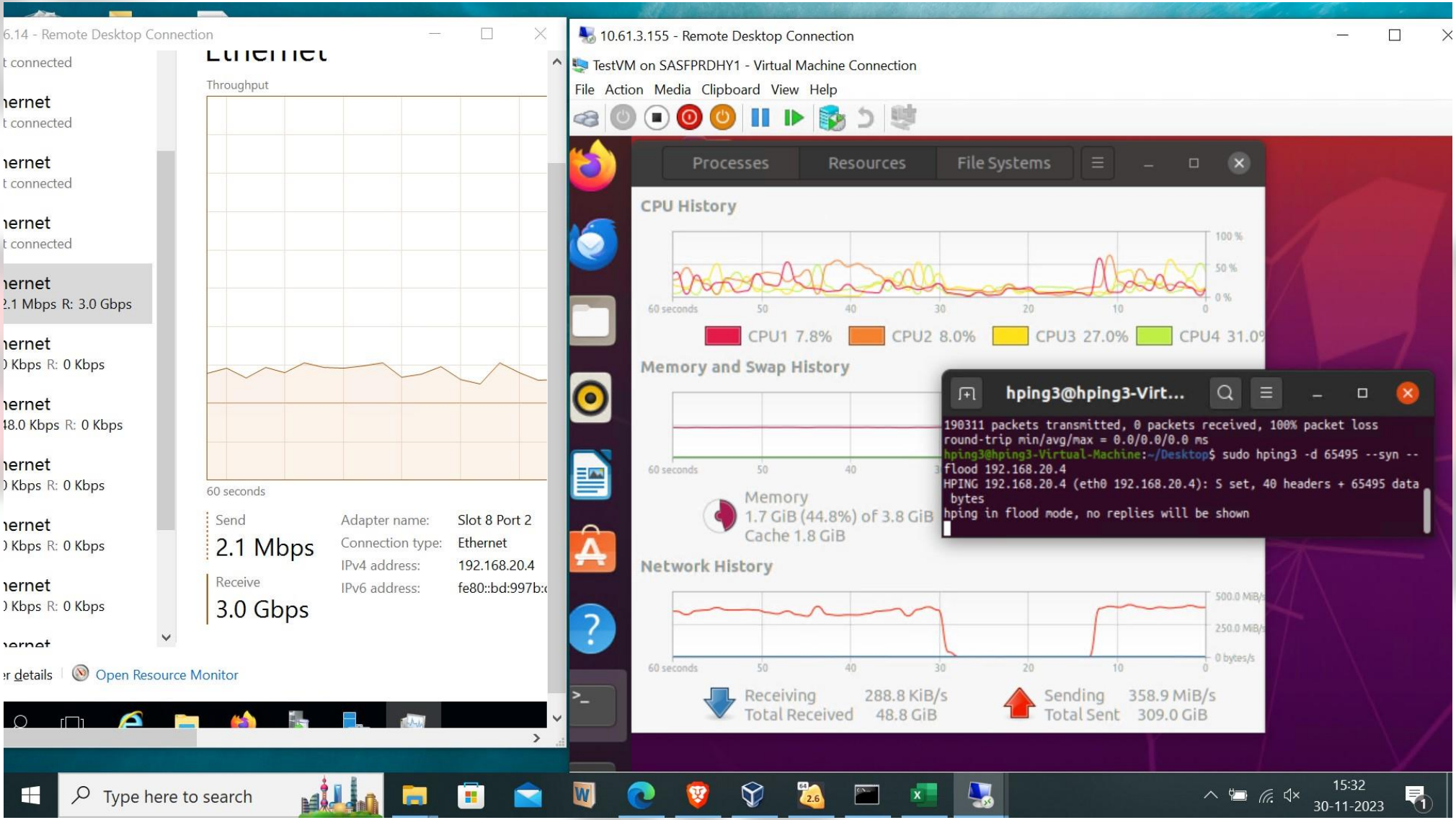
1. ICMP Flood Traffic :

