

## *How To: Seguridad lógica de un sistema.*

*Autora: Aidee Miguelina Lorenzo Mejía*

### **Implementación de políticas de seguridad y configuración de IPTables contra un ataque DoS con Hping3 desde ParrotOS**

#### **Introducción:** *Descripción del ataque y conceptos importantes.*

En este trabajo se llevará a cabo la simulación de un ataque DoS (Denegación de Servicio) utilizando la herramienta Hping3 desde una máquina Linux con ParrotOS hacia otra máquina Linux con Ubuntu Server. El ataque será mitigado mediante la implementación de políticas de seguridad configuradas con IPTables, demostrando cómo se puede reducir el impacto del ataque en el servidor objetivo. Además, se analizará el tráfico de red antes y después de la mitigación utilizando Wireshark, una herramienta clave para el monitoreo y análisis de redes.

Un ataque DoS consiste en saturar un servidor o servicio con tráfico malicioso, proveniente de una fuente, con el objetivo de interrumpir su funcionalidad o anular sus servicios. Este tipo de ataque es comúnmente dirigido a servidores web, sistemas críticos y redes empresariales, causando interrupciones en su disponibilidad.

*Aidee Miguelina Lorenzo Mejía*

IPTables, por su parte, es una herramienta integrada en el kernel de Linux que permite configurar y administrar reglas de firewall. Su principal función es filtrar paquetes, permitiendo o bloqueando el tráfico de red según las políticas definidas por el administrador. Operando en la capa 3 del modelo OSI (capa de red), IPTables es fundamental para garantizar la seguridad y la integridad de los sistemas Linux en entornos locales o empresariales.

Con este escenario práctico, se busca demostrar la importancia de las configuraciones de seguridad en entornos Linux, enseñando cómo prevenir y mitigar un ataque de este tipo con herramientas nativas del sistema operativo. Este ejercicio tiene fines estrictamente educativos y está orientado a reforzar conocimientos sobre ciberseguridad, análisis de redes y la correcta configuración de sistemas Linux.

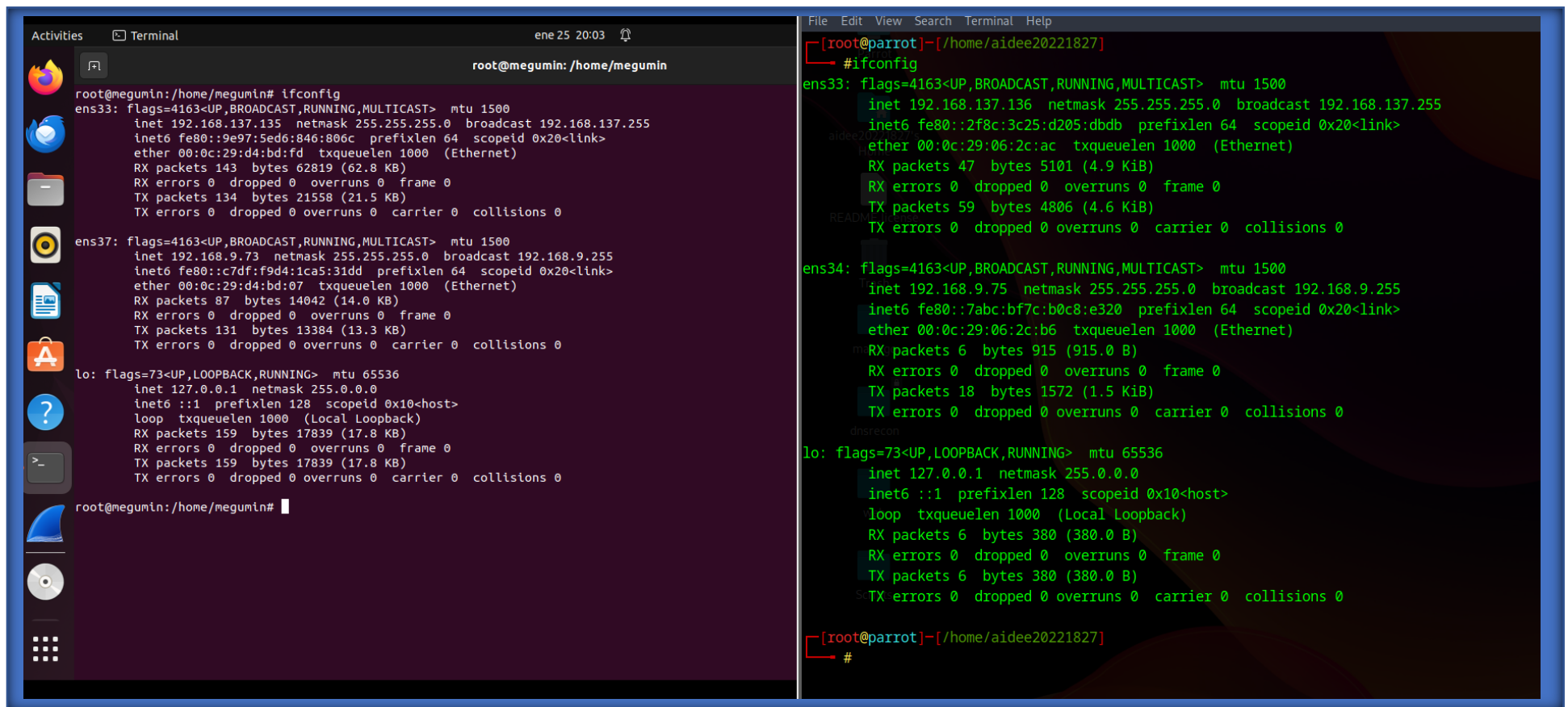
## Contenido

<b>Introducción:</b> Descripción del ataque y conceptos importantes.....	0
<b>Desarrollo:</b> Pasos del How To.....	3
<b>Primera parte: Comunicación entre máquinas.....</b>	3
<b>Segunda parte: Instalación de herramientas necesarias. ....</b>	5
<b>Tercera parte: Ejecución del ataque. ....</b>	8
<b>Cuarta parte: monitoreo del ataque con netstat y wireshark. ....</b>	9
<b>Quinta parte: mitigación del ataque con IPTables. ....</b>	13
<b>¡ALTO AHÍ! ¿Qué son las Syncookies? .....</b>	15
<b>Sexta parte: ¡Atácame de nuevo! .....</b>	17
<b>Conclusión:</b> Reflexión del ataque en el mundo real.....	20

## Desarrollo: Pasos del How To.

### Primera parte: Comunicación entre máquinas.

1. Confirmación de que las máquinas estén en una misma red con el comando **ifconfig** o cualquier otro que ayude a ver las IP de la máquina. La red en este caso es **192.168.9.0 /24**.



