

# **Memoria**

# **Proyecto ASO**

**Marcos Alonso Cabral, ASIR2**

# Índice

<b>1. Descripción de lo utilizado</b>	<b>2</b>
<b>2. Montaje Ubuntu Server (con script)</b>	<b>4</b>
<b>3. Instalación de Apache2</b>	<b>8</b>
<b>4. Montaje de la función Email</b>	<b>9</b>
<b>5. Montaje del script de mitigación</b>	<b>13</b>
<b>6. Montaje kali (Instalación herramientas)</b>	<b>14</b>
<b>7. Simulación de situación real</b>	<b>17</b>
<b>8. Instalando crontab</b>	<b>22</b>
<b>9. Comprobando los correos</b>	<b>23</b>
<b>10. Comparación con aplicaciones externas.</b>	<b>24</b>
Características de mi script	24
Características de Aplicaciones Similares	25
<b>11. Posibles ampliaciones</b>	<b>27</b>
<b>12. Errores durante la realización</b>	<b>28</b>
<b>13. Webgrafía</b>	<b>33</b>

Para comenzar, este proyecto tratará a grandes rasgos, de un script que localiza y elimina las posibles IPS maliciosas (medidas de prevención de posibles ataques Ddos) de manera automatizada. Todo esto con un repositorio en github donde dejaré todos los pasos y guías relacionados con la realización de este proyecto.

A continuación comenzamos con todos los pasos de este proyecto:

## **1. Descripción de lo utilizado**

**Ubuntu Server:** Para la realización de este proyecto utilizaremos la versión 22.04.5 de ubuntu server, esta se puede obtener de la siguiente dirección: <https://ubuntu.com/download/server>. Además configuraremos el servidor con su DHCP y DNS correspondiente, utilizando un script creado por mi que te permite elegir el nombre del dominio.

**Servidor web Apache2:** Una parte muy relevante de este proyecto es el servidor apache2 que instalaremos en el Ubuntu Server ya que pondremos a prueba su capacidad de resistir a muchas peticiones http.

También crearemos una página web personalizada para que podamos comprobar si realmente se ha caído el servicio (aunque realmente el servicio al ser tan liviano, no se caerá y debere ver la diferencia de tiempos de respuesta para comprobar que funciona).

**Script de prevención en bash:** La parte fundamental de este proyecto, sin duda es este script que a pesar de que es relativamente sencillo, será lo que realmente investigaremos y compararemos con otras alternativas en el mercado (aplicaciones). Este script, que se podrá ver más adelante, consta de varias partes:

1. Verifica los permisos con "sudo".
2. Establece un límite de conexiones, que puede ser cambiado, pero que en un principio está en 100 conexiones.

3. Sus funciones principales y la razón principal de este script es que, primero detecta patrones maliciosos mediante el uso de arrays “como inyecciones SQL (sqlmap), escaneos (nmap, nikto), y ataques de fuerza bruta (hydra).” Y luego busca y bloquea conexiones excesivas.
4. Si el script encuentra una ip con demasiadas conexiones, la bloquea añadiéndola a una lista y a las reglas de iptables para que esa ip y rango (solo el último octeto de las posibles direcciones ip) no puedan acceder hasta dentro de una hora.
5. El script también cuenta con una función de correo, es decir, que cada vez que se ejecute, si encuentra alguna ip que bloquear, manda un correo al correo que hayamos configurado en el script del propio correo.
6. Y por último, hace una limpieza de logs de apache, comprimiendolos cada hora en un backup para que al borrarse las reglas de iptables, no se vuelvan a bloquear las ips bloqueadas anteriormente sin realizar conexiones.(Cabe destacar que si el archivo comprimido supera 1GB, se elimina para ahorrar espacio.)

**Kali Linux:** Instalaremos una máquina de kali que será la máquina atacante y en donde aprenderemos algunas de las técnicas y herramientas de ataque enfocadas a el tipo de ataque DDOS.

**Repositorio en github:** Para este proyecto crearé un repositorio exclusivamente para este proyecto, en el que encontraremos entre otras cosas el código usado para configurar el DHCP y el DNS, como el otro script de prevención de DDOS, luego otro script para configurar los mensajes de correo y las url con toda la información de las isos y herramientas utilizadas.

Link al repositorio: [https://github.com/Malonsocabral/Proyecto\\_ASO\\_Marcos](https://github.com/Malonsocabral/Proyecto_ASO_Marcos)

Cabe destacar que los scripts tienen un número identificador al comienzo (010,020,...), lo que muestra en qué orden hay que ejecutarlos, (ya que de nada

vale ejecutar el script de mitigación si no configuramos el correo anteriormente).

Además cada script se puede personalizar según las necesidades de quien quiera volver a realizar esta práctica (por ejemplo ya sea cambiando el número de conexiones en el script de mitigación o cambiando el nombre del dominio así como el rango de ips del DHCP en el script de Montaje de Ubuntu Server).

## **2. Montaje Ubuntu Server (con script)**

Para comenzar este segundo paso, procedemos al montaje de un ubuntu server en una máquina virtual. Utilizaremos como base la guía del fichero del repositorio de github ([Instalacion y configuracion router en Ubuntu Server Marcos](#)) y el script personalizado del mismo, en el que podemos elegir el nombre del dominio y donde la ip del server es 192.168.0.1.

Dicho esto, procederemos a la instalación de este UbuntuServer-22.04.5 y una vez descargado la ISO, creamos una nueva máquina virtual y seguimos los pasos para una instalación normal:

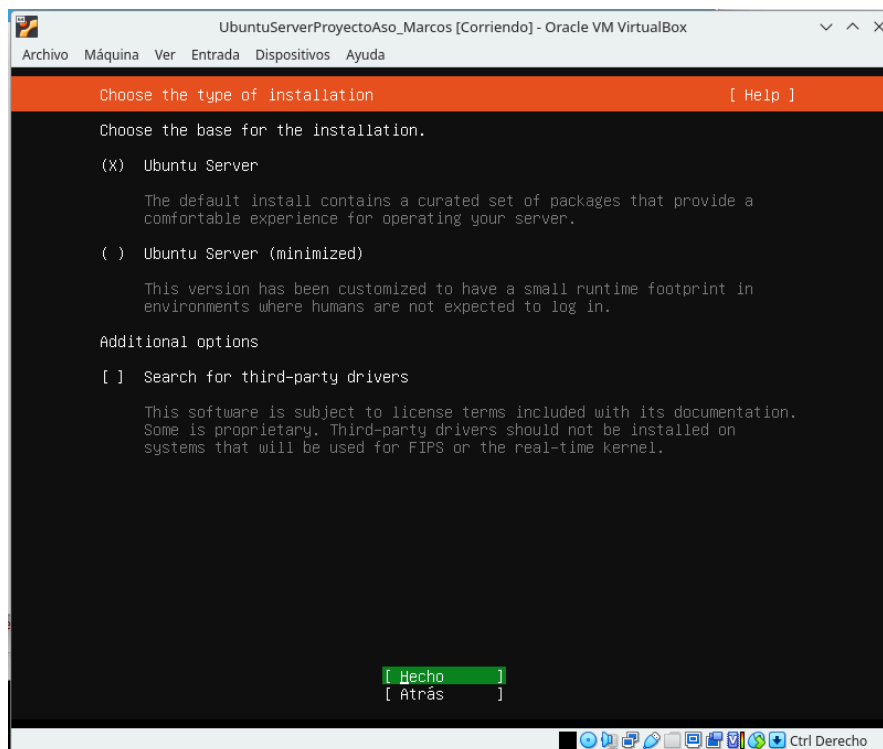


Imagen ilustrativa

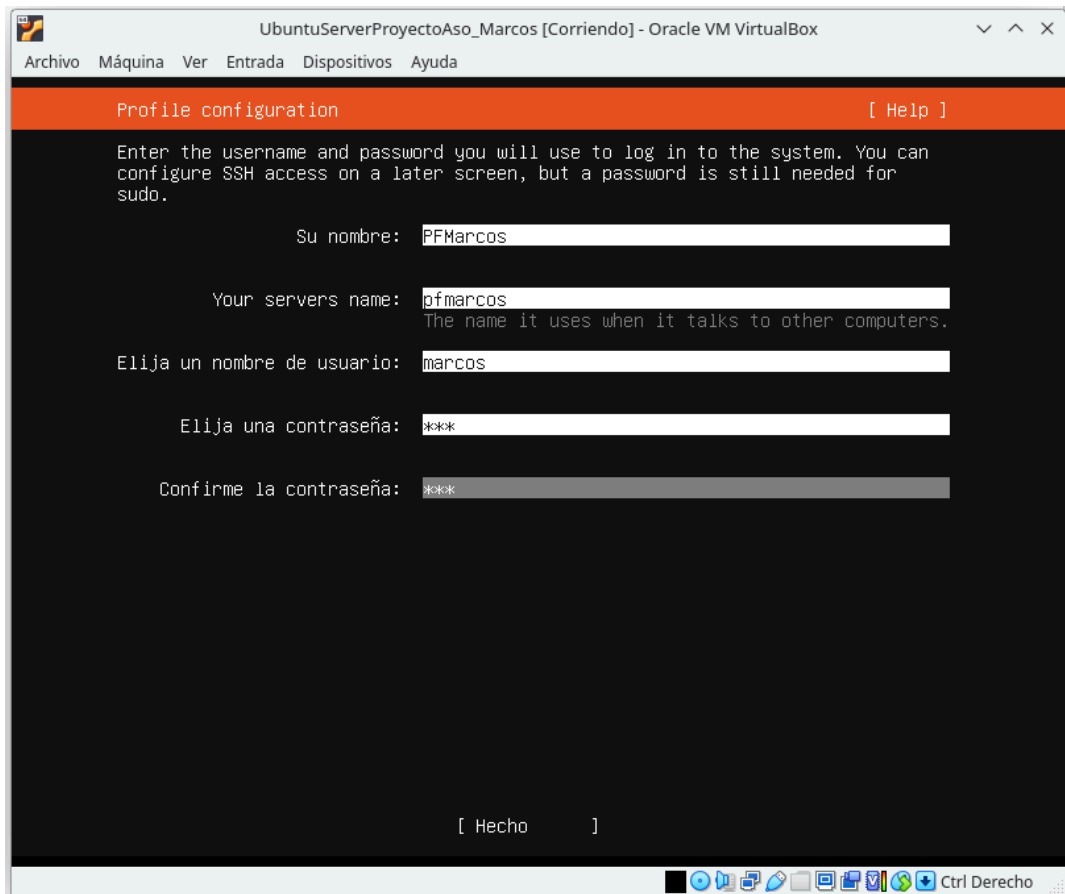


Imagen ilustrativa creando mis credenciales

Antes de realizar la instalación de DHCP y DNS con mi script, una cosa importante es que debemos apagar la máquina y habilitar en VirtualBox la segunda interfaz de red, ya que el script crea un netplan, pensado para que la primera interfaz sea en modo NAT y la segunda interfaz sea en Red interna.

Esto se realiza para que una interfaz coja toda la información de fuera y la otra simplemente se conecte con los posibles clientes.

