PRÁCTICA 4

Seguridad (Cortafuegos)

UNIVERSIDAD DE GRANADA

Titulación: Ingeniería Informática + ADE

FLORIN EMANUEL TODOR GLIGA

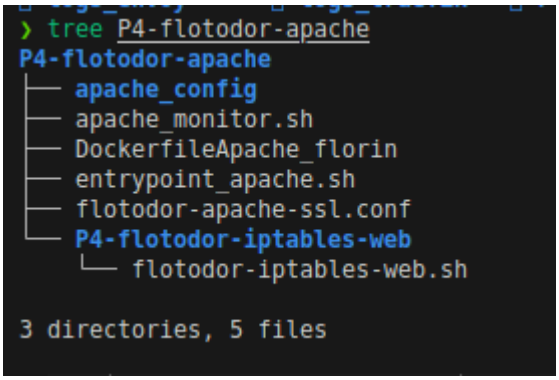
ÍNDICE

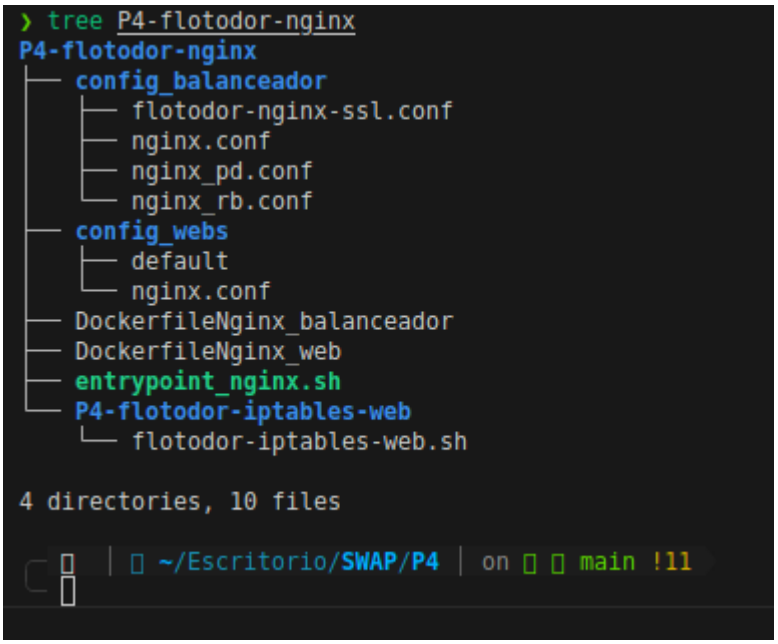
Tareas básicas

Como comenté en la anterior práctica, yo voy acumulando todos los ficheros en cada práctica e insertando lo nuevo, ya que quiero tener todo incorporado en el propio script inicial de init.sh. Además, aunque tenga los docker compose del resto de balanceadores de la práctica 2, por ahora solamente estoy implementando en el docker compose de nginx balanceador (fuera de las prácticas ya lo implementaré todo para el resto de balanceadores). **Por ello estoy ejecutando en todo momento ./init.sh -u nginx**

B1. Preparación del Entorno de Trabajo

Creo el directorio con el script que se nos solicita tanto en la carpeta apache como en la de nginx (para los servidores webs que tengo).





B2. Creación y Configuración de Scripts IPTABLES

Creo el script con GPT (que es tal cual la misma información de las diapositivas que se nos da en la práctica pero para automatizar el proceso):

```
#!/bin/bash
      You, 16 hours ago • P4 SWAP basics finished
# Script de configuración de IPTABLES para contenedores web
# Autor: Florin Emanuel Todor Gliga

# IP del balanceador de carga (ajustar si cambia)
BALANCER_IP="192.168.10.50"

# Establecer políticas por defecto (Denegación implícita)
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Permitir tráfico en la interfaz de loopback (localhost)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Permitir tráfico de conexiones ya establecidas o relacionadas (entrante)
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Permitir tráfico de conexiones nuevas, establecidas y relacionadas (saliente)
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Permitir tráfico HTTP (puerto 80) desde el balanceador
iptables -A INPUT -p tcp -s $BALANCER_IP --dport 80 -j ACCEPT

# Permitir tráfico HTTPS (puerto 443) desde el balanceador
iptables -A INPUT -p tcp -s $BALANCER_IP --dport 443 -j ACCEPT
```

Explicar las líneas de cada política me parece pérdida de tiempo cuando lo explico en los comentarios y vienen en el pdf de la práctica.