The image shows two terminal windows side-by-side. The left window is titled 'Terminal' and shows the output of the 'ifconfig' command on a machine named 'megumin'. The right window is titled 'Terminal' and shows the output of the 'ifconfig' command on a machine named 'parrot'.

```
root@megumin: /home/megumin# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.137.135 netmask 255.255.255.0 broadcast 192.168.137.255
    inet6 fe80::9e97:5ed6:846:806c prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:d4:bd:fd txqueuelen 1000 (Ethernet)
    RX packets 143 bytes 62819 (62.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 134 bytes 21558 (21.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens37: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.9.73 netmask 255.255.255.0 broadcast 192.168.9.255
    inet6 fe80::c7df:f9d4:1ca5:31dd prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:d4:bd:07 txqueuelen 1000 (Ethernet)
    RX packets 87 bytes 14042 (14.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 131 bytes 13384 (13.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 159 bytes 17839 (17.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 159 bytes 17839 (17.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@megumin: /home/megumin#
```

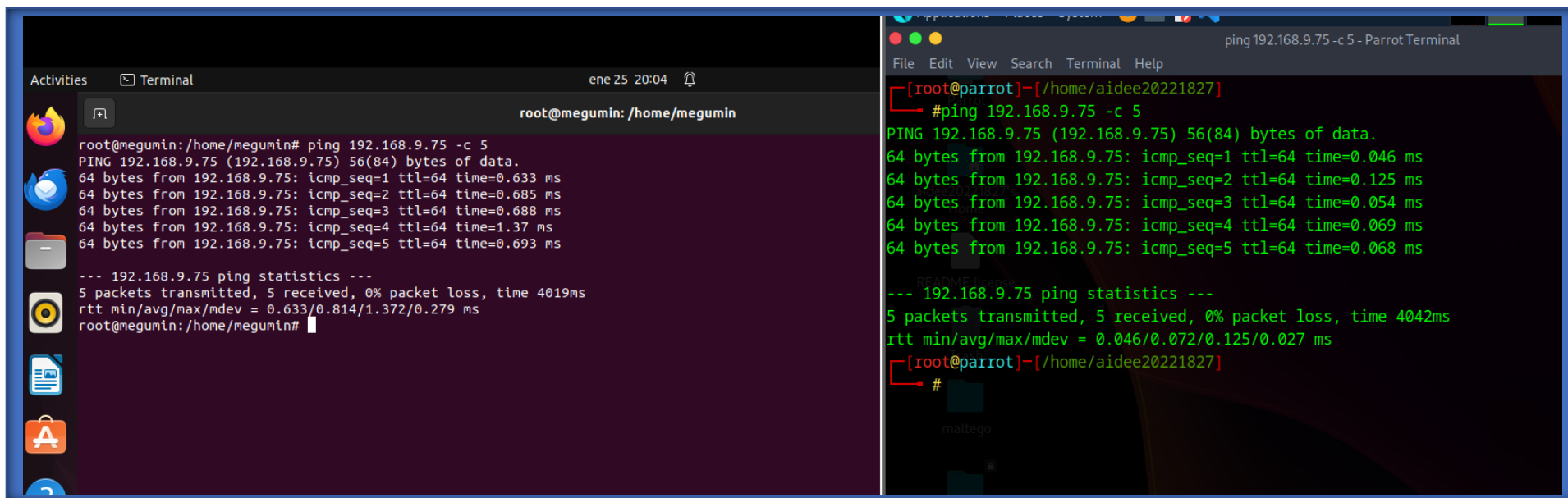
```
[root@parrot]~[/home/aidee20221827]
# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.137.136 netmask 255.255.255.0 broadcast 192.168.137.255
    inet6 fe80::2f8c:3c25:d205:dbdb prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:06:2c:ac txqueuelen 1000 (Ethernet)
    RX packets 47 bytes 5101 (4.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 59 bytes 4806 (4.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens34: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.9.75 netmask 255.255.255.0 broadcast 192.168.9.255
    inet6 fe80::7abc:bf7c:b0c8:e320 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:06:2c:b6 txqueuelen 1000 (Ethernet)
    RX packets 6 bytes 915 (915.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18 bytes 1572 (1.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 6 bytes 380 (380.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6 bytes 380 (380.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@parrot]~[/home/aidee20221827]
#
```

2. Dar ping de una máquina a otra. Si hay comunicación entre ellas, pasa al siguiente paso. De lo contrario, integra ambas máquinas a una misma red para que el ataque logre realizarse.



The image shows two terminal windows side-by-side. The left window is titled 'Terminal' and shows a ping command being executed from a machine named 'megumin' to the IP address 192.168.9.75. The output shows five successful ping requests with varying response times, followed by a summary of the statistics. The right window is titled 'ping 192.168.9.75 -c 5 - Parrot Terminal' and shows the same ping command being executed from a machine named 'parrot'. The output is similar, showing five successful ping requests and a summary of the statistics.

```
root@megumin: /home/megumin
root@megumin:/home/megumin# ping 192.168.9.75 -c 5
PING 192.168.9.75 (192.168.9.75) 56(84) bytes of data:
64 bytes from 192.168.9.75: icmp_seq=1 ttl=64 time=0.633 ms
64 bytes from 192.168.9.75: icmp_seq=2 ttl=64 time=0.685 ms
64 bytes from 192.168.9.75: icmp_seq=3 ttl=64 time=0.688 ms
64 bytes from 192.168.9.75: icmp_seq=4 ttl=64 time=1.37 ms
64 bytes from 192.168.9.75: icmp_seq=5 ttl=64 time=0.693 ms

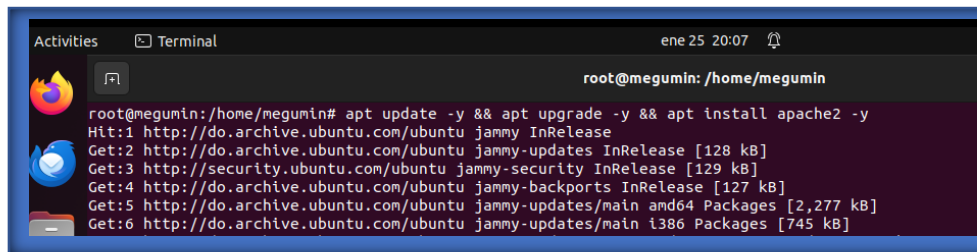
--- 192.168.9.75 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4019ms
rtt min/avg/max/mdev = 0.633/0.814/1.372/0.279 ms
root@megumin:/home/megumin#
```

```
ping 192.168.9.75 -c 5 - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]-[/home/aidee20221827]
#ping 192.168.9.75 -c 5
PING 192.168.9.75 (192.168.9.75) 56(84) bytes of data:
64 bytes from 192.168.9.75: icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from 192.168.9.75: icmp_seq=2 ttl=64 time=0.125 ms
64 bytes from 192.168.9.75: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 192.168.9.75: icmp_seq=4 ttl=64 time=0.069 ms
64 bytes from 192.168.9.75: icmp_seq=5 ttl=64 time=0.068 ms

--- 192.168.9.75 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4042ms
rtt min/avg/max/mdev = 0.046/0.072/0.125/0.027 ms
[root@parrot]-[/home/aidee20221827]
#
```

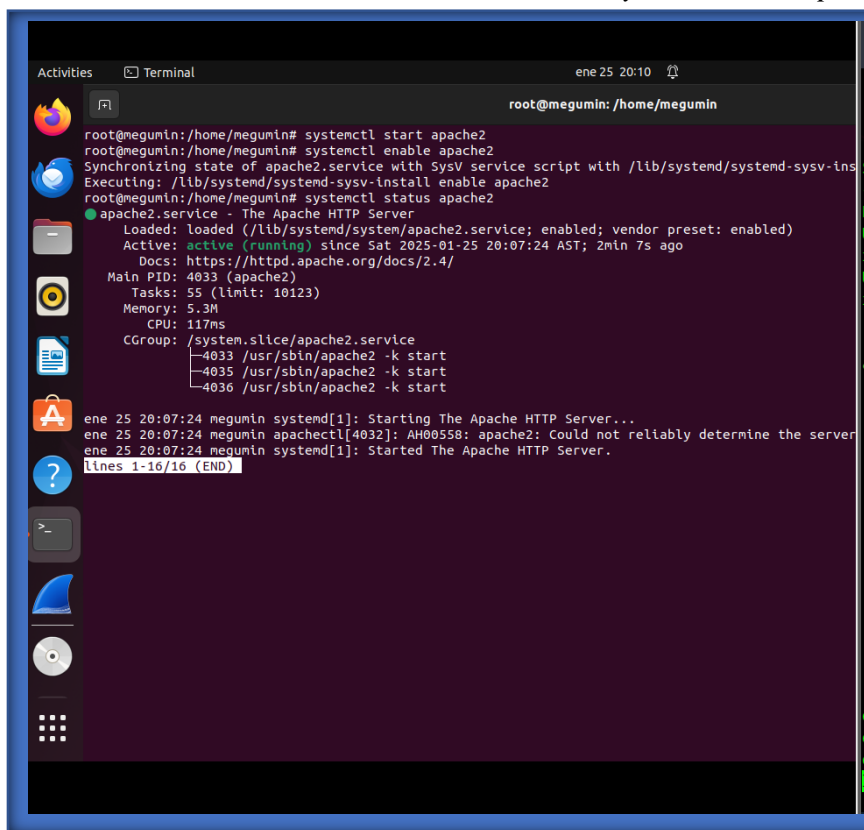
## Segunda parte: Instalación de herramientas necesarias.