1.1 ICMP flood traffic at both server and client side



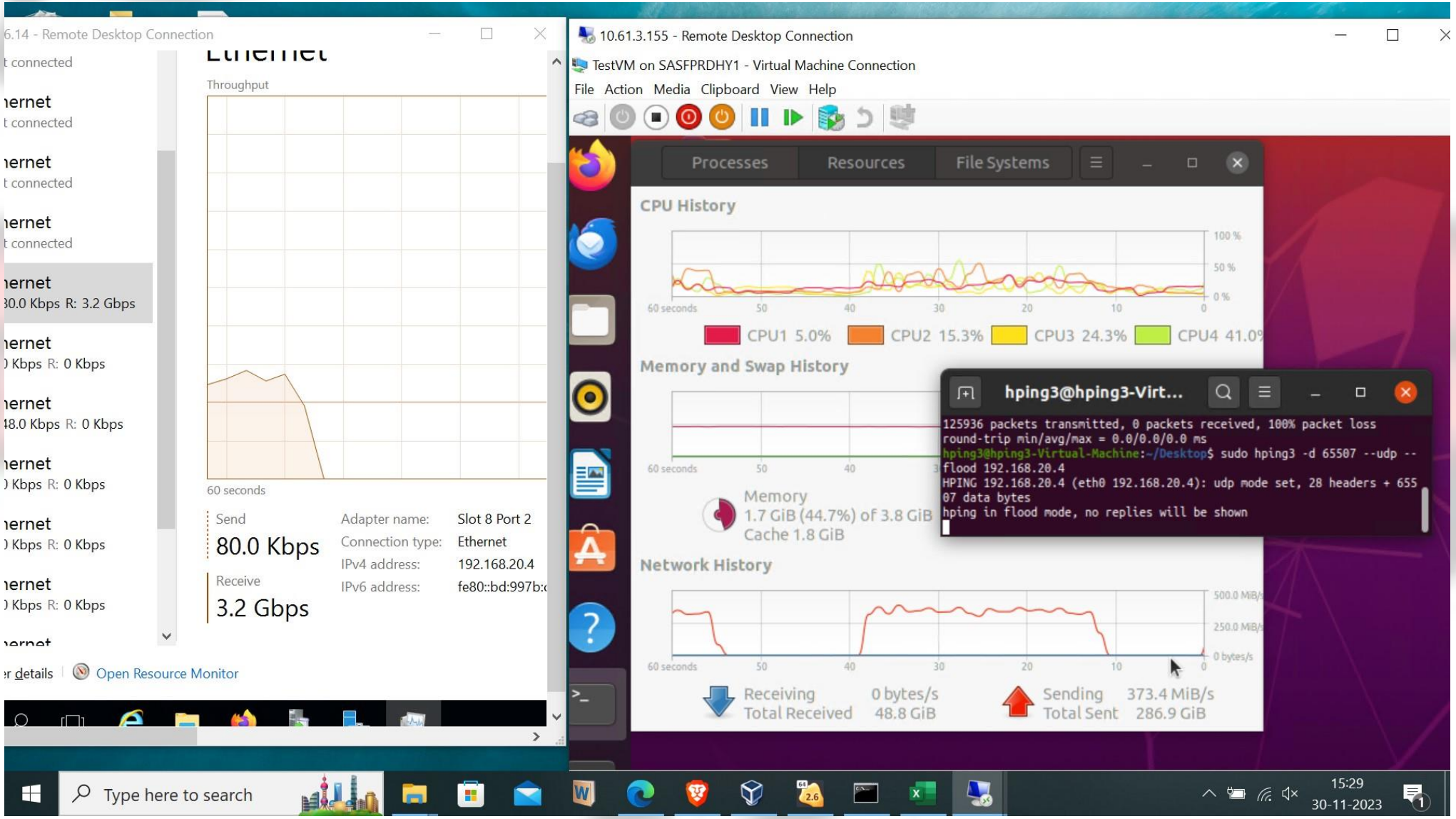
2. TCP SYN Flood Traffic :

2.1 TCP SYN flood traffic at both server and client side



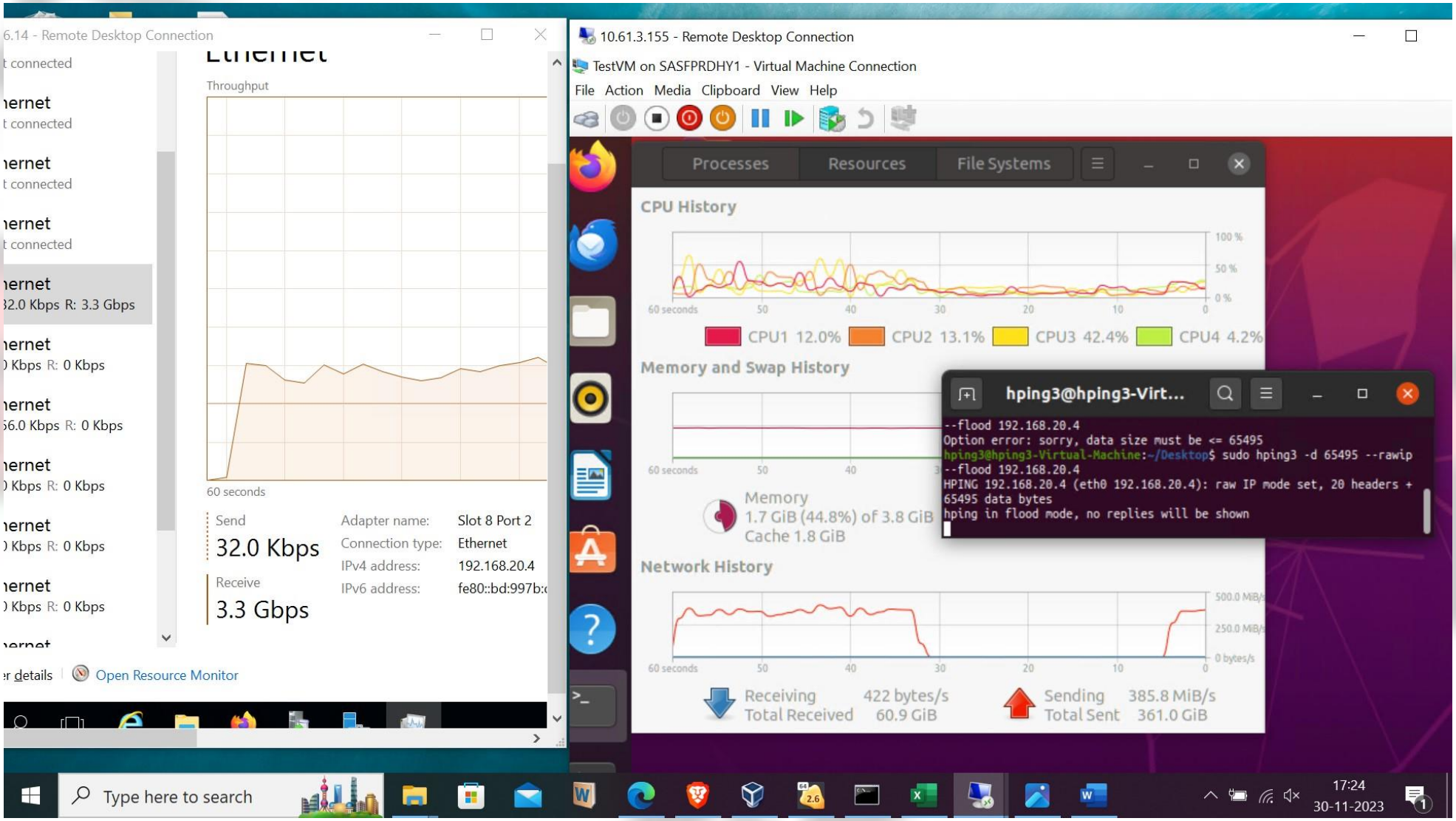
3. UDP Flood Traffic :