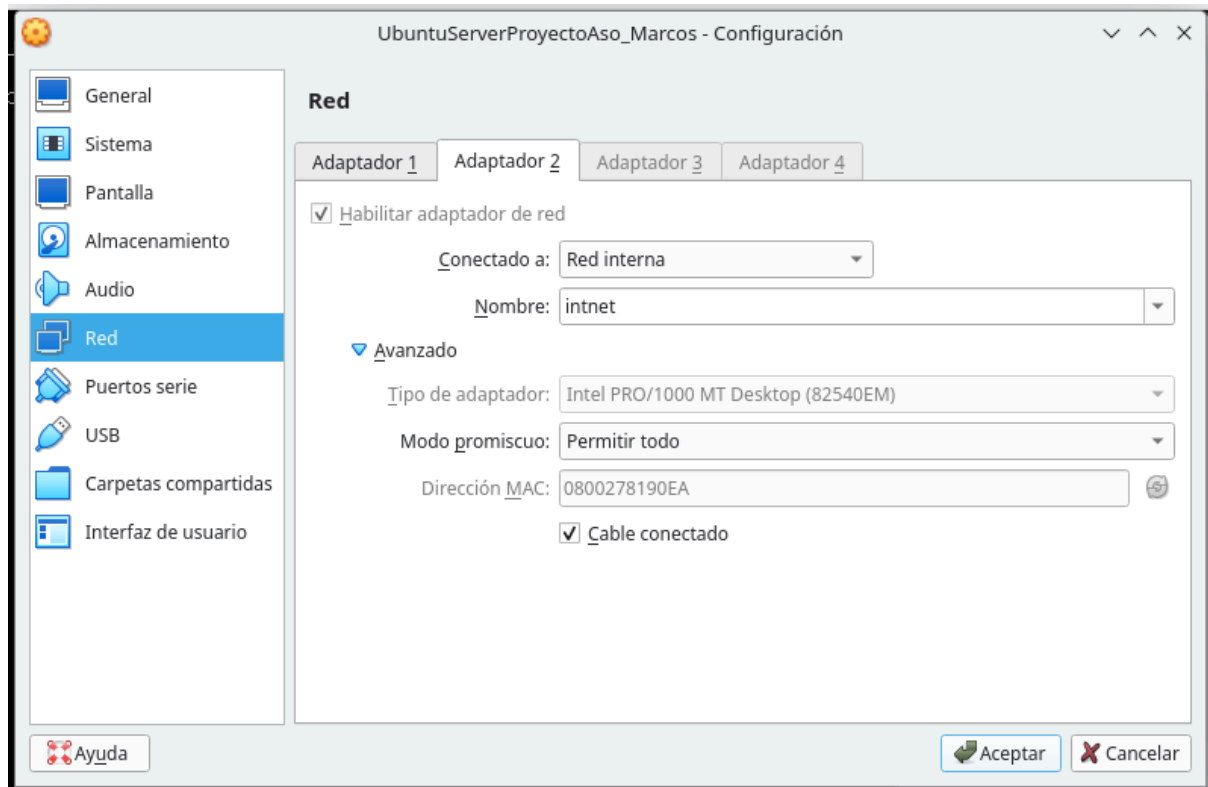


Imagen ilustrativa añadiendo el segundo adaptador

Por lo tanto, procedo a simplemente ejecutar mi script en un Ubuntu server nuevo para que nos configure tanto “Netplan”, como el “DHCP”, como el “DNS” y las iptables. Todo este código, está disponible en el github y ahí se pueden ver todas las reglas y pasos que realiza el script:

```

399     echo -e "$Samarillo 14. Procedemos a crear y configurar las iptables del servidor en el directorio '/usr/bin/set_iptables.sh'
400     sleep 2
401     sudo touch /usr/bin/set_iptables.sh
402     sudo chmod 777 /usr/bin/set_iptables.sh
403     sudo echo "iptables -F
404     iptables -t nat -F
405     iptables -t nat -A POSTROUTING -o enp0s3 -s 192.168.0.0/24 -j MASQUERADE
406     iptables -A FORWARD -i enp0s8 -o enp0s3 -s 192.168.0.0/24 -j ACCEPT
407     iptables -A FORWARD -i enp0s3 -o enp0s8 -m state --state RELATED,ESTABLISHED -j ACCEPT
408     " > /usr/bin/set_iptables.sh
409
410     sudo /usr/bin/set_iptables.sh
411
412     echo -e "$Verde Y por ultimo hacemos restart del servidor DNS y DHCP para que funcionen correctamente. $nc"
413     sudo service named restart
414     sudo service isc-dhcp-server restart
415
416     echo
417     echo -e "$Verde Todo correcto $nc"
418     echo
419     echo -e "$Verde Finalizando script... $nc"

```

Captura de la parte final del script (Montage)

Para realizar la instalación de los servicios comentados anteriormente, deberemos descargar el fichero .sh con el comando “wget” de “010\_Montaje-UbuntuServer” en el repositorio de github. El comando exacto, quedaría tal que así:

“wget

[https://raw.githubusercontent.com/Malonsocabral/Proyecto\\_ASO\\_Marcos/main/010\\_Montage-UbuntuServer.sh](https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/010_Montage-UbuntuServer.sh)”

```
marcos@pfmarcos:~$ wget https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/Montage_UbuntuServer.sh
--2025-03-14 19:34:47-- https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/Montage_UbuntuServer.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14375 (14K) [text/plain]
Saving to: 'Montage_UbuntuServer.sh'

Montage_UbuntuServer.sh 100%[=====] 14,04K --.-KB/s in 0,002s

2025-03-14 19:34:48 (9,04 MB/s) - 'Montage_UbuntuServer.sh' saved [14375/14375]

marcos@pfmarcos:~$ sudo chmod 777 Montage_UbuntuServer.sh
marcos@pfmarcos:~$ sudo ./Montage_UbuntuServer.sh
Por favor, introduce el nombre del dominio (por ejemplo, ola.host):
marcos.proyecto_
```

Imagen ilustrativa

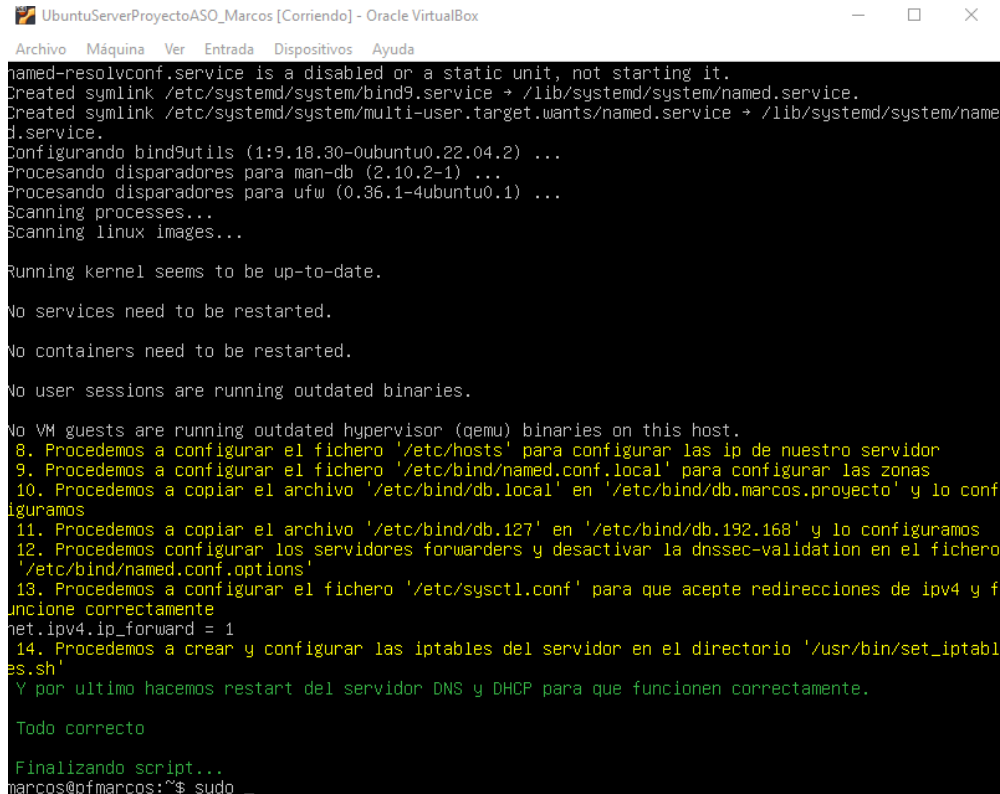
Una vez descargado el fichero .sh debemos darle permisos de ejecución (chmod -x 010\_Montage-UbuntuServer.sh) y luego poner el siguiente comando para ejecutar el script (sudo ./010\_Montage-UbuntuServer.sh):

```
marcos@pfmarcos:~$ sudo ./Montage_UbuntuServer.sh
Por favor, introduce el nombre del dominio (por ejemplo, ola.host):
marcos.proyecto
Para obtener mas informacion pon '--help' o '-h' en el primer parametro
1. Comenzamos con las actualizaciones generales con un sudo apt update
Obj:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Des:5 http://es.archive.ubuntu.com/ubuntu jammy/main Translation-es [332 kB]
Des:6 http://es.archive.ubuntu.com/ubuntu jammy/restricted Translation-es [964 B]
Des:7 http://es.archive.ubuntu.com/ubuntu jammy/universe Translation-es [1.356 kB]
Des:8 http://es.archive.ubuntu.com/ubuntu jammy/multiverse Translation-es [68,2 kB]
Descargados 1.758 kB en 2s (714 kB/s)
Leyendo lista de paquetes... 5%
```

Imagen ilustrativa



Como se puede observar en la anterior captura, antes de ejecutar nada, el script nos pide un nombre de dominio para así cambiar los parámetros correspondientes. Una vez escrito el nombre, el script ya no solicitará nada y realizará todas las tareas correctamente como se puede ver a continuación:



```

UbuntuServerProyectoASO_Marcos [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
named-resolvconf.service is a disabled or a static unit, not starting it.
Created symlink /etc/systemd/system/bind9.service → /lib/systemd/system/named.service.
Created symlink /etc/systemd/system/multi-user.target.wants/named.service → /lib/systemd/system/named.service.
Configurando bind9utils (1:9.18.30-0ubuntu0.22.04.2) ...
Procesando disparadores para man-db (2.10.2-1) ...
Procesando disparadores para ufw (0.36.1-4ubuntu0.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
8. Procedemos a configurar el fichero '/etc/hosts' para configurar las ip de nuestro servidor
9. Procedemos a configurar el fichero '/etc/bind/named.conf.local' para configurar las zonas
10. Procedemos a copiar el archivo '/etc/bind/db.local' en '/etc/bind/db.marcos.proyecto' y lo configuramos
11. Procedemos a copiar el archivo '/etc/bind/db.127' en '/etc/bind/db.192.168' y lo configuramos
12. Procedemos configurar los servidores forwarders y desactivar la dnssec-validation en el fichero '/etc/bind/named.conf.options'
13. Procedemos a configurar el fichero '/etc/sysctl.conf' para que acepte redirecciones de ipv4 y funcione correctamente
net.ipv4.ip_forward = 1
14. Procedemos a crear y configurar las iptables del servidor en el directorio '/usr/bin/set_iptables.sh'
Y por ultimo hacemos restart del servidor DNS y DHCP para que funcionen correctamente.

Todo correcto

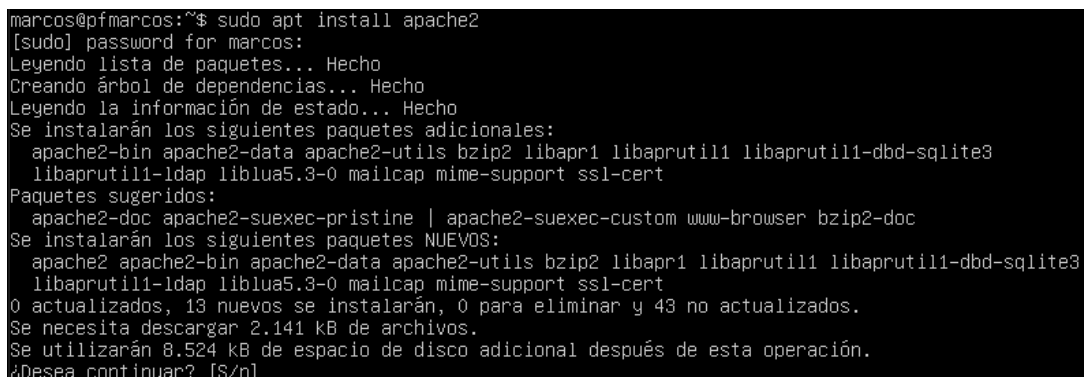
Finalizando script...
marcos@pfmarcos:~$ sudo

```

Imagen ilustrativa

### 3. Instalación de Apache2

Una vez que el DHCP, DNS y iptables estén correctamente configuradas, es momento de instalar el servidor de “Apache2”. Esto lo realizaremos con el comando “sudo apt install apache2”:



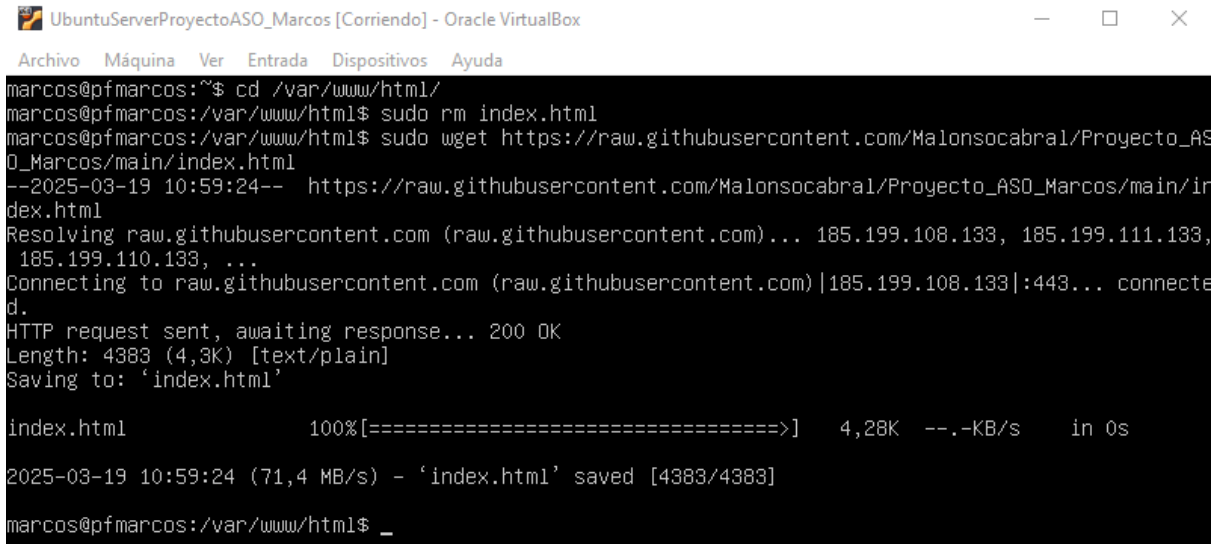
```

marcos@pfmarcos:~$ sudo apt install apache2
[sudo] password for marcos:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
 libaprutil1-ldap liblua5.3-0 mailcap mime-support ssl-cert
Paquetes sugeridos:
 apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser bzip2-doc
Se instalarán los siguientes paquetes NUEVOS:
 apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
 libaprutil1-ldap liblua5.3-0 mailcap mime-support ssl-cert
0 actualizados, 13 nuevos se instalarán, 0 para eliminar y 43 no actualizados.
Se necesita descargar 2.141 kB de archivos.
Se utilizarán 8.524 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]

```

Imagen ilustrativa

Para cambiar la página por defecto que nos aparece, primero debemos movernos con “cd /var/www/html”, para movernos al directorio donde se almacena esta volvemos a descargar con wget el index.html de mi repositorio (aunque esta parte es más opcional ya que no influye en nada):



```

UbuntuServerProyectoASO_Marcos [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
marcos@pfmarcos:~$ cd /var/www/html/
marcos@pfmarcos:/var/www/html$ sudo rm index.html
marcos@pfmarcos:/var/www/html$ sudo wget https://raw.githubusercontent.com/Malonsocabral/Proyecto_AS0_Marcos/main/index.html
--2025-03-19 10:59:24-- https://raw.githubusercontent.com/Malonsocabral/Proyecto_AS0_Marcos/main/index.html
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4383 (4.3K) [text/plain]
Saving to: 'index.html'

index.html          100%[=====] 4,28K  --.-KB/s  in 0s

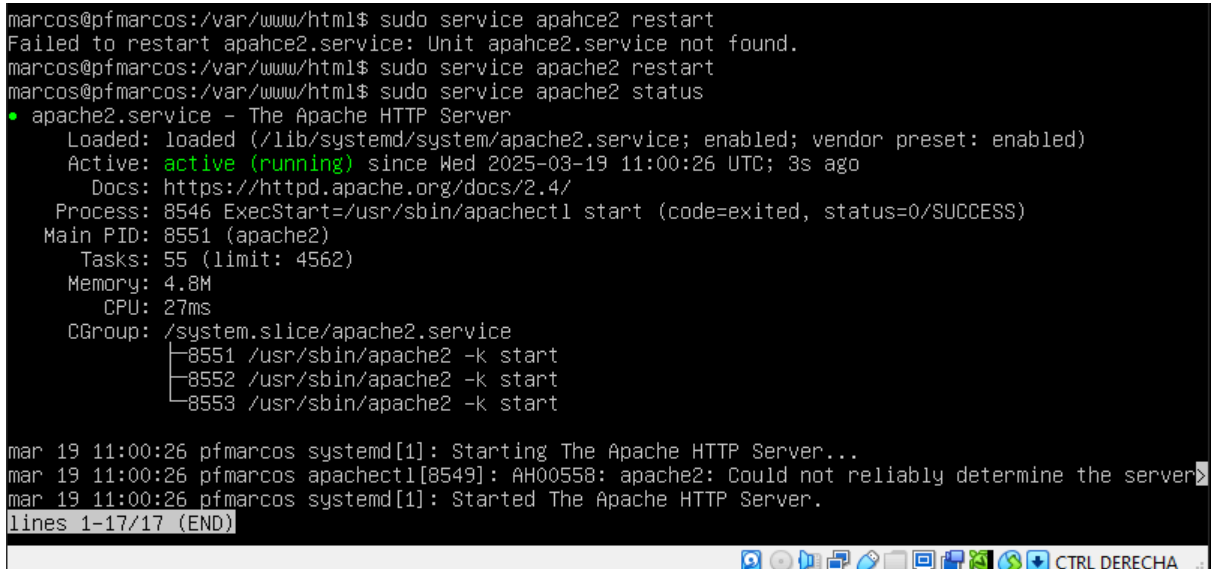
2025-03-19 10:59:24 (71.4 MB/s) - 'index.html' saved [4383/4383]

marcos@pfmarcos:/var/www/html$ _

```

Imagen ilustrativa

Y por último para comprobar que está todo correcto, hacemos un “sudo service apache2 restart” y “sudo service apache2 status” para comprobar su estado.



```

marcos@pfmarcos:/var/www/html$ sudo service apahce2 restart
Failed to restart apahce2.service: Unit apahce2.service not found.
marcos@pfmarcos:/var/www/html$ sudo service apache2 restart
marcos@pfmarcos:/var/www/html$ sudo service apache2 status
• apache2.service - The Apache HTTP Server
  Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2025-03-19 11:00:26 UTC; 3s ago
    Docs: https://httpd.apache.org/docs/2.4/
  Process: 8546 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 8551 (apache2)
   Tasks: 55 (limit: 4562)
  Memory: 4.8M
    CPU: 27ms
  CGroup: /system.slice/apache2.service
          └─8551 /usr/sbin/apache2 -k start
            └─8552 /usr/sbin/apache2 -k start
              └─8553 /usr/sbin/apache2 -k start

mar 19 11:00:26 pfmarcos systemd[1]: Starting The Apache HTTP Server...
mar 19 11:00:26 pfmarcos apachectl[8549]: AH00558: apache2: Could not reliably determine the server's
mar 19 11:00:26 pfmarcos systemd[1]: Started The Apache HTTP Server.
lines 1-17/17 (END)

```

Imagen ilustrativa

#### 4. Montaje de la función Email

Para este punto debemos simplemente descargarnos el script 020 con wget y le damos permisos de 777 con chmod, como se puede ver a continuación:

“wget

https://github.com/Malonsocabral/Proyecto\_ASO\_Marcos/blob/main/020\_mail-conf.sh”

```
marcos@pfmarcos:~$ wget https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/020_mail-conf.sh
--2025-03-21 18:48:10-- https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/020_mail-conf.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1516 (1,5K) [text/plain]
Saving to: '020_mail-conf.sh'

020_mail-conf.sh      100%[=====] 1,48K  --.-KB/s  in 0s

2025-03-21 18:48:10 (18,4 MB/s) - '020_mail-conf.sh' saved [1516/1516]

marcos@pfmarcos:~$ sudo chmod 777 020_mail-conf.sh
marcos@pfmarcos:~$
```

Imagen ilustrativa

Luego para ejecutar el comando simplemente hacemos un “sudo ./020\_mail-conf.sh” y nos pedirá un correo y contraseña. Como se puede ver a continuación:

```
UbuntuServerProyectoASO_Marcos (Instantánea 2) [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
marcos@pfmarcos:~$ sudo ./020_mail-conf.sh
Introduce tu correo electrónico (ejemplo@gmail.com): marcos.cab2005@gmail.com
Introduce tu contraseña de aplicación (o contraseña de Gmail):
```

Imagen ilustrativa

Cabe destacar que para este paso de contraseñas, en principio, no hace falta poner nada, pero si da fallo o no envía ningún correo, haría falta hacer lo siguiente. Primero debemos entrar en el siguiente link, que nos lleva a nuestro apartado de google donde administramos nuestras contraseñas para aplicaciones como se puede ver en la siguiente imagen(<https://myaccount.google.com/apppasswords>):

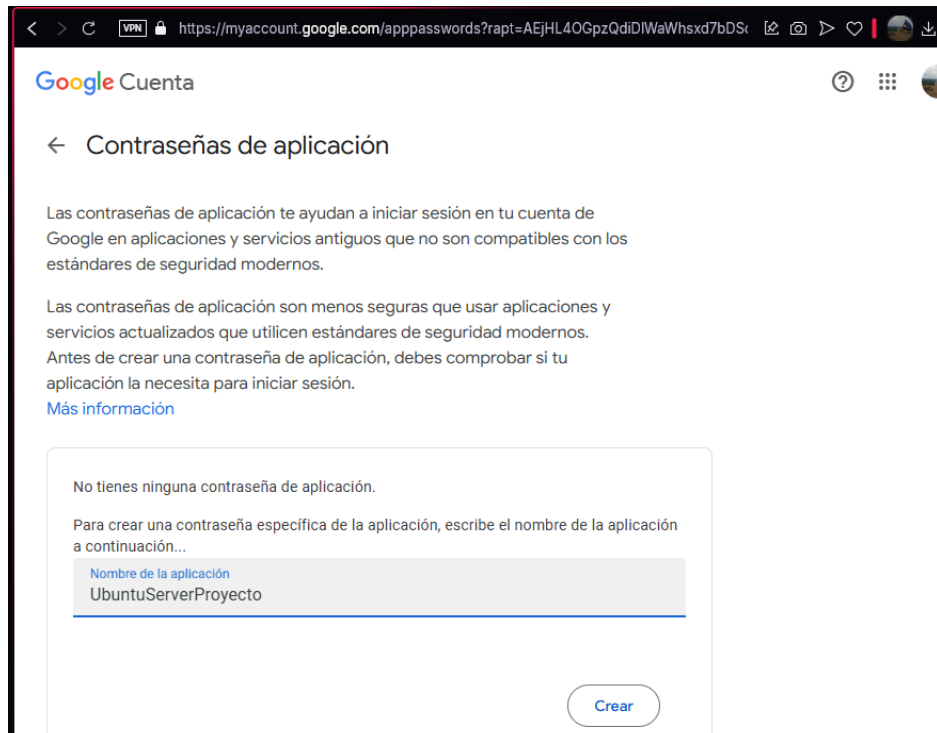


Imagen ilustrativa

Una vez ponemos el nombre y las credenciales de nuestra cuenta de gmail, nos saltara una contraseña generada que es la que debemos poner sin espacios en este campo que requiere mi script:

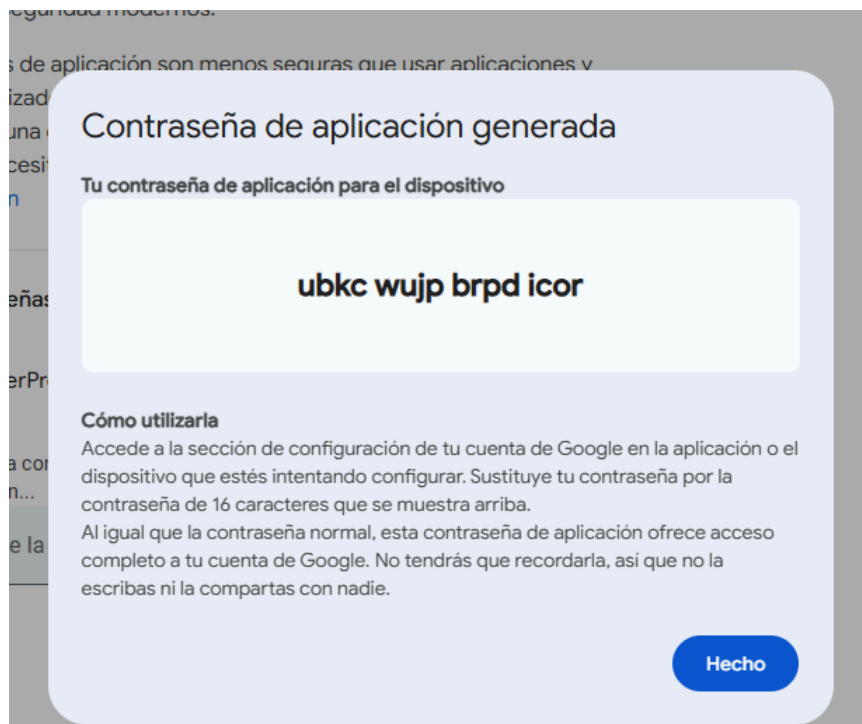


Imagen ilustrativa

Una vez hecho esto, se nos empezara a descargar el paquete de mailutils para que podamos gestionar estos correos:

```
marcos@pfmarcos:~$ sudo ./configure.sh
Introduce tu correo electrónico (ejemplo@gmail.com): marcos.cab2005@gmail.com
Introduce tu contraseña de aplicación (o contraseña de Gmail):
Instalando mailutils y ssmtp...
Obj:1 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Des:2 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Obj:3 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Des:4 http://es.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2.387 kB]
Des:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Descargados 2.644 kB en 11s (243 kB/s)
Leyendo lista de paquetes... Hecho
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... 50%
```