B3. Implementación de Scripts IPTABLES en Docker

Para esta parte debo de comentar que la parte de entrypoint.sh lo llevo usando desde la práctica 1 debido a que ya tuve que insertar un script para el tema de los logs que cree personalizados para apache.

```
# Cambiamos la configuración de apache para poner mayor seguridad
COPY ./P4-flotodor-apache/apache_config/.htaccess /var/www/html/.htaccess

# Copiamos el script de IPTables
COPY ./P4-flotodor-apache/P4-flotodor-iptables-web/flotodor-iptables-web.sh /usr/local/bin/iptables.sh
RUN chmod +x /usr/local/bin/iptables.sh

# Copiar el script de monitoreo
COPY ./P4-flotodor-apache/apache_monitor.sh /usr/local/bin/apache_monitor.sh
RUN chmod +x /usr/local/bin/apache_monitor.sh

### EJecución de servicios de la imagen

COPY ./P4-flotodor-apache/entrypoint_apache.sh /usr/local/bin/entrypoint_apache.sh
RUN chmod +x /usr/local/bin/entrypoint_apache.sh
RUN echo 'DirectoryIndex index.php index.html' >> /etc/apache2/apache2.conf
```

```
    EXPOSE 80 443 9100
    ENTRYPOINT ["/usr/local/bin/entrypoint_apache.sh"]
```

Como vemos en las imágenes, envio desde la práctica 1 el entrypoint que cree con las otras prácticas.

```
You, 16 hours ago | 1 author (You)
#!/bin/bash
      You, 16 hours ago • P4 SWAP basics finished
# Función para verificar que existen los certificados
function check_ssl_certs() {
  CERT_KEY="/etc/apache2/ssl/certificado_flotodor.key"
  CERT_CRT="/etc/apache2/ssl/certificado_flotodor.crt"

  # Esperar si los ficheros no existen todavía
  while [ ! -s "$CERT_KEY" ] || [ ! -s "$CERT_CRT" ]; do
    echo "[!] Esperando certificados SSL..."
    sleep 1
  done

  echo "[✓] Certificados SSL encontrados y válidos. Continuando arranque..."
}

# Inicia el script de monitoreo en segundo plano
/usr/local/bin/apache_monitor.sh &

# Iniciamos el scirpt de IPTABLES en segundo plano
/usr/local/bin/iptables.sh &

# Inicia Node Exporter en segundo plano
/usr/local/bin/node_exporter &

# --- NUEVO ---
# Espera activa por certificados
check_ssl_certs

# Ahora sí, inicia Apache en primer plano
exec /usr/sbin/apache2ctl -D FOREGROUND
```

Este es mi entrypoint, con una función para comprobar en la práctica 3 que con la función copy funcionaba enviar los certificados (aunque es más relevante copiarlos en el propio docker-compose del balanceador).