3.1 UDP flood traffic at both server and client side



4. RAW IP Flood Traffic :

4.1 RAW IP flood traffic at both server and client side



Hping3 Commands Info :

```
cer@acer-Precision-Tower-3620:~$ hping3 --help
sage: hping3 host [options]
 -h  --help      show this help
 -v  --version   show version
 -c  --count     packet count
 -i  --interval  wait (uX for X microseconds, for example -i u1000)
     --fast      alias for -i u10000 (10 packets for second)
     --faster    alias for -i u1000 (100 packets for second)
     --flood      sent packets as fast as possible. Don't show replies.
 -n  --numeric   numeric output
 -q  --quiet     quiet
 -I  --interface interface name (otherwise default routing interface)
 -V  --verbose   verbose mode
 -D  --debug     debugging info
 -z  --bind      bind ctrl+z to ttl           (default to dst port)
 -Z  --unbind    unbind ctrl+z
     --beep      beep for every matching packet received
ode
 default mode     TCP
 -0  --rawip      RAW IP mode
 -1  --icmp       ICMP mode
 -2  --udp        UDP mode
 -8  --scan       SCAN mode.
                  Example: hping --scan 1-30,70-90 -S www.target.host
 -9  --listen     listen mode
P
 -a  --spoof      spoof source address
 --rand-dest      random destionation address mode. see the man.
 --rand-source    random source address mode. see the man.
 -t  --ttl        ttl (default 64)
 -N  --id         id (default random)
 -W  --winid      use win* id byte ordering
 -r  --rel        relativize id field          (to estimate host traffic)
 -f  --frag       split packets in more frag.  (may pass weak acl)
 -x  --morefrag   set more fragments flag
 -y  --dontfrag   set don't fragment flag
 -g  --fragoff    set the fragment offset
-m  --mtu        set virtual mtu, implies --frag if packet size > mtu
-o  --tos        type of service (default 0x00), try --tos help
-G  --rroute     includes RECORD_ROUTE option and display the route buffer
--lsrr           loose source routing and record route
--ssrr           strict source routing and record route
-H  --ipproto    set the IP protocol field, only in RAW IP mode
MP
 -C  --icmptype   icmp type (default echo request)
 -K  --icmpcode   icmp code (default 0)
     --force-icmp send all icmp types (default send only supported types)
     --icmp-gw    set gateway address for ICMP redirect (default 0.0.0.0)
     --icmp-ts    Alias for --icmp --icmptype 13 (ICMP timestamp)
     --icmp-addr  Alias for --icmp --icmptype 17 (ICMP address subnet mask)
     --icmp-help  display help for others icmp options
P/TCP
 -s  --baseport   base source port             (default random)
 -p  --destport   [+][+]<port> destination port(default 0) ctrl+z inc/dec
 -k  --keep       keep still source port
 -w  --win        winsize (default 64)
 -O  --tcpoff     set fake tcp data offset     (instead of tcphdrlen / 4)
 -Q  --seqnum     shows only tcp sequence number
 -b  --badcksum   (try to) send packets with a bad IP checksum
                  many systems will fix the IP checksum sending the packet
                  so you'll get bad UDP/TCP checksum instead.
 -M  --setseq     set TCP sequence number
 -L  --setack     set TCP ack
 -F  --fin        set FIN flag
 -S  --syn        set SYN flag
 -R  --rst        set RST flag
 -P  --push       set PUSH flag
 -A  --ack        set ACK flag
 -U  --urg        set URG flag
 -X  --xmas       set X unused flag (0x40)
 -Y  --ymas       set Y unused flag (0x80)
 --tcpexitcode    use last tcp->th_flags as exit code
 --tcp-mss        enable the TCP MSS option with the given value
```

```
  --tcp-mss        enable the TCP MSS option with the given value
  --tcp-timestamp  enable the TCP timestamp option to guess the HZ/uptime
Common
  -d  --data       data size                  (default is 0)
  -E  --file       data from file
  -e  --sign       add 'signature'
  -j  --dump       dump packets in hex
  -J  --print      dump printable characters
  -B  --safe       enable 'safe' protocol
  -u  --end        tell you when --file reached EOF and prevent rewind
  -T  --traceroute traceroute mode              (implies --bind and --ttl 1)
  --tr-stop        Exit when receive the first not ICMP in traceroute mode
  --tr-keep-ttl    Keep the source TTL fixed, useful to monitor just one hop
  --tr-no-rtt       Don't calculate/show RTT information in traceroute mode
ARS packet description (new, unstable)
  --apd-send       Send the packet described with APD (see docs/APD.txt)
```

Mitigation of Volume/Protocol based Attacks :

ICMP Flood :

•Firewall Rules: Blocking ICMP echo requests can prevent ping flood attacks.
•Rate Limiting: Limiting the number of ICMP packets from a single source.
•Intrusion Detection Systems (IDS): Detecting and blocking ping flood attacks.