Imagen ilustrativa

```
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 44 no actualizados.
Configurando ssmtp...
Configurando revaliases...
Enviando correo de prueba a marcos.cab2005@gmail.com...
Correo de prueba enviado correctamente a marcos.cab2005@gmail.com.
Configuración completada. Ahora puedes enviar correos desde este servidor.
marcos@pfmarcos:~$
```

Imagen ilustrativa

Posteriormente nos mandará un correo de prueba como se puede ver a continuación:

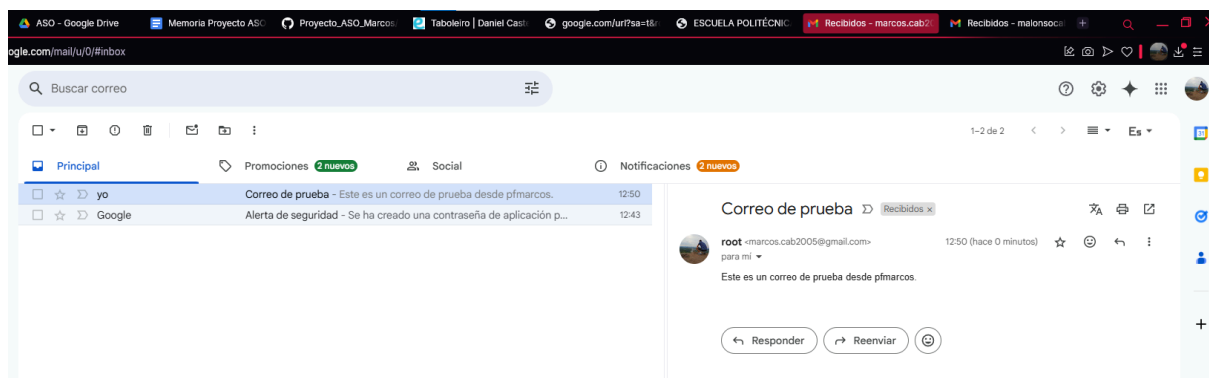


Imagen ilustrativa

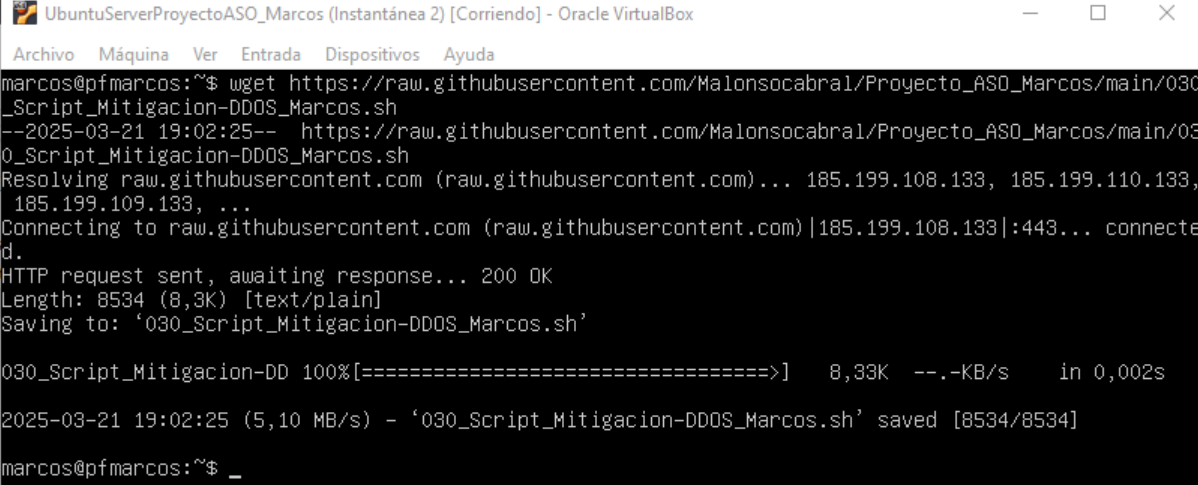
Una vez hecho esto, ya estaremos preparados para ejecutar ejecutar el script de mitigación de ataques DDOS.

## 5. Montaje del script de mitigación

Para continuar, descargamos el script de prevención de Ddos el cual identifica las posibles ips y rango maliciosas como comente en el apartado 1. Por lo que procedemos a descargarlo de mi repositorio de github con el comando:

“wget

[https://raw.githubusercontent.com/Malonsocabral/Proyecto\\_ASO\\_Marcos/main/030\\_Script\\_Mitigacion-DDOS\\_Marcos.sh](https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/030_Script_Mitigacion-DDOS_Marcos.sh)”



```

UbuntuServerProyectoASO_Marcos (Instantánea 2) [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
marcos@pfmarcos:~$ wget https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/030_Script_Mitigacion-DDOS_Marcos.sh
--2025-03-21 19:02:25-- https://raw.githubusercontent.com/Malonsocabral/Proyecto_ASO_Marcos/main/030_Script_Mitigacion-DDOS_Marcos.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.110.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8534 (8.3K) [text/plain]
Saving to: '030_Script_Mitigacion-DDOS_Marcos.sh'

030_Script_Mitigacion-DD 100%[=====>] 8,33K --.-KB/s in 0,002s

2025-03-21 19:02:25 (5,10 MB/s) - '030_Script_Mitigacion-DDOS_Marcos.sh' saved [8534/8534]

marcos@pfmarcos:~$ _

```

Imagen ilustrativa

Una vez instalado , debemos darle permisos de ejecución , y yo en mi caso prefiero darle todos los permisos con “sudo chmod 777 ./030\_Script\_Mitigacion-DDOS\_Marcos.sh”.

Cabe destacar que no ejecutaremos el script ahora sino que esperaremos a la simulación para observar los cambios en el ataque con script y sin script.

## **6.Montaje kali (Instalación herramientas)**

Para comenzar con la instalación de kali, vamos a la página oficial y descargamos el .vdi y vbox para que nos sea más sencillo en <https://www.kali.org/get-kali/#kali-virtual-machines>.

Luego movemos este vdi y .vbox a nuestra carpeta de VirtualBox dentro de la carpeta de la máquina que vamos a crear (en mi caso “Kali\_ProyectoASO”) :

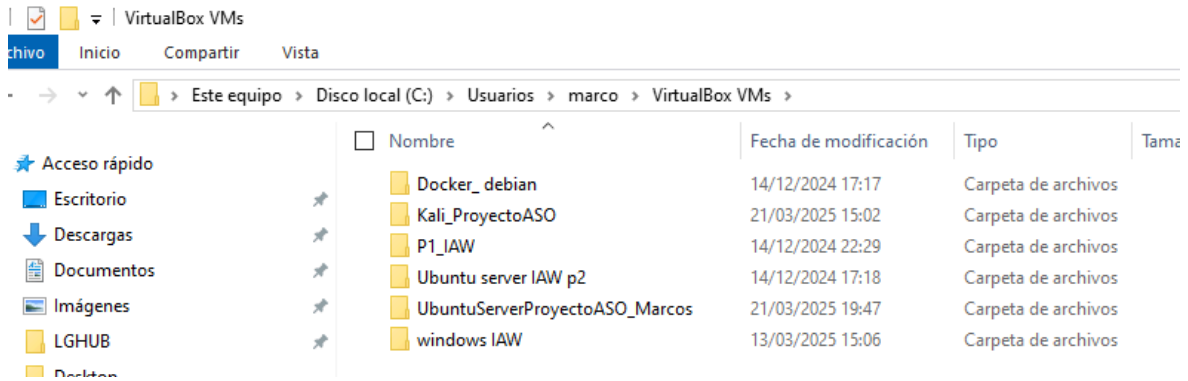


Imagen ilustrativa

Luego en Virtual Box le damos a “Herramientas”, luego “Añadir”, y buscamos en la carpeta donde movimos el .vbox y .vdi el .vbox de la máquina de kali.

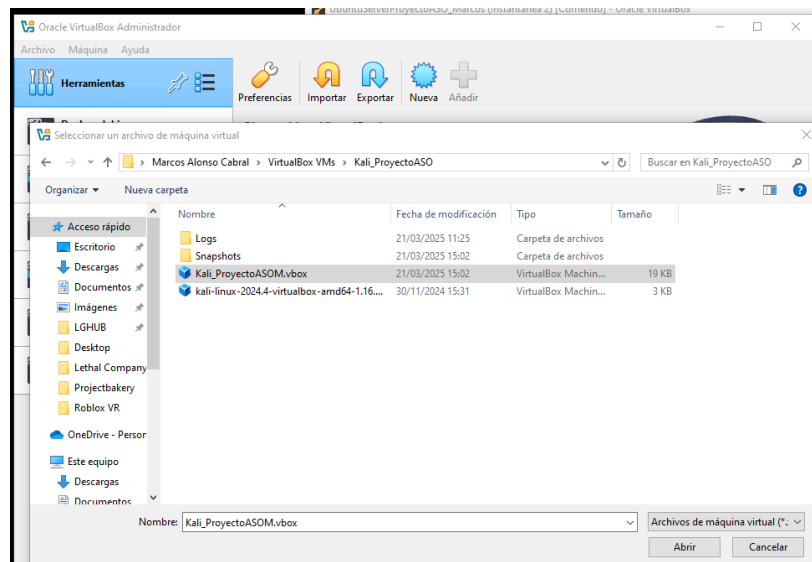


Imagen ilustrativa

Una vez instalado kali, necesitamos instalar dos herramientas, una será la de “Apache Benchmark” y luego la de “Hping3” aunque cabe destacar que me he informado sobre las herramientas, y hay infinidad de estas. Para ello, pondremos los siguientes comandos:

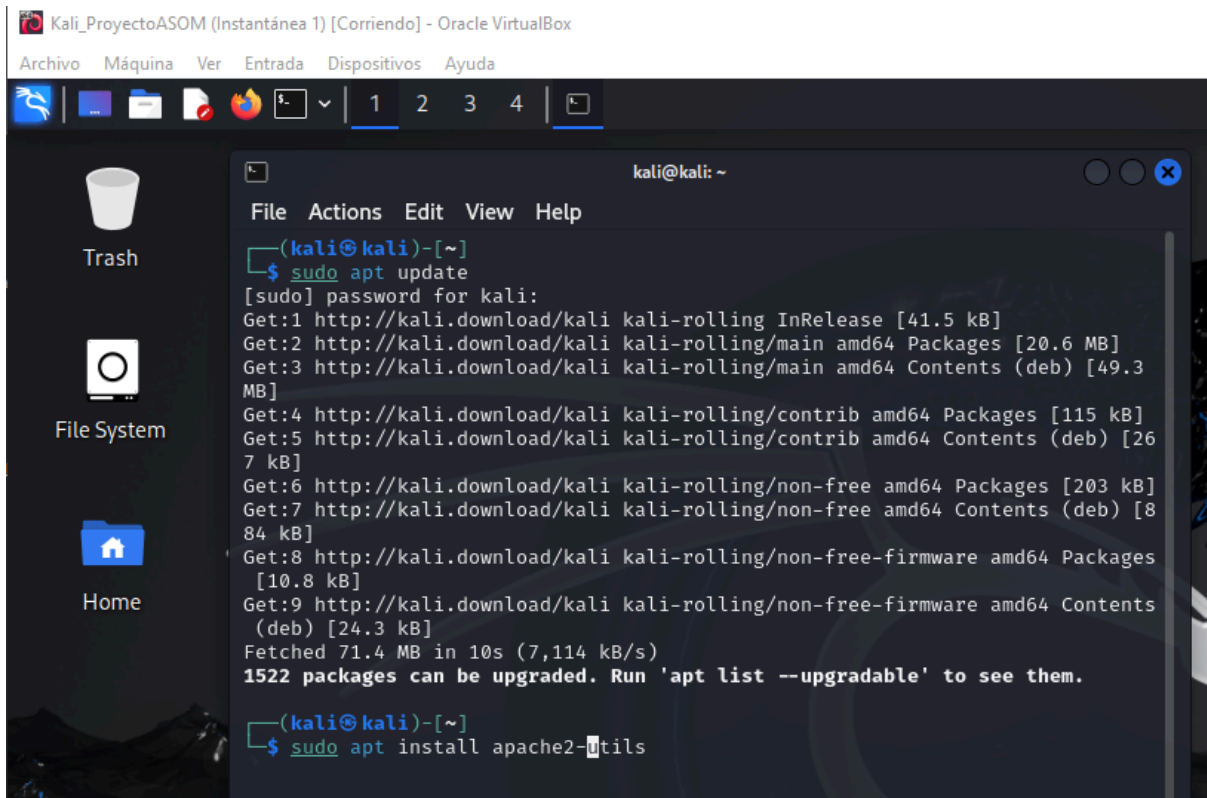


Imagen ilustrativa

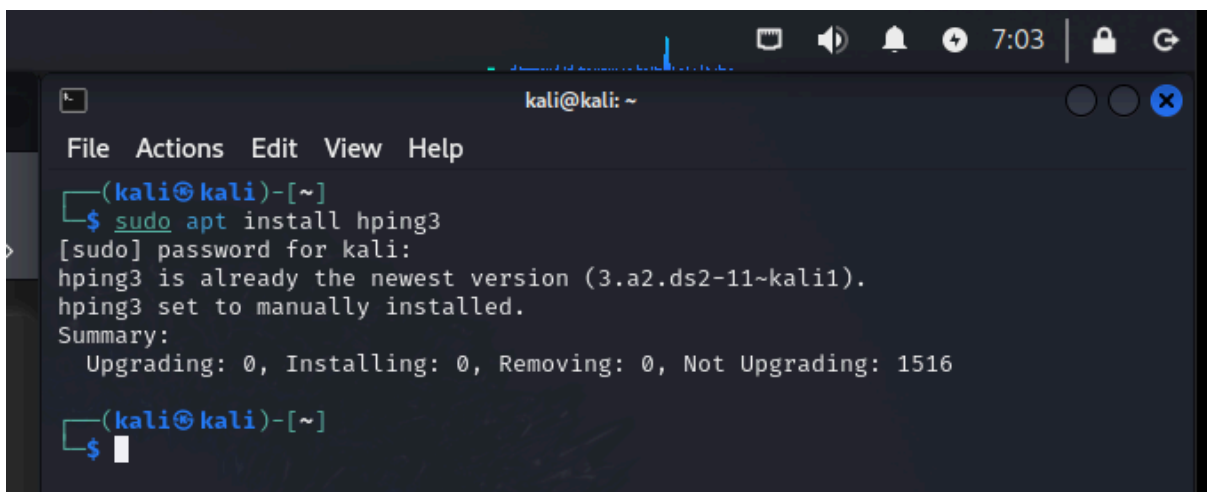


Imagen ilustrativa

Una cosa a tener en cuenta es que después de instalar estos paquetes, debemos meternos en red interna (en la máquina kali) para conectarnos con el servidor. Para esto vamos al apartado de “Red” en “Virtual Box” y le damos a “red interna” y “permitir todo”:



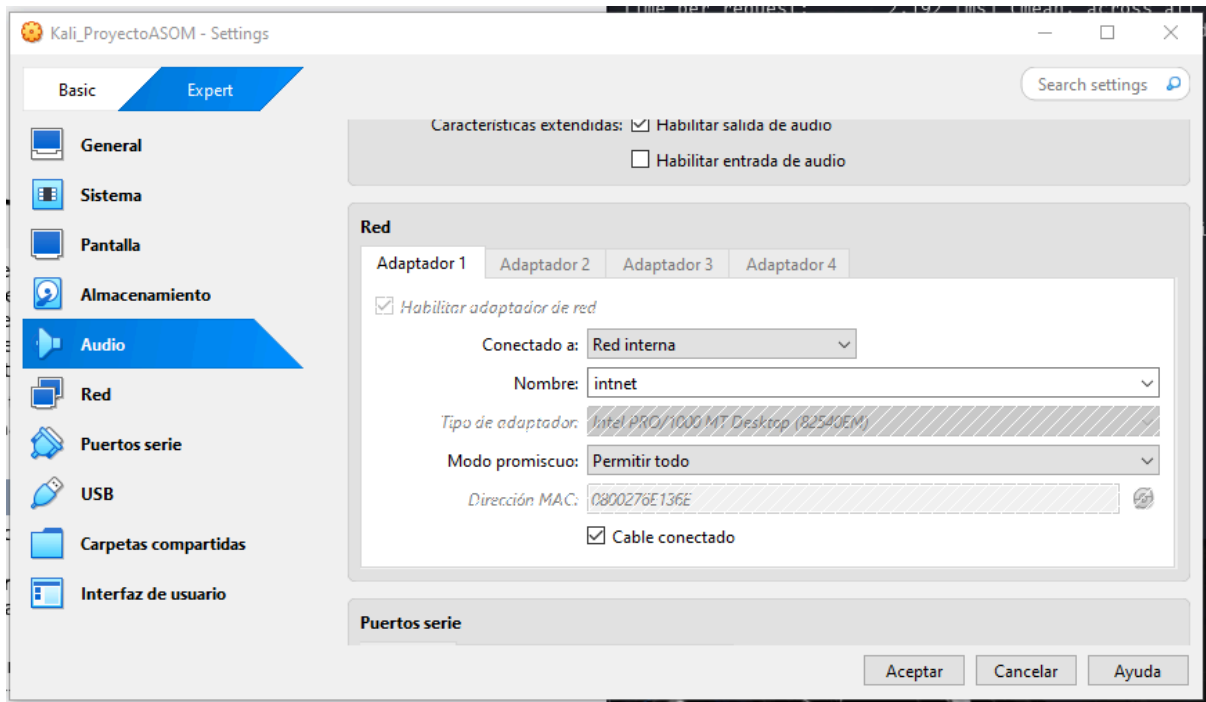


Imagen ilustrativa

Una vez ya configurado esto, podemos probar a acceder a la página web de apache poniendo en el navegador “<http://192.168.0.1/>” ya que 192.168.0.1 es la ip del servidor:

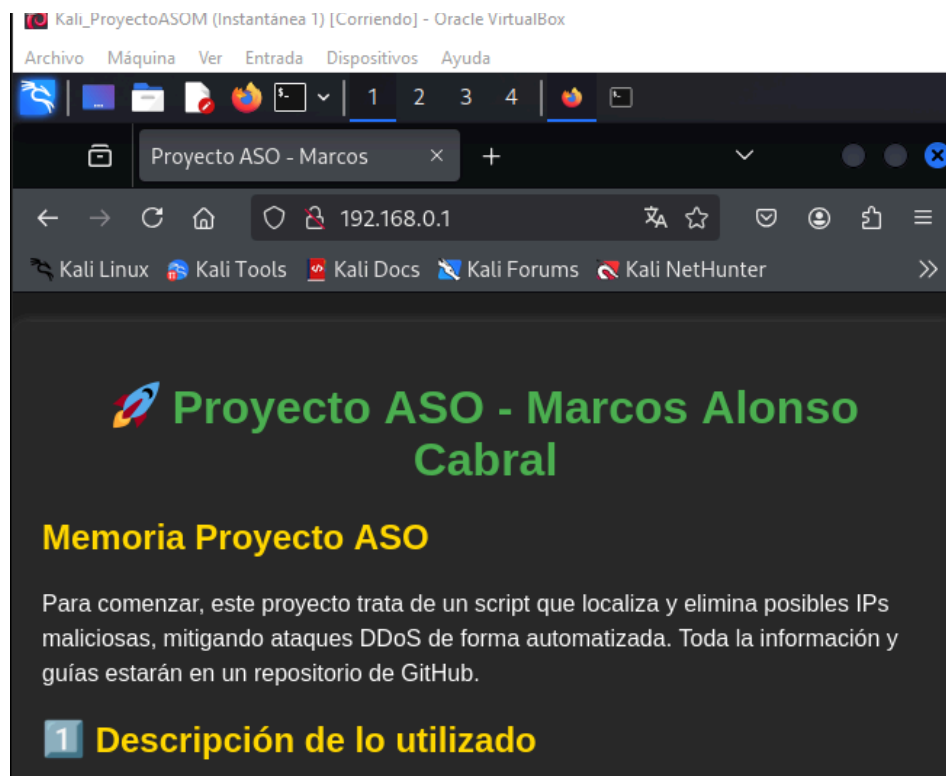


Imagen ilustrativa

## **7.Simulación de situación real**

En este apartado número 7 realizar, primero un ataque de Ddos con peticiones de http con Apache Benchmarks y luego con la herramienta antes instalada, llamada Hping3, primero realizaremos unas comprobaciones sin el script instalado y quitando todas las reglas de iptables (sudo iptables -F) para comprobar si el servidor se cae o baja su tiempo de respuesta. Y posteriormente ya ejecutaremos todas las configuraciones de iptables anteriores y el script para mitigar estos ataques

Primero, para mostrar de una manera más visual las conexiones al servidor por el puerto 80, instalaremos net-tools como se puede ver a continuación:

```
marcos@pfmarcos:~$ netstat -ant | grep :80
Command 'netstat' not found, but can be installed with:
sudo apt install net-tools
marcos@pfmarcos:~$ sudo apt install net-tools
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  net-tools
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 43 no actualizados.
Se necesita descargar 204 kB de archivos.
Se utilizarán 819 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64 1.6
ubuntu:5 [594 kB]
```

Imagen ilustrativa

Una vez instalado, ejecutamos en kali el ataque con el comando “ab -n 5000 -c 500 http://192.168.0.1”. Este script significa que con “-n 5000” envía 1000 solicitudes y con “-c 500” hace que se mantengan 500 solicitudes simultáneas.

Y en el servidor ejecutamos “netstat -ant | grep :80” (para así filtrar por el puerto 80 que es el de apache). Y como se puede ver en la siguiente imagen, el servidor está cargando con estas peticiones:

The image shows two terminal windows. The left window is a Kali Linux terminal running the Apache Benchmark (ab) command: `ab -n 50000 -c 500 http://192.168.0.1/`. The output shows the benchmarking process, including the number of requests completed (30125) and the time taken (53.155 seconds). The right window is a terminal window titled 'UbuntuServerProyectoASO\_Marcos [Corriendo] - Oracle VirtualBox' showing the output of the `netstat -ant | grep :80_` command, displaying a list of active connections to port 80.

Imagen ilustrativa sin script

A continuación realizaremos el comando “time curl -s -o /dev/null http://192.168.0.1/”. Este comando significa que “time” mide el tiempo total de ejecución y “curl -s -o /dev/null” hace una petición silenciosa sin mostrar el contenido. Cabe destacar que este comando lo ejecutaremos en otra terminal junto con el Apache Benchmark como se puede observar a continuación:

The image shows two terminal windows. The left window is a Kali Linux terminal running the Apache Benchmark (ab) command: `ab -n 10000 -c 1000 http://192.168.0.1/`. The output shows the benchmarking process, including the number of requests completed (10000) and the time taken (26.976 seconds). The right window is a terminal window titled 'Kali\_ProyectoASOM (Instantánea 2) [Corriendo] - Oracle VirtualBox' showing the output of the `time curl -s -o /dev/null http://192.168.0.1/` command, displaying the execution time (0.94s) and system resources (user, sys, cpu).