3. Instalación de las siguientes herramientas en ambas máquinas: apache2... Antes de la descarga, asegúrese de tener los repositorios actualizados.



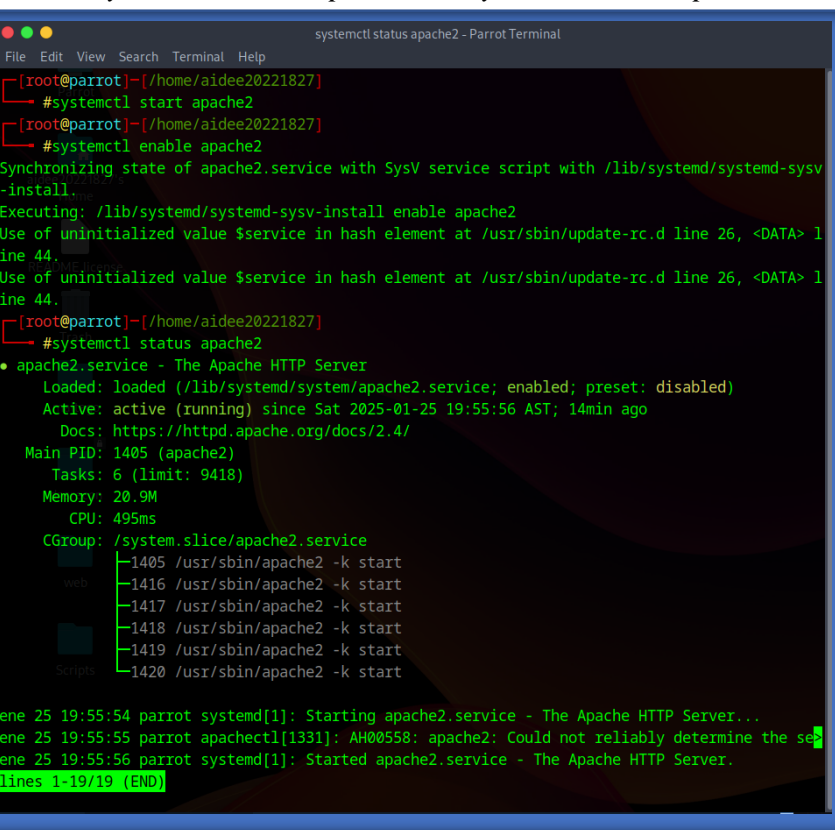
```
root@megumin:/home/megumin# apt update -y && apt upgrade -y && apt install apache2 -y
Hit:1 http://do.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://do.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://do.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://do.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,277 kB]
Get:6 http://do.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [745 kB]
```

4. Habilidad del servicio. **Comando:** `systemctl start apache2` → `systemctl enable apache2` → `systemctl status apache2`.



```
root@megumin:/home/megumin# systemctl start apache2
root@megumin:/home/megumin# systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install
Executing: /lib/systemd/systemd-sysv-install enable apache2
root@megumin:/home/megumin# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-01-25 20:07:24 AST; 2min 7s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 4033 (apache2)
    Tasks: 55 (limit: 10123)
   Memory: 5.3M
      CPU: 117ms
   CGroup: /system.slice/apache2.service
           └─4033 /usr/sbin/apache2 -k start
           └─4035 /usr/sbin/apache2 -k start
           └─4036 /usr/sbin/apache2 -k start

ene 25 20:07:24 megumin systemd[1]: Starting The Apache HTTP Server...
ene 25 20:07:24 megumin apache2[4032]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, because the hostname 'megumin' has no domain name associated with it. Please see the mod_ssl output for more information.
ene 25 20:07:24 megumin systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

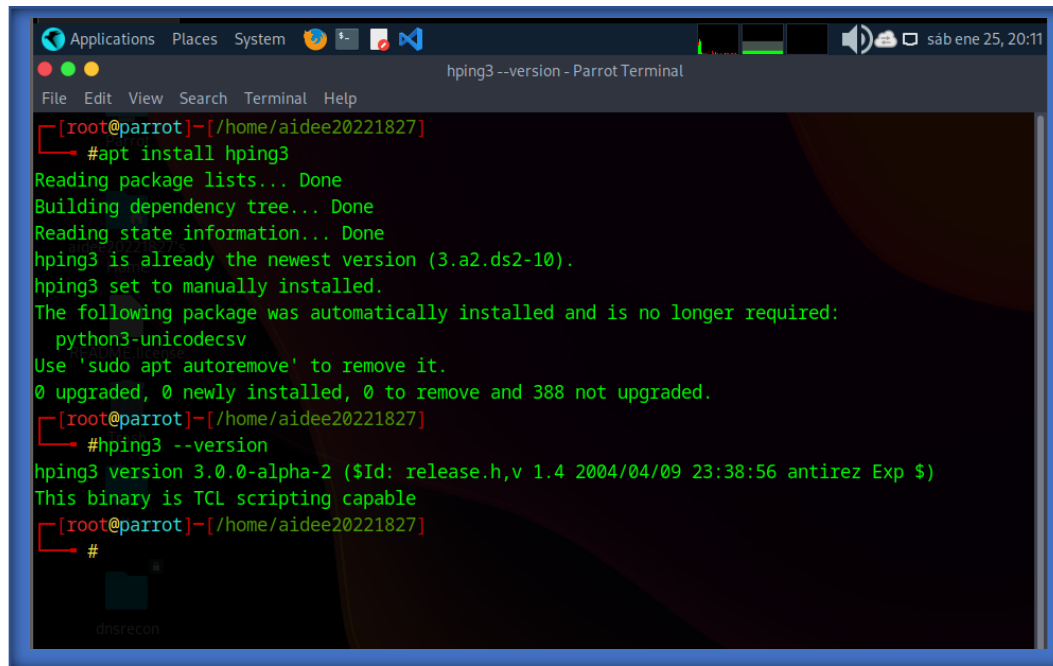


```
[root@parrot]~[/home/aidee20221827]
#apt update -y && apt upgrade -y && install apache2 -y
Get:1 https://deb.parrot.sh/parrot lory InRelease [29.8 kB]
Get:2 https://deb.parrot.sh/direct/parrot lory-security InRelease [29.4 kB]
Get:3 https://deb.parrot.sh/parrot lory-backports InRelease [29.7 kB]
Err:1 https://deb.parrot.sh/parrot lory InRelease
       The following signatures couldn't be verified because the public key is not available: NO KEY 7A8286AF0E81EE4A