TCP SYN Flood :

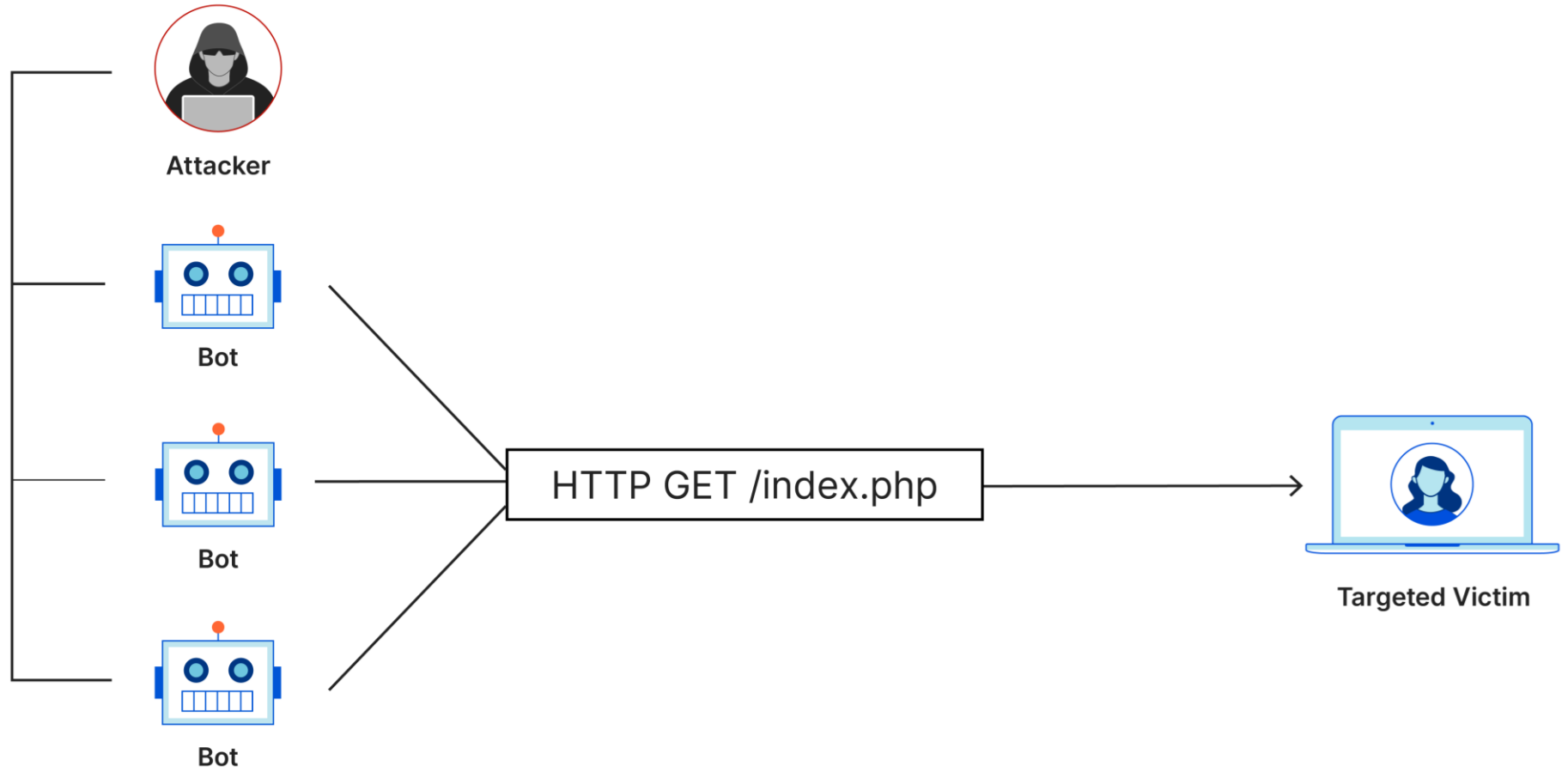
•SYN Cookies: A technique where the server sends a cookie in the SYN-ACK packet instead of allocating resources immediately.
•Rate Limiting**:** Limiting the number of SYN packets from a single source.
•Firewall and Intrusion Prevention Systems (IPS): Detecting and blocking SYN flood attacks.

UDP Flood :

ICMP Rate-Limiting: By limiting the rate at which ICMP responses are sent at the operating system level, systems can prevent being overwhelmed by the flood of return packets.
Firewall-Level Filtering: Deploying firewalls at strategic points in a network can filter out malicious traffic. With this approach, the potential victim neither receives nor responds to the malicious UDP packets.
Fingerprint Filtering : The features of attack packets may be hidden in the data segment, source IP address, source port, destination IP address, and destination port of UDP packets. The known attack features can be directly added to the filter parameters of the device. After static fingerprint filtering is configured, the device discards or rate-limits the traffic of the packets that match the attack features.