Imagen ilustrativa

Luego para comprobar la mejoría y que el script funcione correctamente, ejecutamos el script que descargamos anteriormente de mitigación, con un comando como el siguiente “sudo ./030\_Script\_Mitigacion-DDOS\_Marcos.sh” aunque depende de donde hayamos guardado el script en el paso anterior, deberemos poner una ruta o otra. A continuación muestro como se ejecuta y bloquea la ip del kali que sigue haciendo peticiones al servidor:

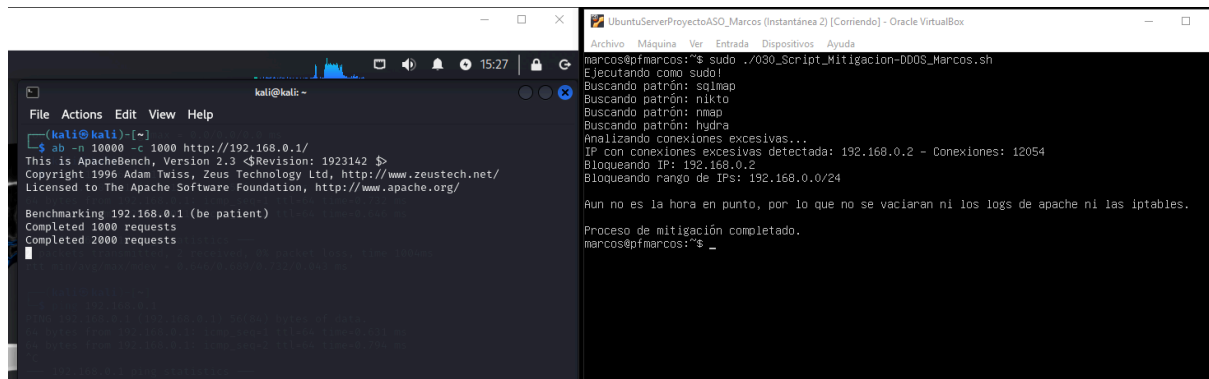


Imagen ilustrativa

Posteriormente volvemos a ejecutar en el kali el comando de “time curl -s -o /dev/null http://192.168.0.1/” no funciona ya que esta bloqueado y me veo obligado a poner este comando en el servidor y ver que el tiempo de respuesta ha disminuido de manera considerable (de 1 segundo a 0.01):

```

marcos@pfmarcos:~$ time curl -s -o /dev/null http://192.168.0.1/
real    0m0.008s
user    0m0.007s
sys     0m0.000s
  
```

Imagen ilustrativa

A continuación repetimos el mismo proceso pero atacando con la herramienta “hping3”. Por lo que primero dejamos deshabilitado el script (aun no le hemos metido el crontab para que no se nos ejecute todo el rato mientras hacemos la pruebas) y segundo, con el comando “sudo hping3 -S -p 80 --flood --rand-source 192.168.0.1” que manda multitud de peticiones http al servidor. Podemos comprobar con el comando “time curl -s -o /dev/null http://192.168.0.1/” el tiempo de respuesta sin el script:

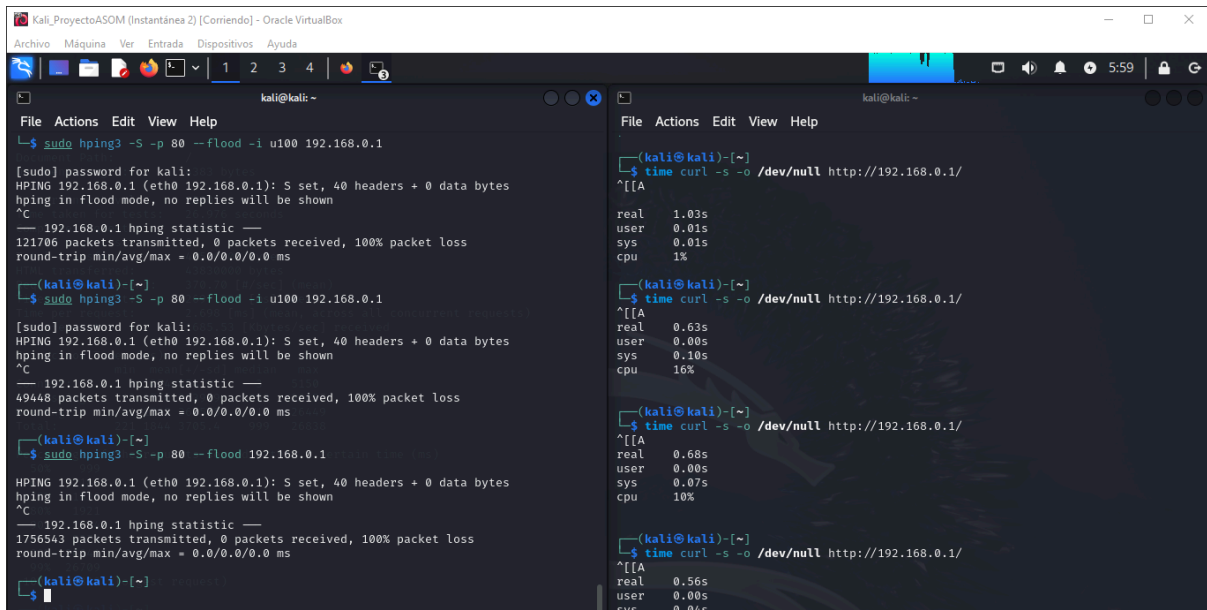


Imagen ilustrativa

Como podemos observar en la anterior imagen, la herramienta hping no es tan potente como la de Apache Benchmark, aunque cabe destacar que podemos abrir varias terminales con el mismo comando, si que se ralentiza mucho más incluso llegando a los 6 segundos en ocasiones:

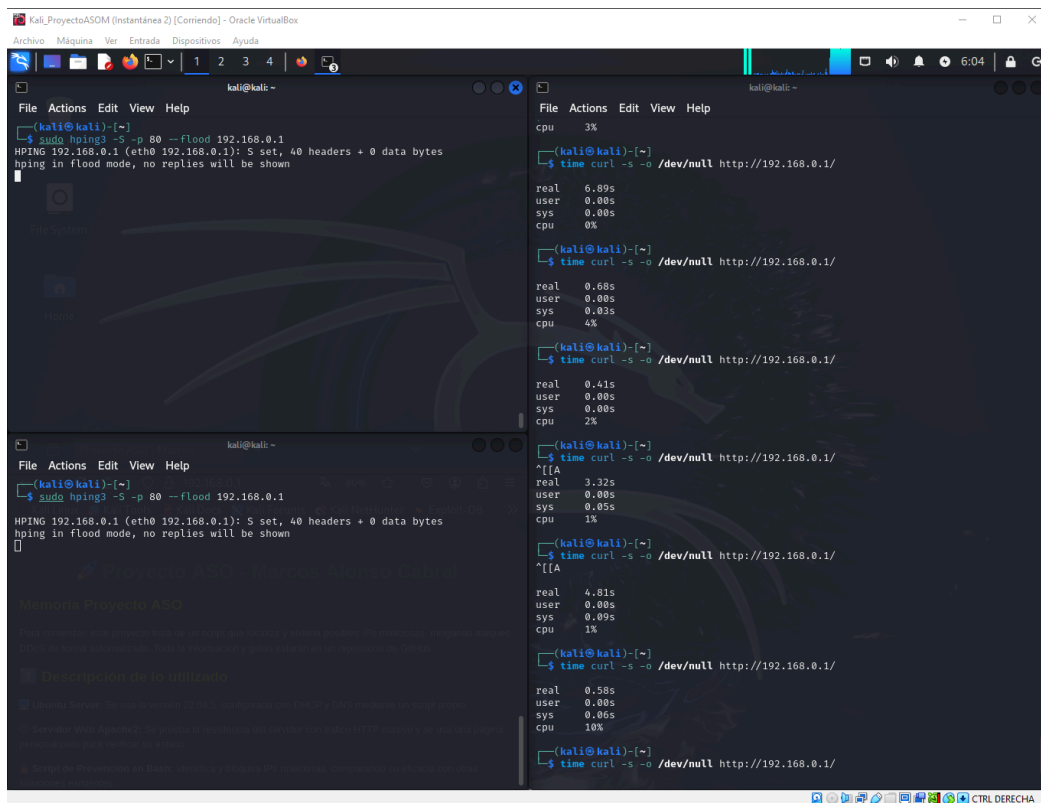


Imagen ilustrativa con dos terminales

Y ya a continuación ejecutamos el script y ponemos otra vez el comando time y el netstat para ver si sigue haciendo peticiones al servidor:

```

kali@kali:~$ sudo hping3 -S -p 80 --flood --rand-source 192.168.0.1
HPING 192.168.0.1 (eth0 192.168.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

marcos@pfmarcos:~$ sudo ./030_Script_Mitigacion-DDOS_Marcos.sh
Ejecutando como sudo!
Buscando patrón: sqlmap
Buscando patrón: nikto
Buscando patrón: nmap
Buscando patrón: hydra
Analizando conexiones excesivas...
./030_Script_Mitigacion-DDOS_Marcos.sh: line 124: warning: command substitution: ignored null byte in input
IP con conexiones excesivas detectada: 192.168.0.2 - Conexiones: 12669
Bloqueando IP: 192.168.0.2
Como el rango de ips, es el 192.168.0.0/24 y engloba a este propio servidor, no sera bloqueado
Aun no es la hora en punto, por lo que no se vaciaran ni los logs de apache ni las iptables.

Proceso de mitigación completado.
marcos@pfmarcos:~$ netstat -ant | grep :80
tcp6      0      0 :::80          :::*           LISTEN
marcos@pfmarcos:~$ time curl -s -o /dev/null http://192.168.0.1/
real    0m0.008s
user    0m0.007s
sys     0m0.000s
marcos@pfmarcos:~$

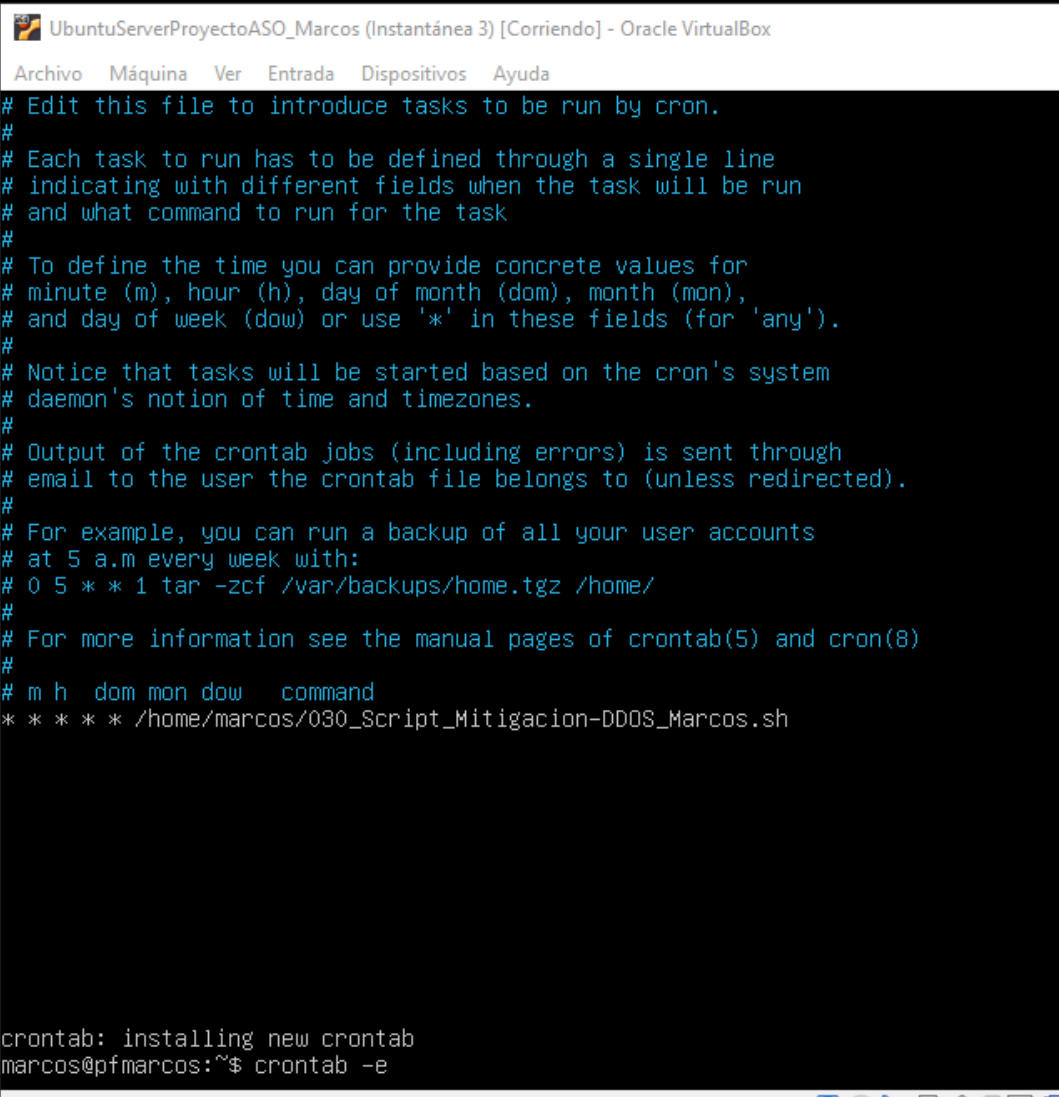
```

Imagen ilustrativa

Como podemos ver al lanzar este comando hping3 con rand-source (hace que parezcan ips de fuentes aleatorias) el servidor bloquea todas las peticiones y nos deja con el tiempo de respuesta normal (menos de un segundo).