[root@parrot]~[/home/aidee20221827]
#systemctl start apache2
[root@parrot]~[/home/aidee20221827]
#systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install
Executing: /lib/systemd/systemd-sysv-install enable apache2
Use of uninitialized value $service in hash element at /usr/sbin/update-rc.d line 26, <DATA> 1 line 44.
Use of uninitialized value $service in hash element at /usr/sbin/update-rc.d line 26, <DATA> 1 line 44.
[root@parrot]~[/home/aidee20221827]
#systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: disabled)
   Active: active (running) since Sat 2025-01-25 19:55:56 AST; 14min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 1405 (apache2)
    Tasks: 6 (limit: 9418)
   Memory: 20.9M
      CPU: 495ms
   CGroup: /system.slice/apache2.service
           └─1405 /usr/sbin/apache2 -k start
           └─1416 /usr/sbin/apache2 -k start
           └─1417 /usr/sbin/apache2 -k start
           └─1418 /usr/sbin/apache2 -k start
           └─1419 /usr/sbin/apache2 -k start
           └─1420 /usr/sbin/apache2 -k start

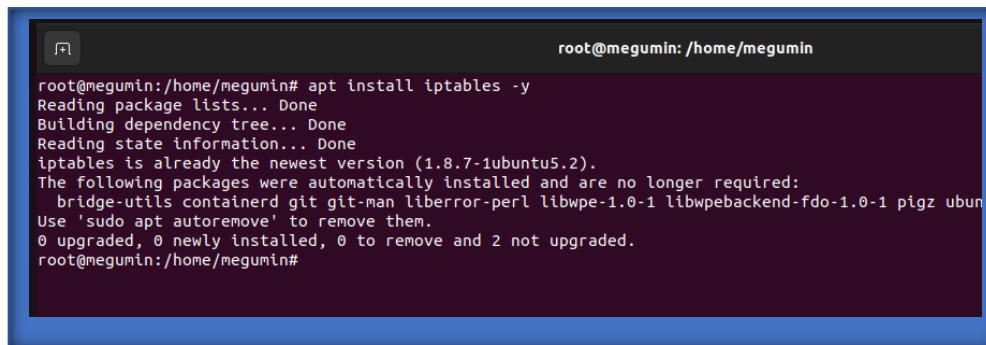
ene 25 19:55:54 parrot systemd[1]: Starting apache2.service - The Apache HTTP Server...
ene 25 19:55:55 parrot apache2[1331]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, because the hostname 'parrot' has no domain name associated with it. Please see the mod_ssl output for more information.
ene 25 19:55:56 parrot systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-19/19 (END)
```

## 5. Instalación de **hping3** en ParrotOS.



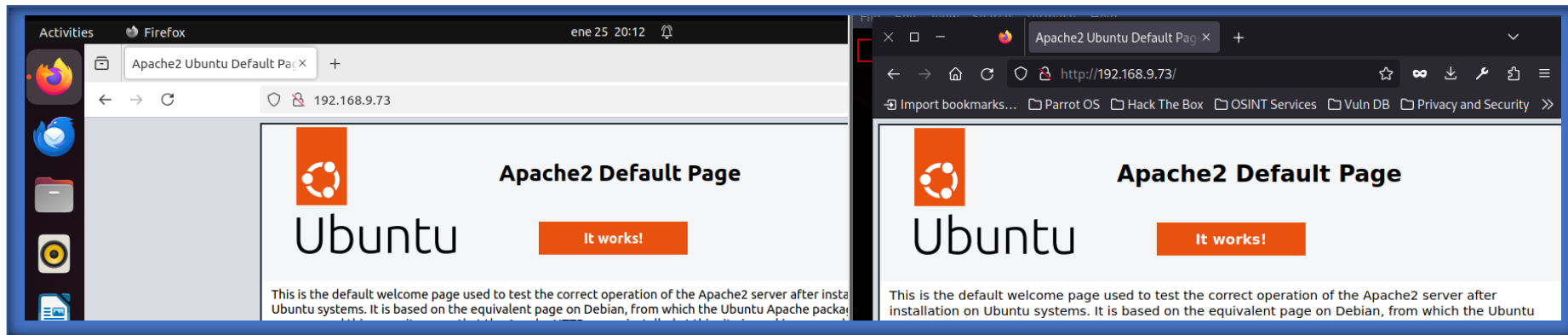
```
hping3 --version - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]-[/home/aidee20221827]
#apt install hping3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
hping3 is already the newest version (3.a2.ds2-10).
hping3 set to manually installed.
The following package was automatically installed and is no longer required:
  python3-unicodecsv
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 388 not upgraded.
[root@parrot]-[/home/aidee20221827]
#hping3 --version
hping3 version 3.0.0-alpha-2 ($Id: release.h,v 1.4 2004/04/09 23:38:56 antirez Exp $)
This binary is TCL scripting capable
[root@parrot]-[/home/aidee20221827]
#
```

## 6. Instalación de iptables en Ubuntu.



```
root@megumin: /home/megumin
root@megumin:/home/megumin# apt install iptables -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iptables is already the newest version (1.8.7-1ubuntu5.2).
The following packages were automatically installed and are no longer required:
  bridge-utils containerd git git-man liberror-perl libwpe-1.0-1 libwpebackend-fdo-1.0-1 pigz ubun
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
root@megumin:/home/megumin#
```

7. Al tener todas las herramientas necesarias, confirmará si el servidor web de la máquina víctima está funcionando. Para esto, coloque en el buscador: **http://<ip de la máquina víctima>**.





### Tercera parte: Ejecución del ataque.

8. Desde nuestra máquina atacante, con la herramienta hping3, ejecutarás el siguiente comando: **hping3 -S -p 80 --flood 192.168.9.73**.

Te explico para qué es cada argumento:

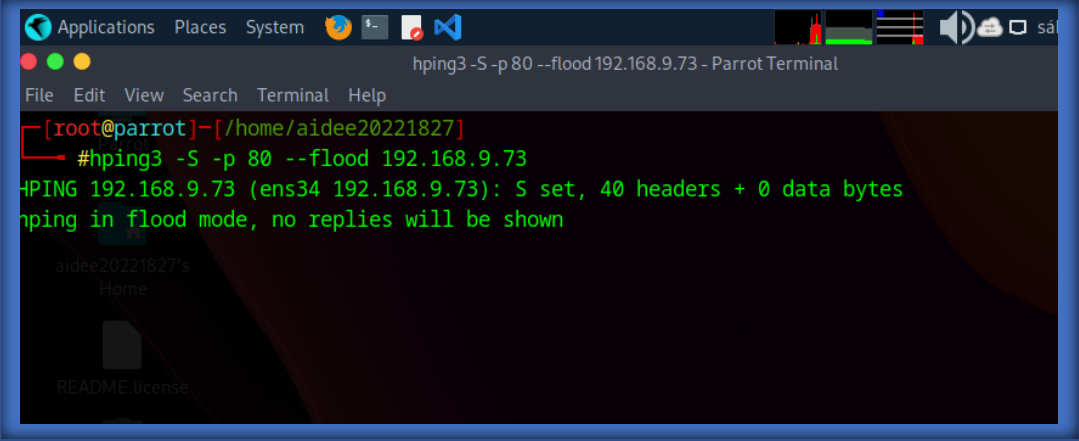
**-S** envía paquetes de tipo TCP con el flag SYN.

**-p 80** especifica el puerto destino al cuál se enviarán los paquetes. En este caso atacaremos el puerto HTTP.

**--flood** hace que los paquetes se envíen tan rápido como sea posible.

**192.168.9.73** es la ip del servidor Ubuntu, la víctima. Si en tu caso tienes una IP distinta, es la que corresponde al final del comando.

Al aplicar el comando, esto debe aparecer en tu pantalla:



```
Applications  Places  System  [Icons]  [Volume]  [Network]  [Battery]  [Clock]  [Logout]
hping3 -S -p 80 --flood 192.168.9.73 - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]~[/home/aidee20221827]
#hping3 -S -p 80 --flood 192.168.9.73
HPING 192.168.9.73 (ens34 192.168.9.73): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

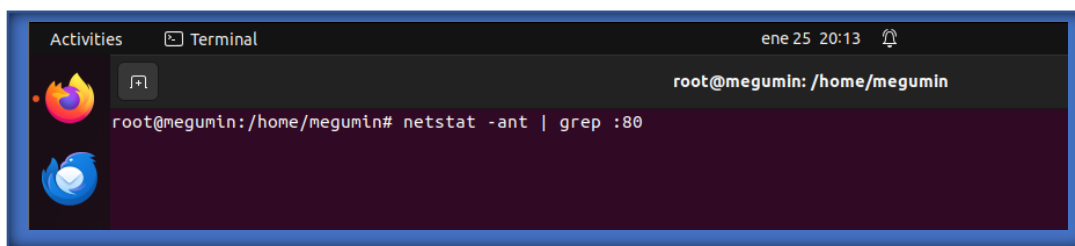
aidee20221827's
Home
README.license
```

## Cuarta parte: monitoreo del ataque con netstat y wireshark.

9. Con el siguiente comando: **netstat -ant | grep :80**, identificaremos el volumen de tráfico y origen del ataque.

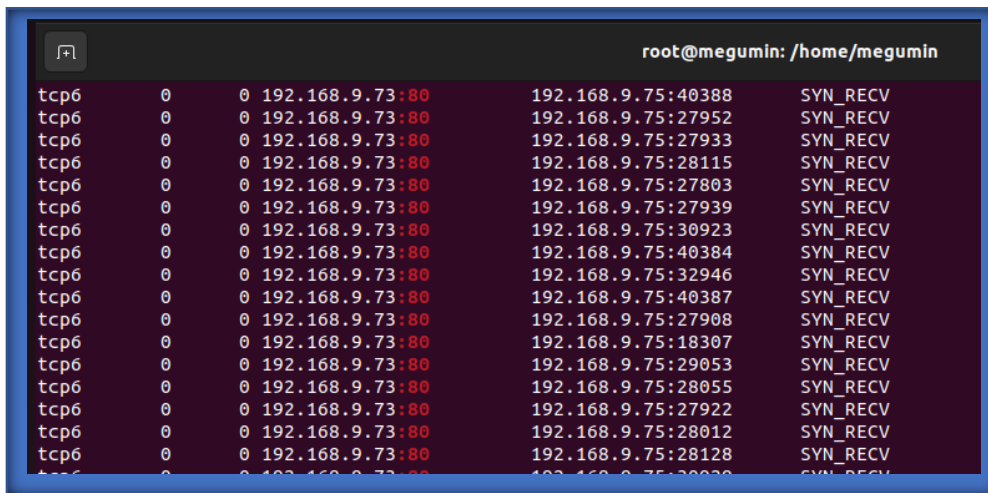
El comando **netstat -ant** muestra todas aquellas conexiones activas y puertos que están en uso.

**grep :80** filtrará las conexiones del puerto HTTP, es decir, el puerto 80.



```
Activities Terminal ene 25 20:13
root@megumin: /home/megumin
root@megumin:/home/megumin# netstat -ant | grep :80
```

Al ejecutar el comando, esto debería verse en tu pantalla:



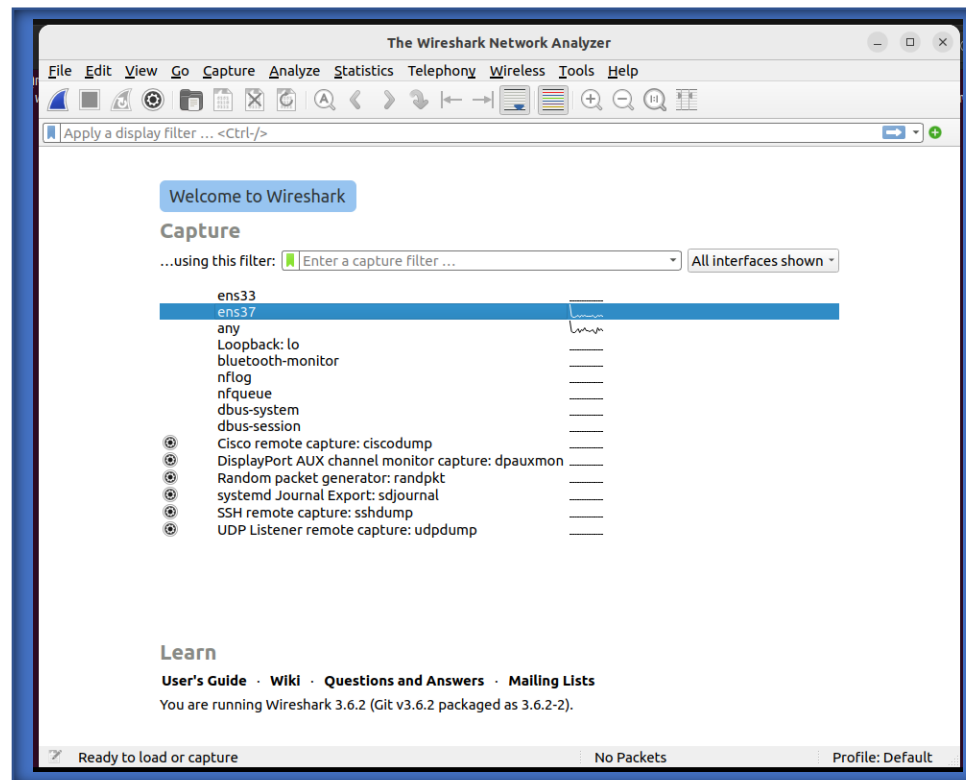
```
root@megumin: /home/megumin
tcp6      0      0 192.168.9.73:80      192.168.9.75:40388    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:27952    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:27933    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:28115    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:27803    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:27939    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:30923    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:40384    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:32946    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:40387    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:27908    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:18307    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:29053    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:28055    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:27922    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:28012    SYN_RECV
tcp6      0      0 192.168.9.73:80      192.168.9.75:28128    SYN_RECV
```

Las muchas conexiones **SYN\_RECV** desde una misma o distintas IP, señalan el ataque DoS.

10. Para ejecutar la herramienta wireshark desde la terminal, simplemente escriba el nombre de la herramienta en la terminal. O ejecútela desde la interfaz gráfica. Para evitar inconvenientes de permisos, lo ideal es ejecutar desde el terminal.

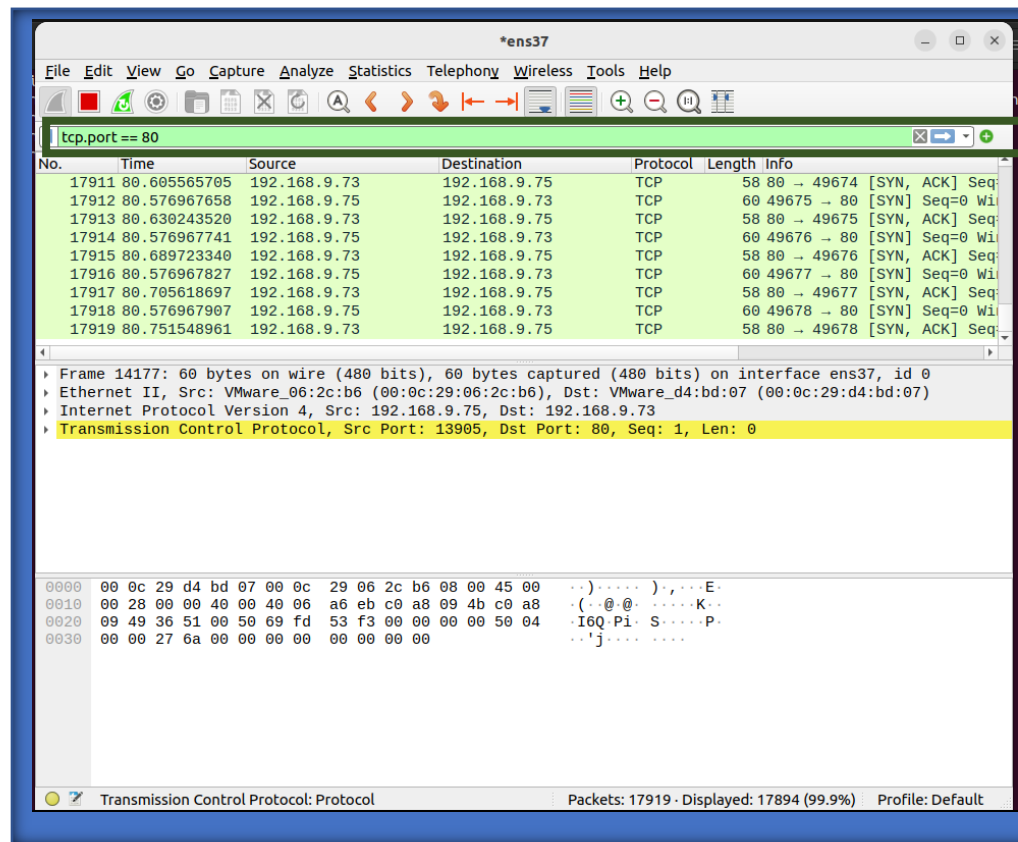
```
root@megumin:/home/megumin# wireshark
** (wireshark:5600) 20:25:47.370930 [GUI WARNING] -- QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-r
```

11. Una vez abierta la herramienta, seleccionas la interfaz de tu red. Con el comando **ifconfig**, utilizado al inicio de esta práctica, podrá confirmarla. En este caso, la interfaz correspondiente a la red es la **ens37**. Al seleccionarla, inicie la captura de tráfico.



12. Esto es lo que se aprecia:

- El atacante de ip **192.168.9.75** envía paquetes **TCP** con solicitudes **SYN**, las cuales forman parte del ***Three-Way Handshake***.
- El servidor web de ip **192.168.9.73** le responde con **SYN, ACK**. En términos simples le dice: Recibí tu solicitud, estoy listo para continuar con la conexión.



- Por último, debería haber paquetes **ACK** por parte de la ip **.75** del atacante, los cuales confirmarían la comunicación entre la máquina y el servidor. Pero nunca llega. Lo cual confirma que, efectivamente está habiendo un **ataque DoS**. Ya que el atacante **pide** el servicio, el servidor le responde, pero **no existen** más respuestas. El atacante solo está para enviar solicitudes infinitas, más nunca responde nada.

Si te fijas, en el recuadro **verde oscuro**, está puesto el filtro **tcp.port == 80**. Esto ayuda a encontrar el tipo de tráfico que estamos buscando. Si ves solicitudes inusuales, este tipo de filtros ayuda a encontrar con más facilidad lo que probablemente buscas.

## Quinta parte: mitigación del ataque con IPTables.

13. Desde la terminal, es hora de ponerle fin a todas esas solicitudes sin respuestas. Con el commando: **iptables -A INPUT -p tcp --dport 80 --syn -m connlimit --connlimit-above 10 -j DROP**.