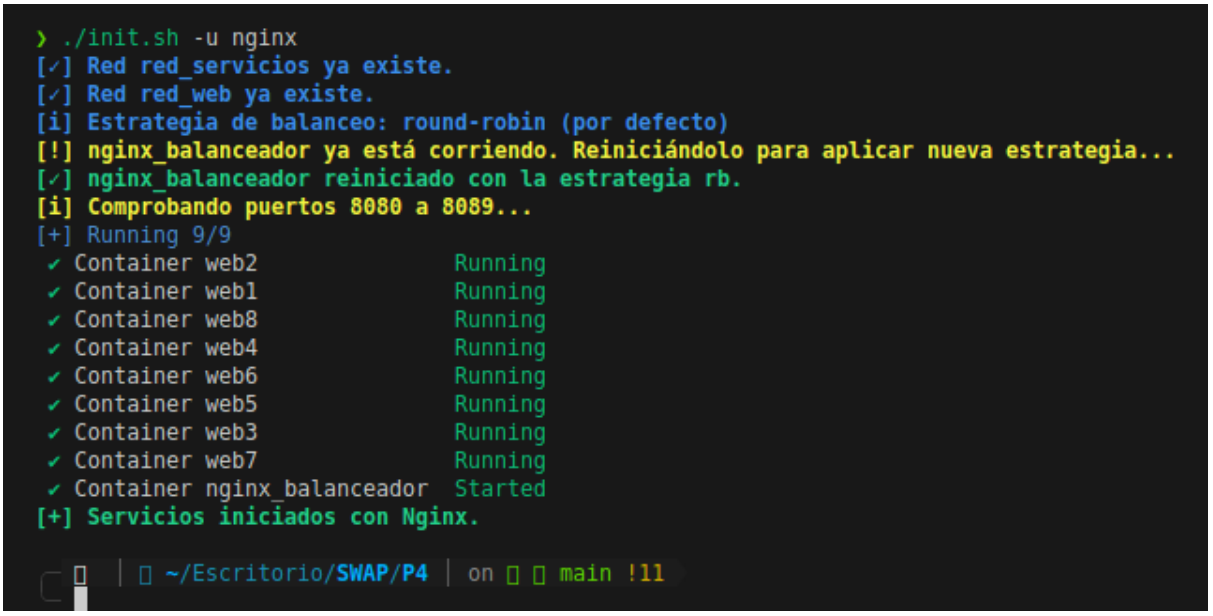
Vemos cómo ejecutamos en segundo plano todos los scripts de las prácticas, como es el caso de iptables que es para esta.

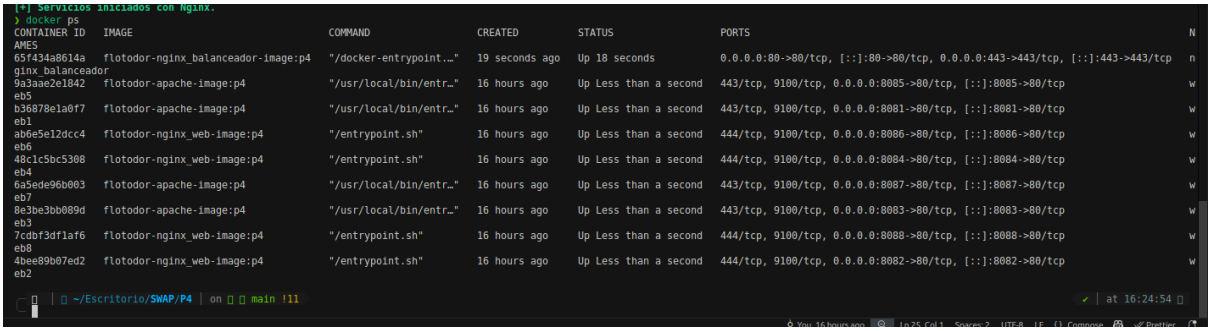
B4. Configuración de Docker Compose

```yaml
# Datos comunes para todos los servicios de apache
x-common-apache-config: &common-apache-config
  image: flotodor-apache-image:p4
  restart: always
  volumes:
    - ./web_flotodor:/var/www/html
    - ./logs_apache:/var/log/apache2
    - ./P4-flotodor-certificados:/etc/apache2/ssl
  cap_add:
    - NET_ADMIN


# Datos comunes para todos los servicios de nginx
x-common-nginx_web-config: &common-nginx_web-config
  image: flotodor-nginx_web-image:p4
  restart: always
  volumes:
    - ./web_flotodor:/usr/share/nginx/html:ro
    - ./logs_nginx:/var/log/nginx
    - ./P4-flotodor-certificados:/etc/nginx/ssl
  cap_add:
    - NET_ADMIN

      You, 16 hours ago • P4 SWAP basics finished
x-common-nginx_balanceador-config: &common-nginx_balanceador-config
  image: flotodor-nginx_balanceador-image:p4
  restart: always
  volumes:
    - ./P4-flotodor-nginx/config_balanceador/flotodor-nginx-ssl.conf:/etc/nginx/nginx.conf  # << Montar nginx.conf SSL
    - ./P4-flotodor-certificados:/etc/nginx/ssl                                             # << Montar certificados SSL
    - ./logs_nginx:/var/log/nginx
  cap_add:
    - NET_ADMIN
```