## 8. Instalando crontab

Una vez comprobado que el script funciona correctamente, debemos poner el crontab para que se ejecute cada minuto y así poder parar posibles ataques de DDOS. Por lo tanto, debemos poner el comando “sudo crontab -e” y poner al final del fichero la instrucción “\* \* \* \* \* /home/marcos/030\_Script\_Mitigacion-DDOS\_Marcos.sh” :



```

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
* * * * * /home/marcos/030_Script_Mitigacion-DDOS-Marcos.sh

crontab: installing new crontab
marcos@pfmarcos:~$ crontab -e

```

Imagen ilustrativa

Cabe destacar que si queremos parar estas instrucciones para realizar cualquier prueba tranquilos (como me paso a mi) debemos poner “sudo crontab -f” para eliminar todos los crontabs activos en ese momento.

## **9. Comprobando los correos**

Una vez ejecutado todo, podemos ver si se mandaron los correos correctamente, por lo que simplemente accedemos a nuestro correo configurado y podremos ver emails como estos:



Imagen ilustrativa

Y cuando pasa una hora en punto, se mandará un mensaje como el siguiente:



Imagen ilustrativa

## **10. Comparación con aplicaciones externas.**

Para comenzar con esta comparación de mi script y otras aplicaciones para mitigar ataques de DDOS, hay que decir que mi script es una solución simple a un problema o ataque que se puede realizar de muchas maneras y es complejo de mitigar, por lo que las aplicaciones externas, en las que se ha invertido mucho más tiempo y conocimiento que en mi script, son claramente superiores, pero aun asi, comencemos con la comparación.



## **Características de mi script**

A continuación iré enumerando algunas de las características principales que realiza mi script (aunque ya se hayan dicho en la introducción):

### **1. Tiene detección basada en Umbrales:**

Esto quiere decir que bloquea IPs que superan un número específico de conexiones (ej: 100). Y analiza logs de Apache (access.log) para identificar conexiones sospechosas.

### **2. Bloquea IPs, Rangos y IDS simple :**

Mi script usa iptables para bloquear IPs individuales y rangos /24 (primeros 3 octetos). Y para ello registra IPs bloqueadas en un archivo (ips\_bloqueadas.log) para evitar duplicados. También bloquea algunas cadenas maliciosas con un IDS simple.

### **3. Pequeño sistema de detección de Patrones Maliciosos:**

Tras investigar sobre los ataques de Ddos, he visto que los ataques a grandes servidores se suelen hacer desde una botnet o estructuras similares que dificultan su mitigación por lo que intente implementar un pequeño código que busca herramientas comunes de ataque en los logs (ej: sqlmap, nmap, hydra).Y bloquea instantáneamente las IPs asociadas a estos patrones.

### **4. Notificaciones por Correo:**

Es capaz de envía alertas al bloquear una IP o rango a mi correo personal que es [malonsocabral@danielcastelao.org](mailto:malonsocabral@danielcastelao.org).

### **5. Gran simplicidad:**

Ya que está escrito en Bash, que es fácil de modificar y entender. Y no requiere ningún tipo de instalación compleja.

### **6. Características a mayores:**

Como puede ser el sistema de limpiar logs y desbloquear las ips cada hora.

## **Características de Aplicaciones Similares**

Pasando con las características de las aplicaciones similares, iré comentando a grandes rasgos las principales características pero sin pararme mucho ya que hay infinidad de estas aplicaciones.

### **1. Fail2Ban:**

Esta aplicación es una de las más conocidas y usadas debido a que monitorea múltiples logs en tiempo real. Usa expresiones regulares (regex) para detectar patrones complejos. Además bloquea IPs temporalmente (con temporizadores) y soporta múltiples servicios (SSH, Apache, etc.). Y también tiene integración con firewalls como iptables, ufw, o nftables.

Fail2Ban es la más similar a mi script que encuentre y creo que es interesante ya que con mi simple script, casi hacemos lo mismo que esta aplicación

### **2. Cloudflare:**

Este servicio es mucho más complejo y tiene muchas más características de las que voy a decir a continuación pero las principales son las siguientes:

- Ofrece protección en capa de red (CDN) y mitigación DDoS distribuida.
- Filtra tráfico malicioso antes de llegar al servidor.
- Y también incluye rate limiting, WAF (Web Application Firewall), y análisis de comportamiento.

Además creo que una opción interesante que tiene es que antes de pasar las consultas por tu servidor apache2, el servicio de Cloudflare, analiza las peticiones en su servidor, y si detecta un pico inusual de conexiones las bloquea sin afectar al rendimiento del servidor.

### **3. ModSecurity + OWASP CRS:**

El ModSecurity es básicamente un firewall que detecta ataques en tiempo real y bloquea solicitudes malformadas antes de que entren en tu servidor y también

las bloquea, además WAF también cuenta con reglas avanzadas para bloquear ataques (SQLi, XSS, etc.).

A continuación dejaré una tabla que creo que representa muy bien de manera gráfica, las diferencias entre mi script y otros servicios externos:

<b>Aspecto</b>	<b>Mi script</b>	<b>Otras Aplicaciones</b>
<b>Detección</b>	Umbral fijo y patrones simples.	Regex, machine learning (Cloudflare), rate limiting.
<b>Bloqueo</b>	IPs y rangos /24 (puede generar falsos positivos).	Bloqueos temporales, rangos personalizados, CDN.
<b>Escalabilidad</b>	Adecuado para servidores pequeños.	Diseñado para alto tráfico (ej: Cloudflare).
<b>Tiempo Real</b>	Ejecución manual o vía cron.	Monitoreo 24/7 (Fail2Ban, WAFs).
<b>Tipos de Ataque</b>	DDoS básico y escaneos.	Mitiga DDoS de capa 3/4/7, SQLi, XSS, etc.
<b>Recursos</b>	Bajo consumo (scripts simples).	Puede requerir infraestructura adicional (ej: CDN).

A modo de conclusión final, puedo decir que el script que cree en este proyecto es una solución básica y eficaz para ataques sencillos, más optimizado para entornos pequeños. Sin embargo, carece de mecanismos avanzados como rate limiting dinámico, o protección contra falsos positivos y solo tiene detección de ataques de capa 7. Además hay otras Aplicaciones (como Fail2Ban, Cloudflare) que ofrecen más escalabilidad, precisión y protección en tiempo real, aunque también hay que decir que requieren de configuración compleja o costos adicionales.

## **11. Posibles ampliaciones**

Segun me he ido informando, a continuación dejare posibles ampliaciones que yo no podré hacer en este proyecto pero es bueno tenerlo anotado para saber qué cosas más se le podrían añadir a este script para hacerlo más efectivo e interesante:

### **1. Análisis más avanzado de tráfico**

Podría integrar herramientas como Wireshark o Suricata para realizar un análisis más detallado del tráfico de la red y obtener estadísticas de los ataques.

### **2. Usar Machine Learning**

Junto con el punto anterior, se podrían utilizar conocimientos de machine learning para identificar conexiones anómalas de manera más precisa.

### **3. Mejorar la automatización con un sistema de monitoreo**

Podría crear un sistema de monitoreo más sofisticado utilizando por ejemplo Prometheus para capturar métricas y alertas cuando el tráfico en el servidor sea inusualmente alto (lo que puede ser indicativo de un ataque DDoS).

### **4. Integración con Fail2ban**

Como ya he explicado en el apartado de comparación, Fail2ban es una herramienta que monitorea los logs de los servicios (como SSH, Apache, etc.) y bloquea automáticamente las IPs que parecen maliciosas.

### **5. Uso de iptables con recent para Bloqueos Temporales**

Podría usar el módulo recent de iptables para bloquear IPs temporalmente y luego desbloquearlas automáticamente después de un tiempo. Aunque en este proyecto lo he realizado de manera menos efectiva pero más sencilla de aprender.

### **6. Uso de Cloudflare como CDN**

Podría usar el servidor de Cloudflare ya que en situaciones reales, los ataques DDoS pueden ser tan masivos que es necesario utilizar servicios de

mitigación en la nube como Cloudflare o AWS Shield. Estos servicios proporcionan protección adicional y pueden manejar grandes volúmenes de tráfico lo que puede ayudar a mitigar ataques antes de que lleguen a tu servidor.

## 7. Uso de mod\_security para Protección Adicional

Podría usar el mod\_security que es un módulo de Apache que actúa como un firewall de aplicaciones web (WAF). Este mod también es muy usado sobretodo en apache (sudo apt-get install libapache2-mod-security2)

## 12. Errores durante la realización

Esta penúltima sección de la memoria tratará de ir enseñando algunos de los errores que he tenido durante la realización de este trabajo ya que muchos errores eran por pequeñas cosas que no tienen mayor importancia. Una vez dicho esto, comienzo:

Uno de los primeros errores y posiblemente más tontos, es que al hacer un ataque desde el kali y que luego me baneara el servidor, claramente no podía hacer curl en el kali y debía hacerlo en el servidor.

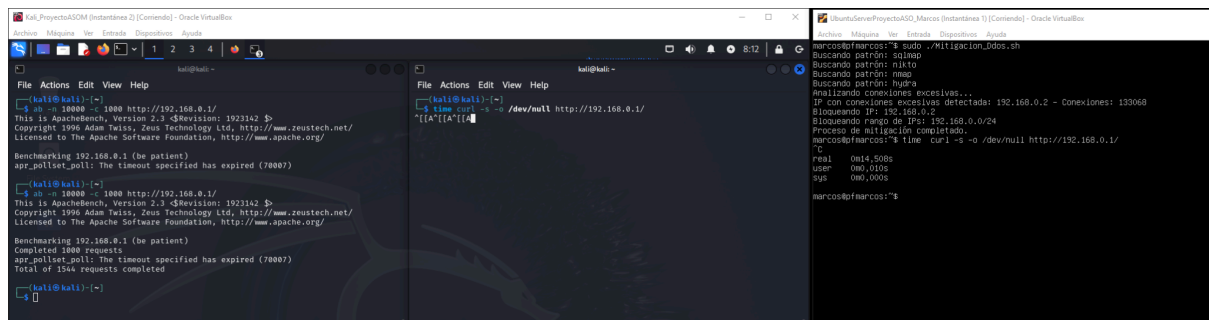


Imagen ilustrativa

Otro error es que al principio tenía un único comando para mandar emails pero me di cuenta de que había que configurar un fichero e instalar dos aplicaciones para realizarlo. A parte que el proceso no era tan fácil como pensaba ya que en algunas cuentas hace falta la contraseña para aplicaciones que es la que muestro en la memoria. Y a partir de aquí es de donde saque la idea de también hacer un script para automatizar la tarea del correo.

Pasando ahora por los errores del propio código de Mitigación, uno de los errores era que al reiniciar apache, no había logs y por lo tanto daba un error en la línea 111 y el script se paraba, por lo que tuve que añadir unas líneas para solucionarlo.