```
root@megumin:/home/megumin# iptables -A INPUT -p tcp --dport 80 --syn -m connlimit --connlimit-above 10 -j DROP
root@megumin:/home/megumin#
```

Te explico:

**-A INPUT** aplica la regla de conexiones entrantes.

**-p tcp** especifica a la regla los paquetes TCP, para que se aplique a ellos.

**--dport 80** aplica la regla al puerto 80 (HTTP)

**--syn** aplica la regla únicamente a paquetes con flag SYN

**-m connlimit -connlimit-above 10** limitará las conexiones simultáneas por IP a un máximo de 10.

**-j DROP** descarta las conexiones que excedan el límite.

14. Este es más fácil. Con: **iptables -A INPUT -s 192.168.9.75 -j DROP** bloqueamos directamente las solicitudes que vengan de la ip atacante.

En este caso **-s** quiere decir “source”. Es decir, “origen”.

```
root@megumin:/home/megumin# iptables -A INPUT -s 192.168.9.75 -j DROP
root@megumin:/home/megumin#
```

15. Con este último comando: **sysctl -w net.ipv4.tcp\_syncookies=1** activaremos protección contra SYN floods para prevenir ataques SYN flood a nivel del kernel.

**sysctl** es para modificar o leer parámetros del kernel (núcleo) en tiempo de ejecución.

**-w** indica que quiere modificar un valor.

**-net.ipv4.tcp\_syncookies= (1 o 0)** es el parámetro del kernel que habilita o deshabilita las syncookies. 1 = activado. 0 = desactivado.

```
root@megumin:/home/megumin# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
root@megumin:/home/megumin#
```

## ¡ALTO AHÍ! ¿Qué son las Syncookies?

Veamos si hacemos esto en pocas palabras...

Cuando un servidor detecta que está siendo atacado con un **SYN flood**, evita reservar recursos de inmediato para las conexiones pendientes. En su lugar, utiliza **TCP Syncookies** para validar si las solicitudes son legítimas.

Con el comando anteriormente ejecutado, se activa esta defensa, permitiendo al servidor manejar cada solicitud sin consumir recursos innecesariamente. Si la solicitud es legítima, el servidor verifica la información del **Syncookie**, establece la conexión y asigna los recursos correspondientes. Si la solicitud no es legítima, el servidor simplemente la descarta sin impactar su rendimiento. Es decir, si la solicitud **es válida, se procesa; si es falsa, se ignora**.



16. Una vez hecho todo esto, verifica que las reglas estén bien establecidas. Con: **iptables -L -v**, puedes hacerlo.

**-L** listará las reglas actuales en las cadenas de tablas por defecto: **INPUT**, **FORWARD** y **OUTPUT**...

**-v** (verbose) muestra información detallada:

- Número de paquetes que han coincidido con cada regla.
- Cantidad de datos procesados por cada regla (en bytes).
- Interfaces de red involucradas.