Application Layer Attacks :

- Attacker directly attacks the L-7 protocols/services/applications.
- Most commonly used application protocol is HTTP(S). So, Attacker send HTTP traffic to the DUT(Victim).
- **Tools :** Slowloris, R.U.D.Y

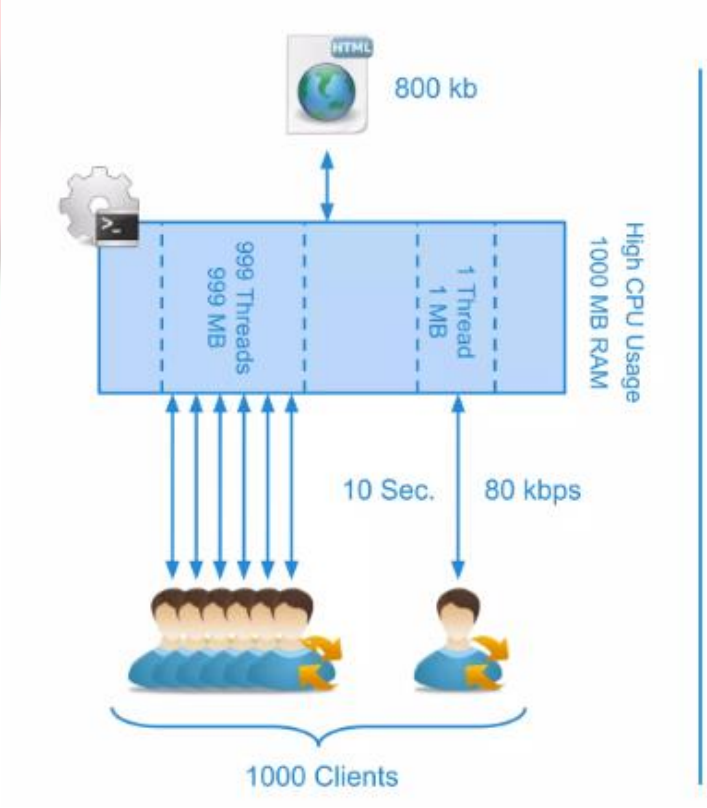


Webserver Architecture

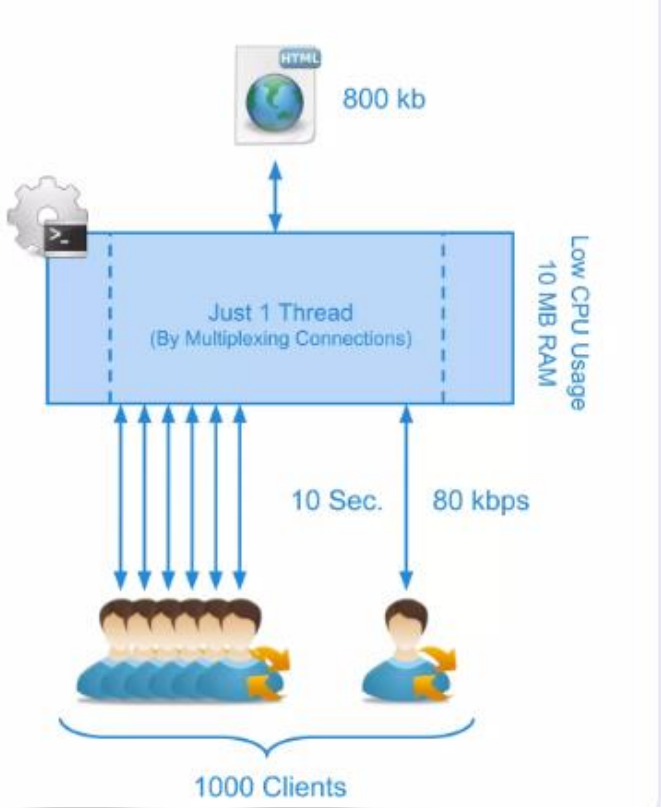
Thread based : For each HTTP request, It will create a new thread and the new thread will create a new process. So, Server resource consumption will be more.
Ex: Apache

Event based : For each HTTP request, It will use the same thread. So, Server resource handled efficiently.
Ex: Nginx

Thread based



Event based



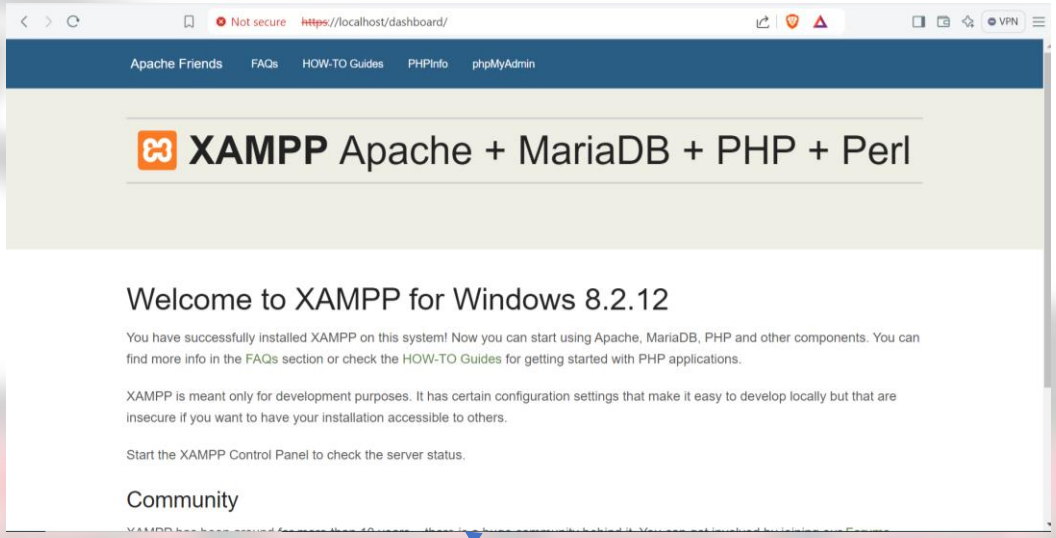
How HTTP request processed?