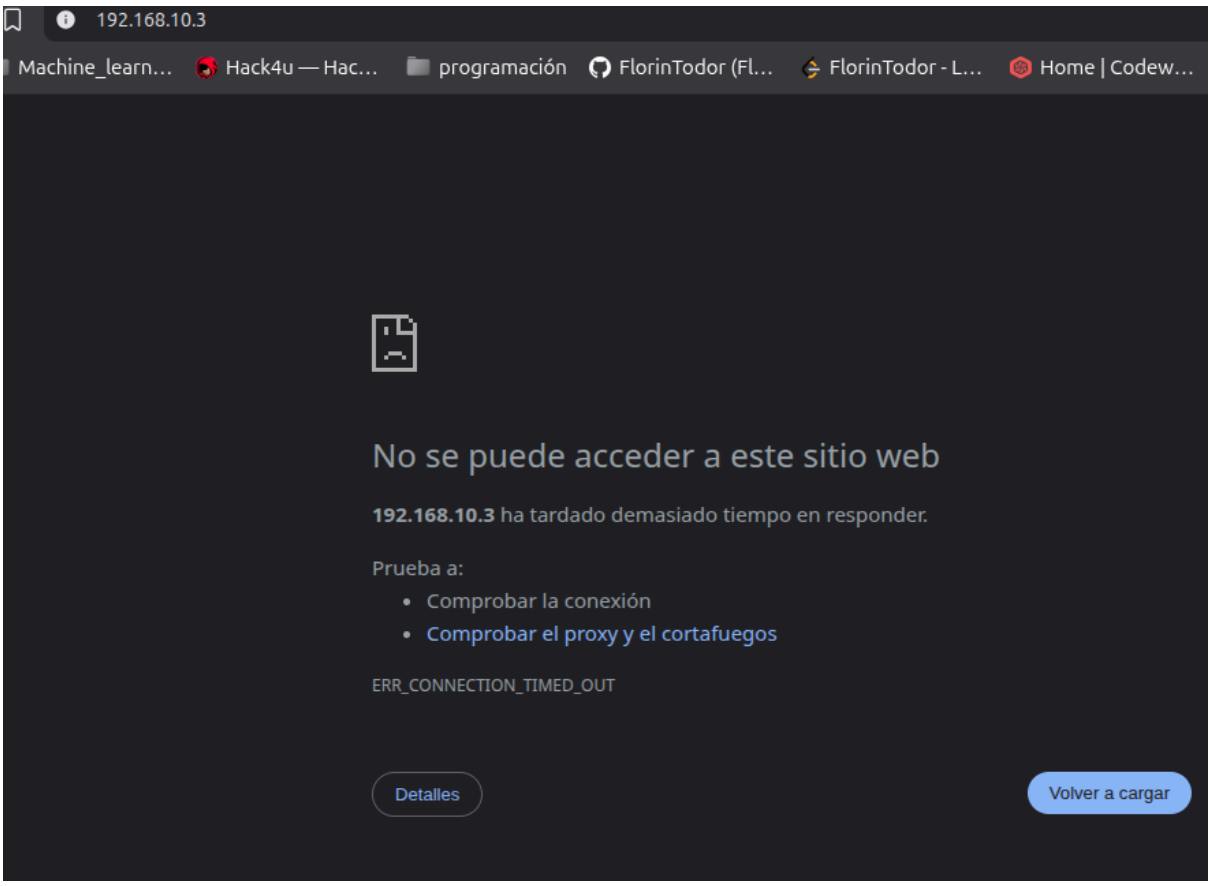
Como vemos, le estoy dando permisos a los contenedores webs (apache y nginx) como al balanceador para poder ejecutar los iptables, aunque el balanceador no hace uso del iptables (Podría realmente borrarlo, pero creo que luego en la parte avanzada si hago uso de iptables en el balanceador, a si que lo voy a dejar).