```
root@megumin:/home/megumin# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
 140K 5610K DROP      tcp  --  any    any      anywhere          anywhere          tcp dpt:http flags:FIN,SYN,RST,ACK/SYN #con
n src/32 > 10
 2179K  87M DROP      all  --  any    any      192.168.9.75      anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination

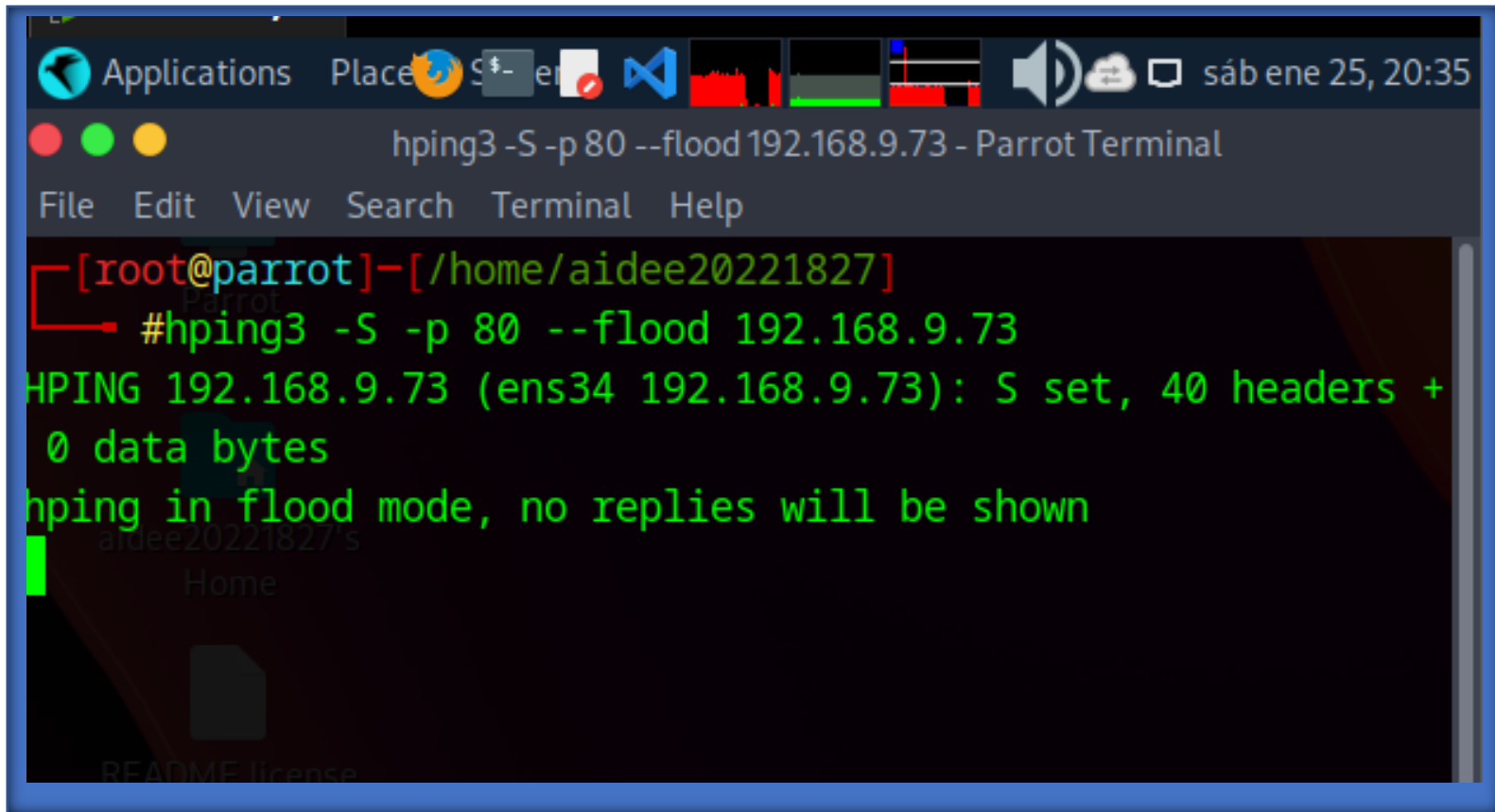
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
root@megumin:/home/megumin#
```

Verificar las reglas es importante, esto ayuda a saber si se está **bloqueando** o **permitiendo** el tráfico no deseado.

## Sexta parte: ¡Atácame de nuevo!

17. Con todas las reglas y medidas ya aplicadas, volverás a ejecutar el ataque desde la máquina Parrot. Repetirás todo lo hecho antes de aplicar las reglas de **iptables**. Te mostraré:

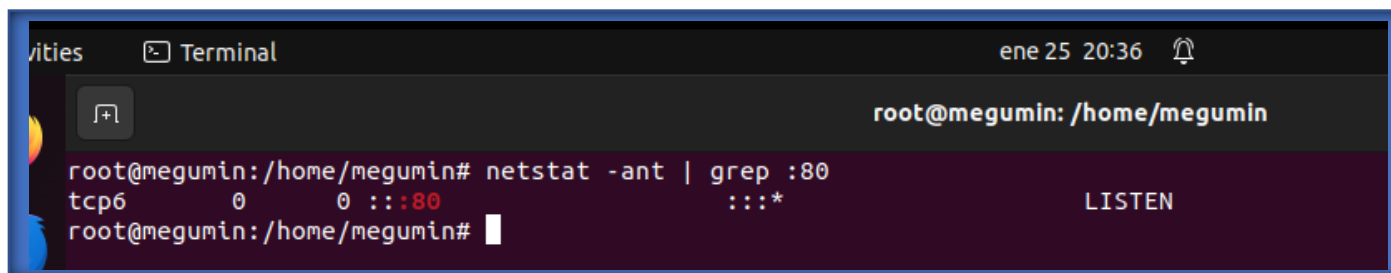
**Primero: Re-ejecuta** el ataque con **hping3**.



The screenshot shows a terminal window titled "hping3 -S -p 80 --flood 192.168.9.73 - Parrot Terminal". The terminal output is as follows:

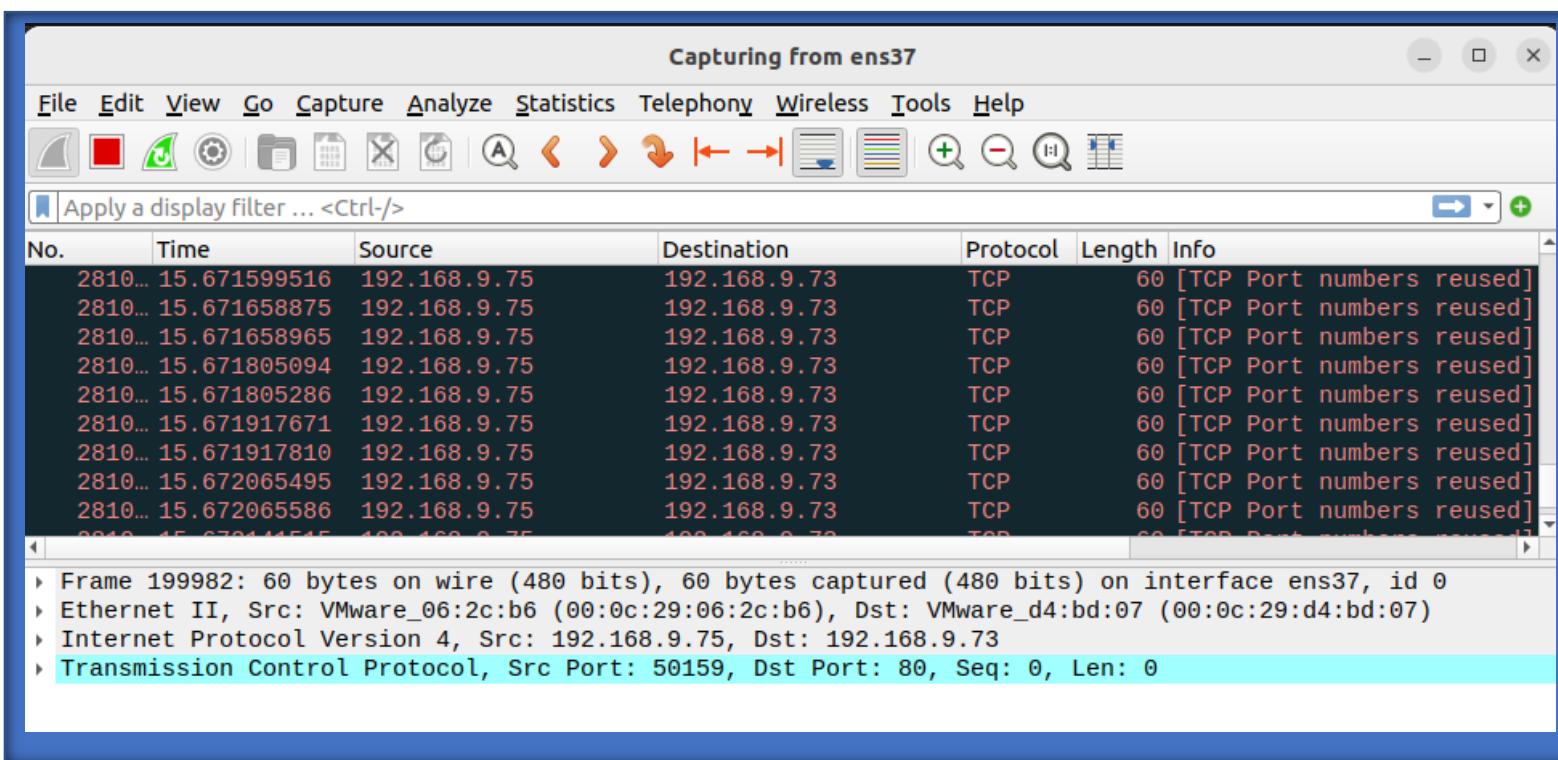
```
[root@parrot]-[/home/aidee20221827]
#hping3 -S -p 80 --flood 192.168.9.73
HPING 192.168.9.73 (ens34 192.168.9.73): S set, 40 headers +
0 data bytes
hping in flood mode, no replies will be shown
```

**Segundo:** Monitorea el tráfico desde tu Ubuntu con netstat.