Slowloris(DoS/DDoS) :

What is slowloris?
Slowloris is basically an HTTP Denial of Service attack that affects threaded servers

Slowloris Attack Flow :
1.We start making lots of HTTP requests.
2.We send headers periodically (every ~15 seconds) to keep the connections open.
3.We never close the connection unless the server does so. If the server closes a connection, we create a new one keep doing the same thing.
This exhausts the servers thread pool and the server can't reply to other people.

Test Setup:




```
┌──(hari㉿ ra9)-[~/Slowloris/slowloris]
└─$ python3 slowloris.py --help
usage: slowloris.py [-h] [-p PORT] [-s SOCKETS] [-v] [-ua] [-x] [--proxy-host PROXY_HOST] [--proxy-port PROXY_PORT]
                    [--https] [--sleeptime SLEEPTIME]
                    [host]

Slowloris, low bandwidth stress test tool for websites

positional arguments:
  host                  Host to perform stress test on

options:
  -h, --help            show this help message and exit
  -p PORT, --port PORT  Port of webserver, usually 80
  -s SOCKETS, --sockets SOCKETS
                        Number of sockets to use in the test
  -v, --verbose         Increases logging
  -ua, --randuseragents
                        Randomizes user-agents with each request
  -x, --useproxy        Use a SOCKS5 proxy for connecting
  --proxy-host PROXY_HOST
                        SOCKS5 proxy host
  --proxy-port PROXY_PORT
                        SOCKS5 proxy port
  --https               Use HTTPS for the requests
  --sleeptime SLEEPTIME
                        Time to sleep between each header sent.
```

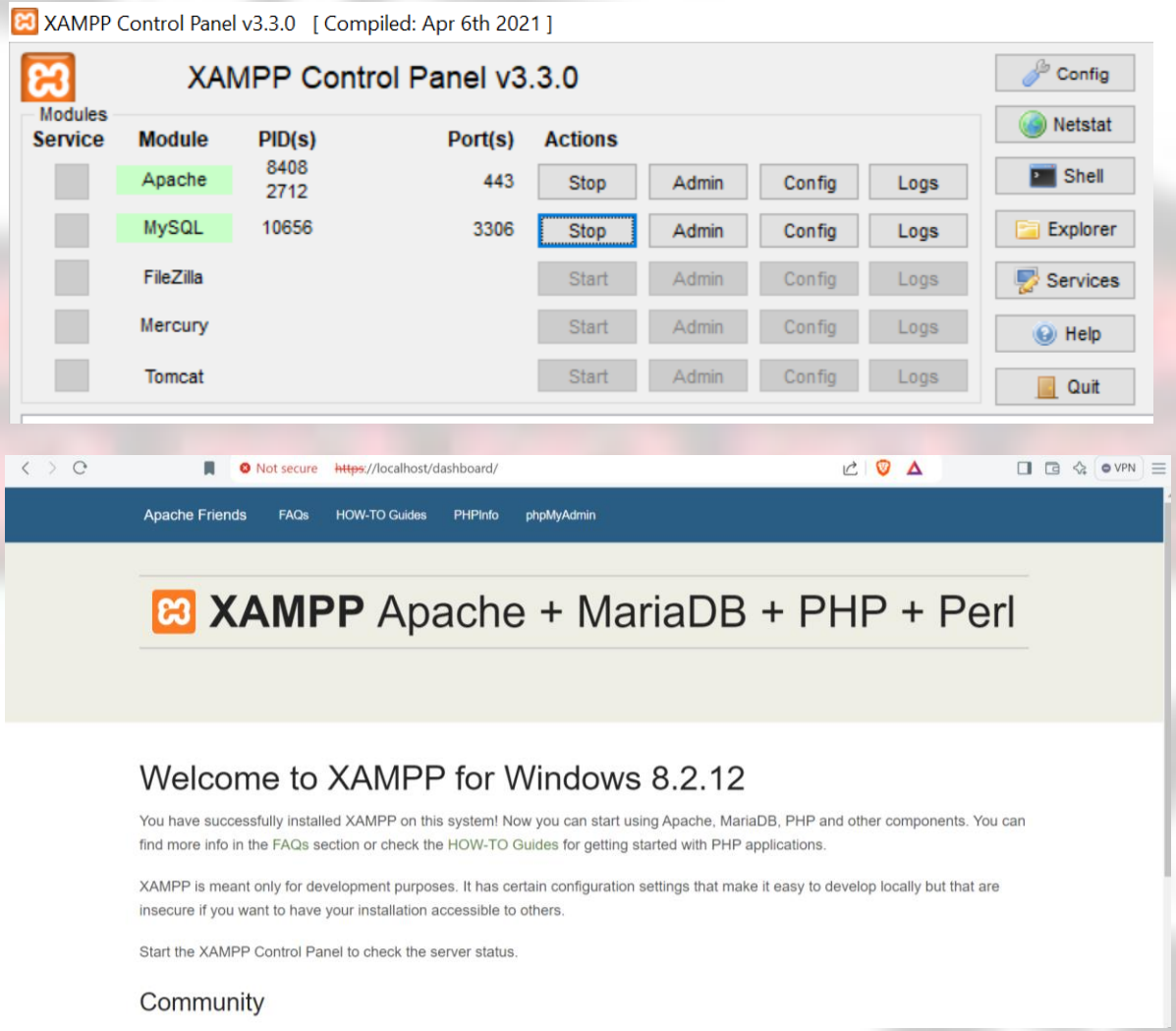
Apache(XAMPP) Server running in Host OS(Windows 10)

Slowloris tool running in WSL(Linux) of Host system

Test Case 1 : Send Request

Description : Slowloris tool sending partial HTTP request to Apache(XAMPP) server