Otro error de código fue para que contara las ips bien, ya que yo realice el código a mano, y ayudándome en las partes más complejas, pero en las líneas que recorría las ips con un for, como que el for cogía mal las líneas y el script no detectaba bien las ips. Y tras consultarlo, decidí cambiarlo por una estructura que no conocía con while, pero que soluciona el error como se puede ver a continuación:

```
# Recorro cada línea de la lista de IPs y sus conexiones
for ip_por_ip in $ips
do
```

Imagen ilustrativa

Y lo cambie por un while:

```
# Recorro cada línea de la lista de
while read -r ip_por_ip;
do
```

Imagen ilustrativa

```
done <<< $ips
```

Imagen ilustrativa

El siguiente punto no es un error un error como tal, pero yo iba probando y duplicando códigos a los que les ponía nombres poco identificativos, por lo que a veces me liaba y no sabía cual era el que tenía lo último que había cambiado, por ello, esto es un recordatorio para nombrar mejor las copias y pruebas que haga:

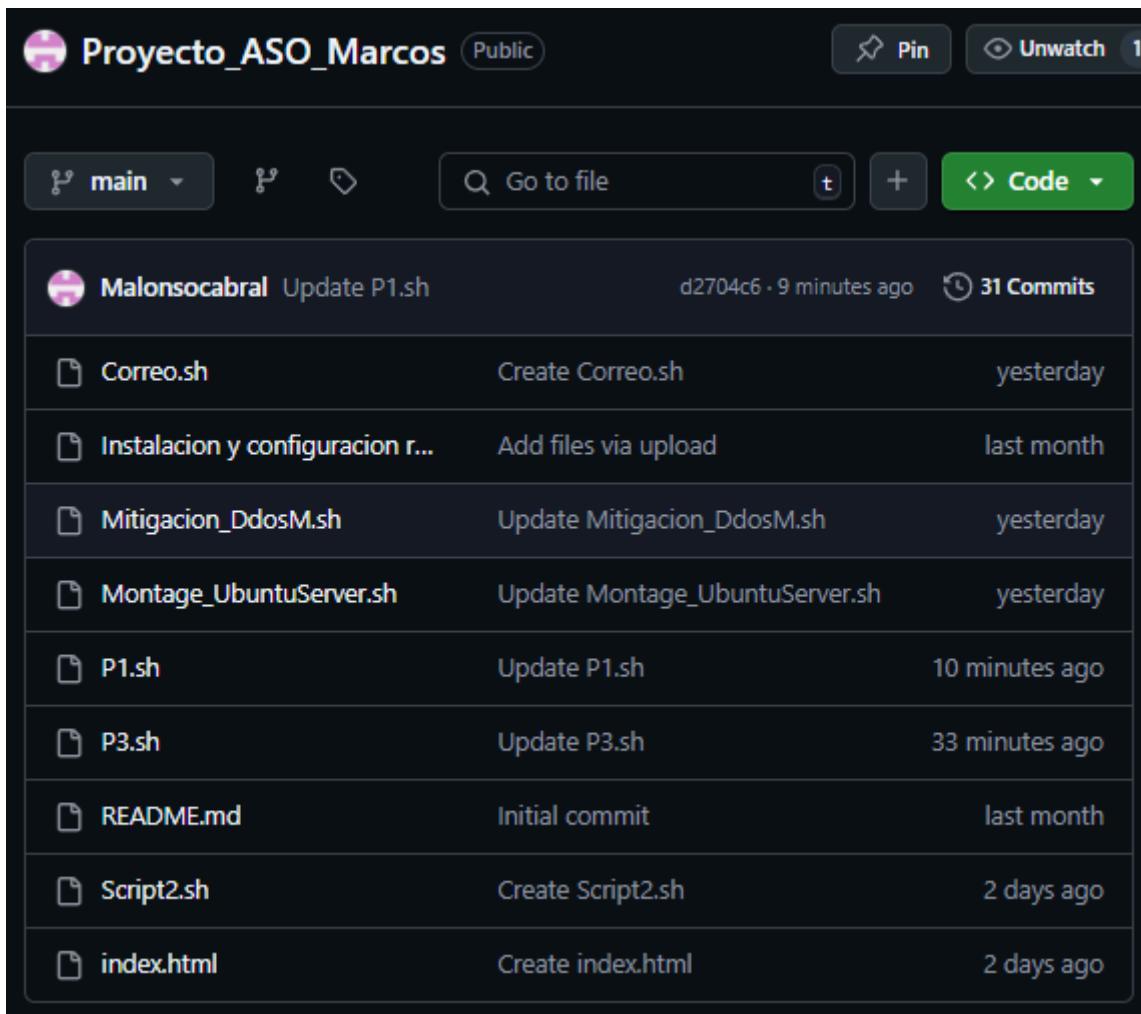


Imagen ilustrativa

El siguiente error es simplemente una aclaración y que al integrar la función de tamaño al final del script me di cuenta de que bash no interpreta las “ñ”.

Otro error importante fue que al integrar la función de bloquear el rango luego no podía comprobar con “time” el tiempo de respuesta ya que estaba baneado el propio servidor.

```

UbuntuServerProyectoASO_Marcos (Instantánea 2) [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
marcos@pfmarcos:~$ sudo ./030_Script_Mitigacion-DDOS_Marcos.sh
Ejecutando como sudo!
Buscando patrón: sqlmap
Buscando patrón: nikto
Buscando patrón: nmap
Buscando patrón: hydra
Analizando conexiones excesivas...
IP con conexiones excesivas detectada: 192.168.0.2 - Conexiones: 12054
Bloqueando IP: 192.168.0.2
Bloqueando rango de IPs: 192.168.0.0/24

Aun no es la hora en punto, por lo que no se vaciaran ni los logs de apache ni las iptables.

Proceso de mitigación completado.
marcos@pfmarcos:~$ time curl -s -o /dev/null http://192.168.0.1/
^C

real    0m29,301s
user    0m0,007s
sys     0m0,007s
marcos@pfmarcos:~$

```

Imagen ilustrativa

Por lo que debí añadir las siguientes líneas en el script principal para que no lo bloqueara:

```

# Permito la ip 192.168.0.1 que es el server para poder hacer comp
sudo iptables -I INPUT -s "192.168.0.1" -j ACCEPT

```

Imagen ilustrativa

```

if [ "$rango_ip.0/24" != "192.168.0.0/24" ]
then
    # Bloqueamos el rango de ips añadiendolo al final de nuestro fichero de ip tables y ejecutandolo.
    echo "Bloqueando rango de IPs: $rango_ip.0/24"
    echo "#Bloqueando el rango de ips: '$rango_ip.0/24'"
    iptables -A INPUT -s '$rango_ip.0/24' -j DROP" >> $f_iptables
    sudo $f_iptables
    # Debido a un error mencionado en el apartado de errores de la memoria implemento el siguiente codigo
    if [ "$rango_ip.0/24" != ".0/24" ]
    then
        echo "$rango_ip.0/24" >> "$ips_bloqueadas"
    fi
else
    echo "Como el rango de ips, es el $rango_ip.0/24 y engloba a este propio servidor, no sera bloqueado"
fi

```

Imagen ilustrativa



El siguiente error trata de que es importante poner el crontab con “sudo” en sudo crontab -e para que el script se ejecute como admin y que no lleguen emails tal que este:

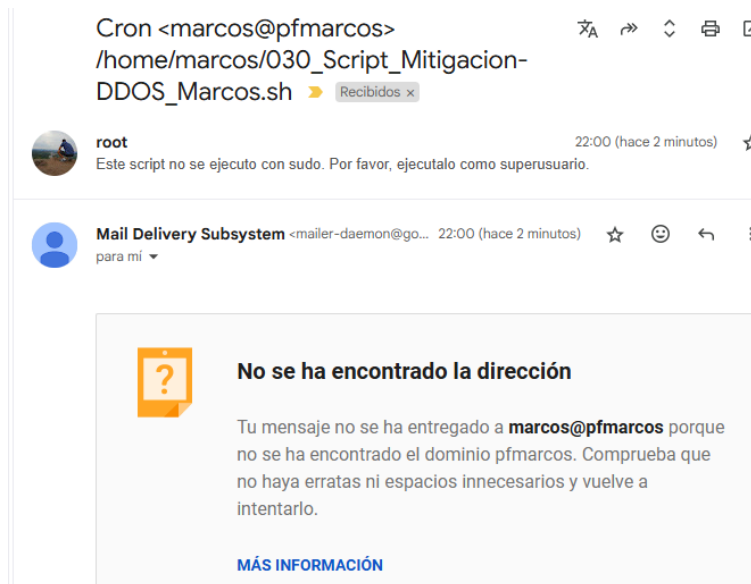


Imagen ilustrativa

Y que en su lugar lleguen tal que así:



Imagen ilustrativa

Cabe destacar que he tenido muchos más errores y aprendizajes durante este proyecto pero no me ha dado para anotarlos y redactarlos.

## **13. Webgrafía**

### **1. Descripción de lo utilizado**

- **Ubuntu Server 22.04.5:** <https://ubuntu.com/download/server>
- **Servidor web Apache2:**
  - Documentación oficial: <https://httpd.apache.org/docs/>
  - Guía instalación:  
<https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-es>
- **Script de prevención en bash:**
  - Explicación sobre iptables:  
<https://help.ubuntu.com/community/IptablesHowTo>
  - Listado de ataques y herramientas (hay demasiadas y no me informé de todas):  
[https://owasp.org/www-community/Vulnerability\\_Scanning\\_Tools](https://owasp.org/www-community/Vulnerability_Scanning_Tools)
- **Repositorio en GitHub del proyecto:**  
[https://github.com/Malonsocabral/Proyecto\\_ASO\\_Marcos](https://github.com/Malonsocabral/Proyecto_ASO_Marcos)

### **2. Montaje Ubuntu Server (con script)**

- Mi propio conocimiento (ASIR)

### **3. Instalación de Apache2**

- Guía instalación:  
<https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-es>

### **4. Montaje de la función Email**

- Instalación de mailutils:  
<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-postfix-as-a-send-only-smtp-server-on-ubuntu-20-04-es>

- Generación de contraseñas para aplicaciones en Google:

<https://myaccount.google.com/apppasswords>

## 5. Montaje del script de Mitigación

- Mi propio conocimiento (ASIR), con ayuda de diferentes cursos y fuentes en para las partes conflictivas.

## 6. Montaje Kali (instalación herramientas)

- Informe sobre ataques DDOS:

[https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjRxp3xopyMAxXvT6QEHcHCMSoQFnoECCYQAQ&url=https%3A%2F%2Fwww.dilemascontemporaneoseduccionpoliticaayvalores.com%2Findex.php%2Fdilemas%2Farticle%2Fdownload%2F2248%2F2301%2F&usg=AOvVaw1nm9UOfk2\\_3kmmjshCfCG&opi=89978449](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjRxp3xopyMAxXvT6QEHcHCMSoQFnoECCYQAQ&url=https%3A%2F%2Fwww.dilemascontemporaneoseduccionpoliticaayvalores.com%2Findex.php%2Fdilemas%2Farticle%2Fdownload%2F2248%2F2301%2F&usg=AOvVaw1nm9UOfk2_3kmmjshCfCG&opi=89978449)

- Descarga oficial de Kali Linux:

<https://www.kali.org/get-kali/#kali-virtual-machines>

- Apache Benchmark (ab):

<https://httpd.apache.org/docs/current/programs/ab.html>

- Hping3:

<https://www.redeszone.net/tutoriales/seguridad/hping3-manipular-paquetes-tcp-ip-ataques/>

## 7. Simulación de situación real

- Monitorización con netstat:

<https://go.lightnode.com/es/tech/linux-netstat-command>

- Uso de time junto con curl para medir tiempos de respuesta:

<https://www.hostinger.com/es/tutoriales/comando-linux-time> y Mi propio

conocimiento (ASIR).

## 8. Instalación de crontab

- Configuración de tareas automáticas en Linux

- <https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-ubuntu-1804-es>

## 10. Comparación de mi script con otras aplicaciones similares

- Fail2Ban:

<https://www.arsys.es/blog/instalar-fail2ban>

- Cloudflare:

<https://www.cloudflare.com/es-es/learning/ddos/ddos-mitigation/>

- ModSecurity + OWASP CRS:

<https://docs.bluehosting.cl/tutoriales/servidores/instalacion-y-configuracion-de-la-aplicacion-mod-security-en-apache.html>

## 11. Posibles ampliaciones (suposiciones)

- Wireshark para análisis de tráfico
- Suricata IDS
- Uso de Machine Learning para detección de anomalías:
- Prometheus para monitoreo avanzado:

<https://prometheus.io/>