```
root@megumin:/home/megumin# netstat -ant | grep :80
tcp6      0      0 ::::80          :::*            LISTEN
root@megumin:/home/megumin#
```

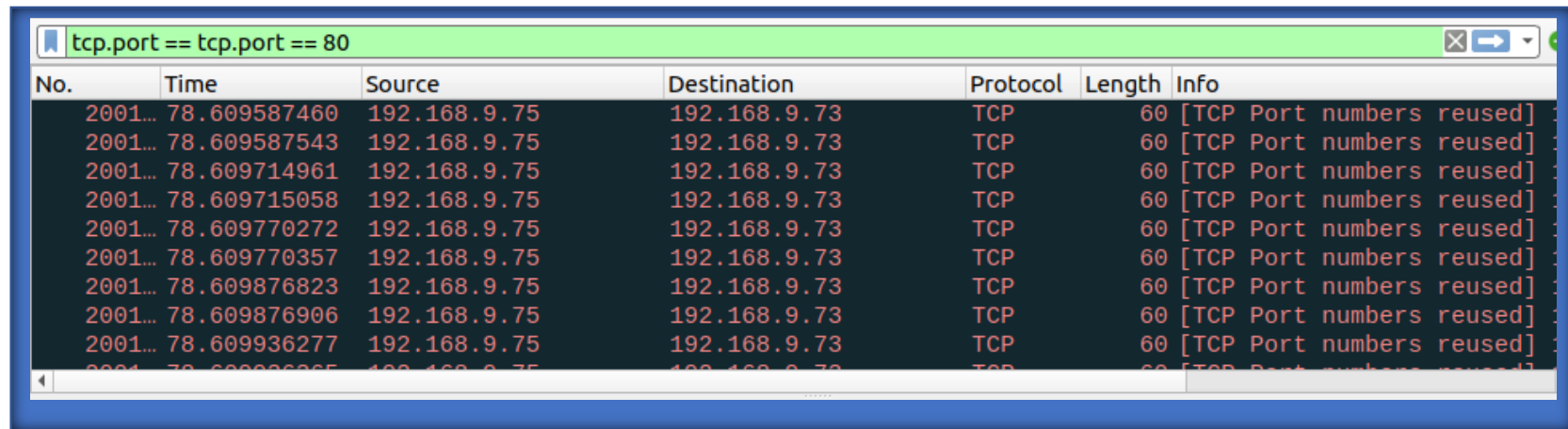
**Tercero:** Monitorea el tráfico desde wireshark. Esta vez, las solicitudes desde la máquina atacante (**ip .75**) deberían llegar de forma **reducida** o **nula** al servidor. Y, **no debe haber** respuestas del servidor (**ip .73**) a aquellas solicitudes **no legítimas**.



No.	Time	Source	Destination	Protocol	Length	Info
2810...	15.671599516	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.671658875	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.671658965	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.671805094	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.671805286	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.671917671	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.671917810	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.672065495	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2810...	15.672065586	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]

Frame 199982: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface ens37, id 0  
Ethernet II, Src: VMware\_06:2c:b6 (00:0c:29:06:2c:b6), Dst: VMware\_d4:bd:07 (00:0c:29:d4:bd:07)  
Internet Protocol Version 4, Src: 192.168.9.75, Dst: 192.168.9.73  
Transmission Control Protocol, Src Port: 50159, Dst Port: 80, Seq: 0, Len: 0

Con el filtro anteriormente usado:



The image shows a Wireshark packet capture window. The filter bar at the top contains the expression `tcp.port == tcp.port == 80`. Below the filter, a table of captured packets is displayed. The table has columns for 'No.', 'Time', 'Source', 'Destination', 'Protocol', 'Length', and 'Info'. There are 10 rows of data, all showing TCP packets from source IP 192.168.9.75 to destination IP 192.168.9.73. Each packet's 'Info' field indicates '[TCP Port numbers reused]'. The packet numbers range from 2001 to 2010.

No.	Time	Source	Destination	Protocol	Length	Info
2001...	78.609587460	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609587543	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609714961	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609715058	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609770272	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609770357	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609876823	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609876906	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609936277	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]
2001...	78.609936365	192.168.9.75	192.168.9.73	TCP	60	[TCP Port numbers reused]

## **Conclusión:** *Reflexión del ataque en el mundo real.*

Este trabajo ha proporcionado una pequeña pero valiosa demostración de uno de los muchos riesgos que se pueden encontrar en el ciberespacio: los ataques de denegación de servicio (DoS). A través de la simulación de un ataque DoS utilizando Hping3 y su mitigación mediante la herramienta IPTables, se ha ilustrado cómo un sistema Linux puede ser tanto vulnerable como protegido frente a este tipo de amenazas. Además, se ha destacado el uso de Wireshark como una herramienta clave para monitorear y analizar el tráfico de red, facilitando la detección de ataques y la evaluación de la efectividad de las contramedidas implementadas.

El propósito principal de este ejercicio ha sido educativo, con el objetivo de enseñar la importancia de implementar buenas prácticas de seguridad en entornos Linux. El manejo adecuado de herramientas como IPTables es esencial para proteger nuestros sistemas y redes, no solo contra ataques DoS, sino también frente a otros tipos de amenazas que podrían comprometer la confidencialidad, integridad y disponibilidad de los datos.

La capacitación en ciberseguridad es fundamental para cualquier profesional de tecnología. Este trabajo subraya la necesidad de comprender cómo funcionan los ataques y cómo se pueden mitigar de manera eficaz. Este ejercicio ha sido una oportunidad para poner en práctica conceptos teóricos y mostrar cómo, con las herramientas adecuadas, es posible minimizar los efectos de un ataque cibernético, refuerza la importancia de aplicar políticas de seguridad robustas y de contar con conocimientos prácticos sobre cómo proteger nuestras infraestructuras críticas.

Al seguir aprendiendo y mejorando nuestras habilidades, podemos contribuir a un ciberespacio más seguro y resistente frente a las amenazas emergentes.