1.1 Before running slowloris

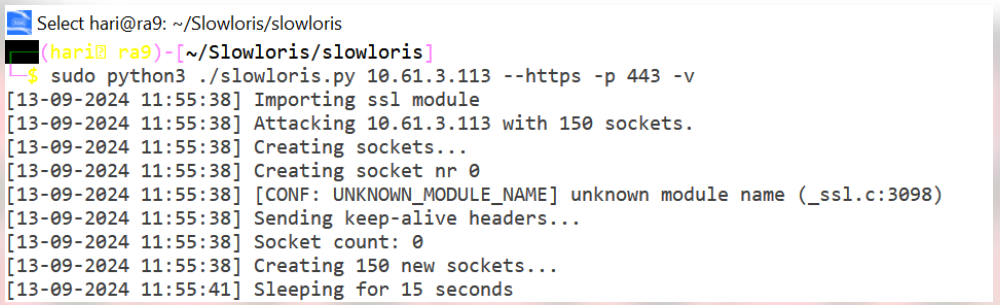


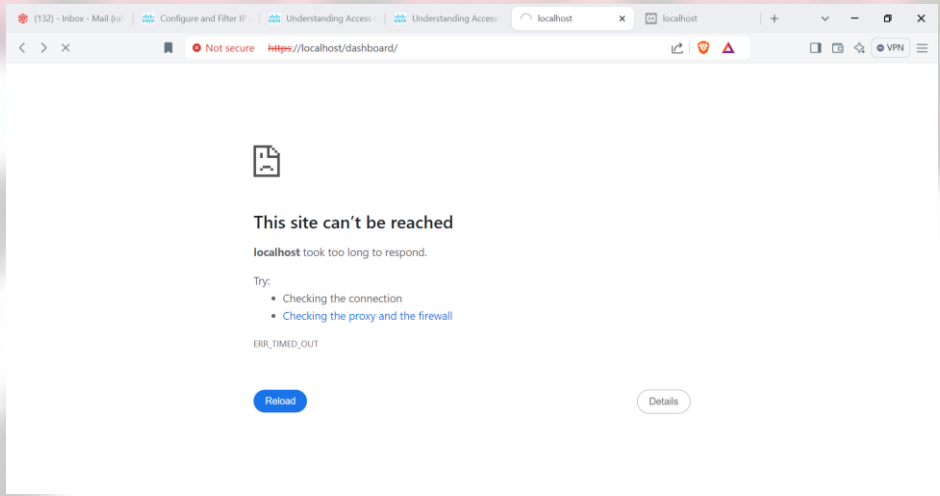
1.2 After running slowloris

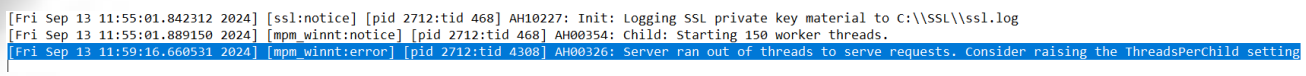
Slowloris.py -> python script for generating partial HTTP request

--https -> use HTTPS request packet

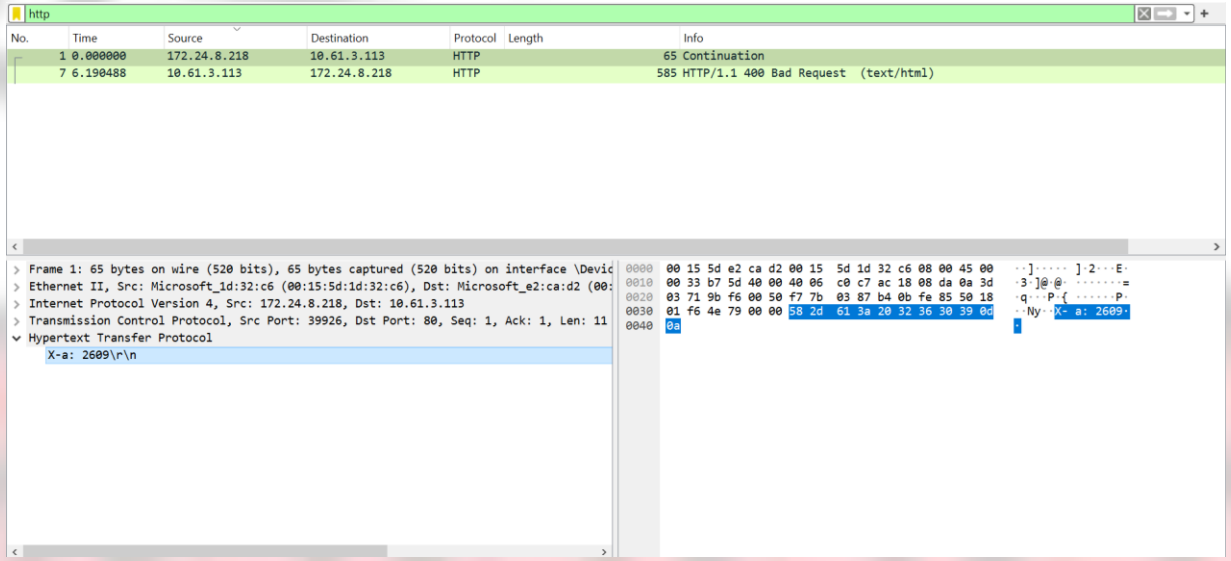
-p -> Webserver Port **-v ->** version

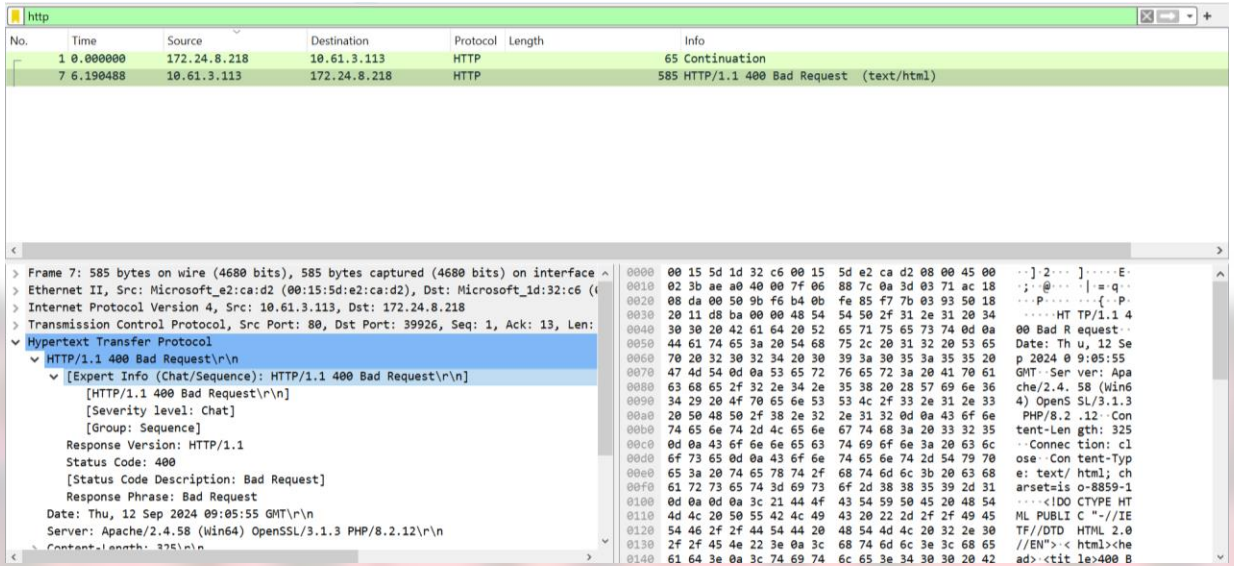






Wireshark Analysis:





Test Case 2 : Send Request
Description : Slowloris tool sending partial HTTP request to Apache(XAMPP) server

Slowloris Command Info:

```
-$ sudo python3 slowloris.py --help
sudo] password for hari:
sage: slowloris.py [-h] [-p PORT] [-s SOCKETS] [-v] [-ua] [-x] [--proxy-host PROXY_HOST] [--proxy-port PROXY_PORT]
                   [--https] [--sleeptime SLEEPTIME]
                   [host]

lowloris, low bandwidth stress test tool for websites

ositional arguments:
 host                  Host to perform stress test on

ptions:
 -h, --help            show this help message and exit
 -p PORT, --port PORT  Port of webserver, usually 80
 -s SOCKETS, --sockets SOCKETS
                       Number of sockets to use in the test
 -v, --verbose         Increases logging
 -ua, --randuseragents
                       Randomizes user-agents with each request
 -x, --useproxy        Use a SOCKS5 proxy for connecting
 --proxy-host PROXY_HOST
                       SOCKS5 proxy host
 --proxy-port PROXY_PORT
                       SOCKS5 proxy port
 --https               Use HTTPS for the requests
 --sleeptime SLEEPTIME
                       Time to sleep between each header sent.
```

Mitigation of Application layer based Attacks :

Activate a WAF: A Web Application Firewall (WAF) is a set of rules or policies that helps protect web applications or APIs from malicious traffic. WAF sits between an application and the HTTP traffic and filters the common web exploits that can affect availability.
There are various WAF solutions available, but you need to analyze which WAF solution is suitable for your application.

Rate Limit
Attackers can make so many repeated calls on the APIs. It can make resources unavailable to its genuine users. A rate limit is the number of API calls or requests that a user can make within a given time frame. When this limit is exceeded, block API access temporarily and return the 429 (too many requests) HTTP error code.

DoS/DDoS Tools used in our ITSAR :

2.8.1 Network Level and application-level DDoS

Requirement:

UPF shall have protection mechanism against Network level and Application-level DDoS attacks.

UPF shall provide security measures to deal with overload situations which may occur as a result of a denial of service attack or during periods of increased traffic. In particular, partial or complete impairment of system availability shall be avoided.

For example, potential protective measures may include:

- Restricting of available RAM per application
- Restricting of maximum sessions for a Web application
- Defining the maximum size of a dataset
- Restricting CPU resources per process
- Prioritizing processes
- Limiting of amount or size of transactions of an user or from an IP address in a specific time range
- Limiting of amount or size of transactions to an IP address/Port Address in a specific time range

[Reference: TEC 25848:2022 / TSDSI STD T1.3GPP 33.117-16.7.0 V.1.0.0. Section 4.2.3.3.1]

DoS/DDoS Tools used in our ITSAR(Continued) :

2.8.2 Excessive Overload Protection

Requirement:

UPF shall act in a predictable way if an overload situation cannot be prevented. UPF shall be built in this way that it can react on an overload situation in a controlled way.

However, it is possible that a situation happens where the security measures are no longer sufficient. In such case it shall be ensured that UPF cannot reach an undefined and thus potentially insecure, state.

OEM shall provide a technical description of the UPF's Over Load Control mechanisms. (especially whether these mechanisms rely on cooperation of other network elements e.g. RAN)

[Ref: TEC 25848:2022 / TSDSI STD T1.3GPP 33.117-16.7.0 V.1.0.0. Section 4.2.3.3.3]

Thank You