B5. Verificación y Pruebas

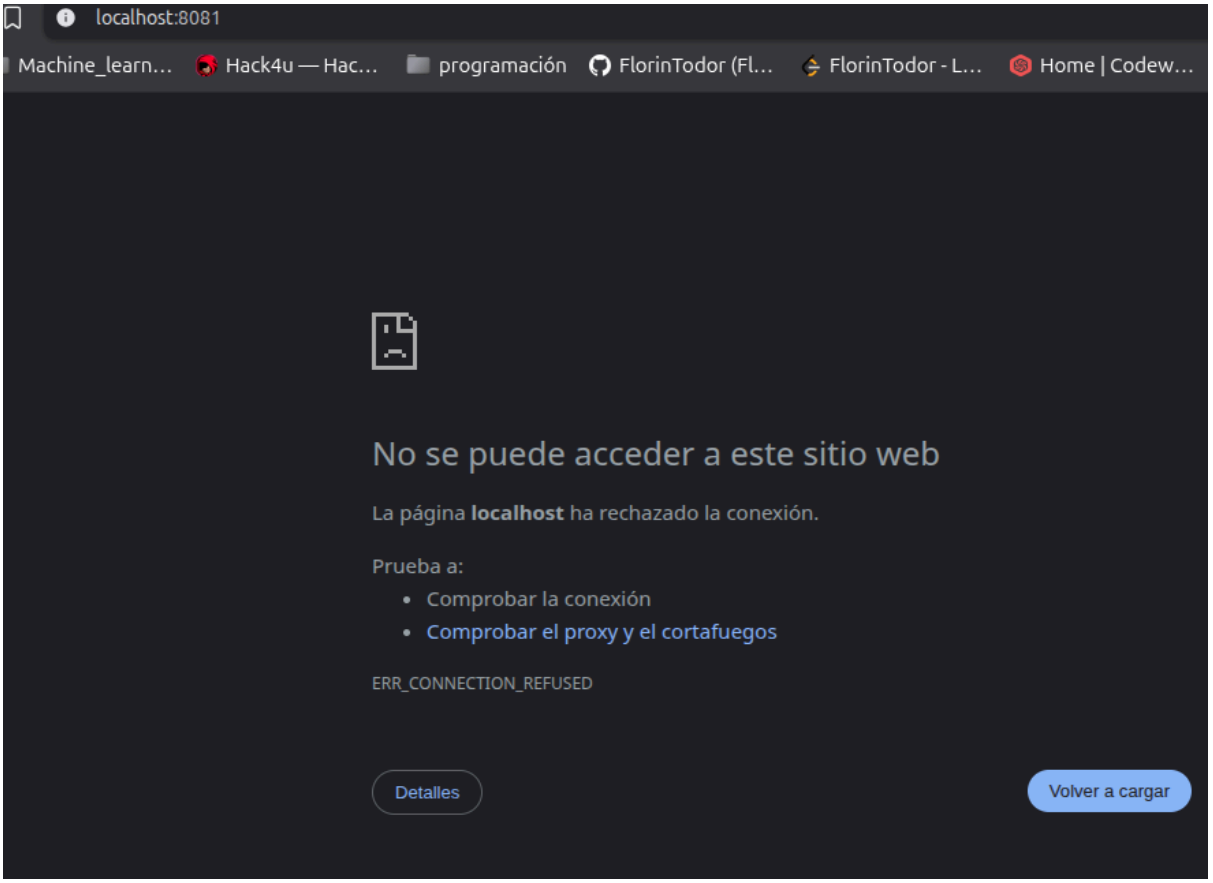




Comprobaciones:

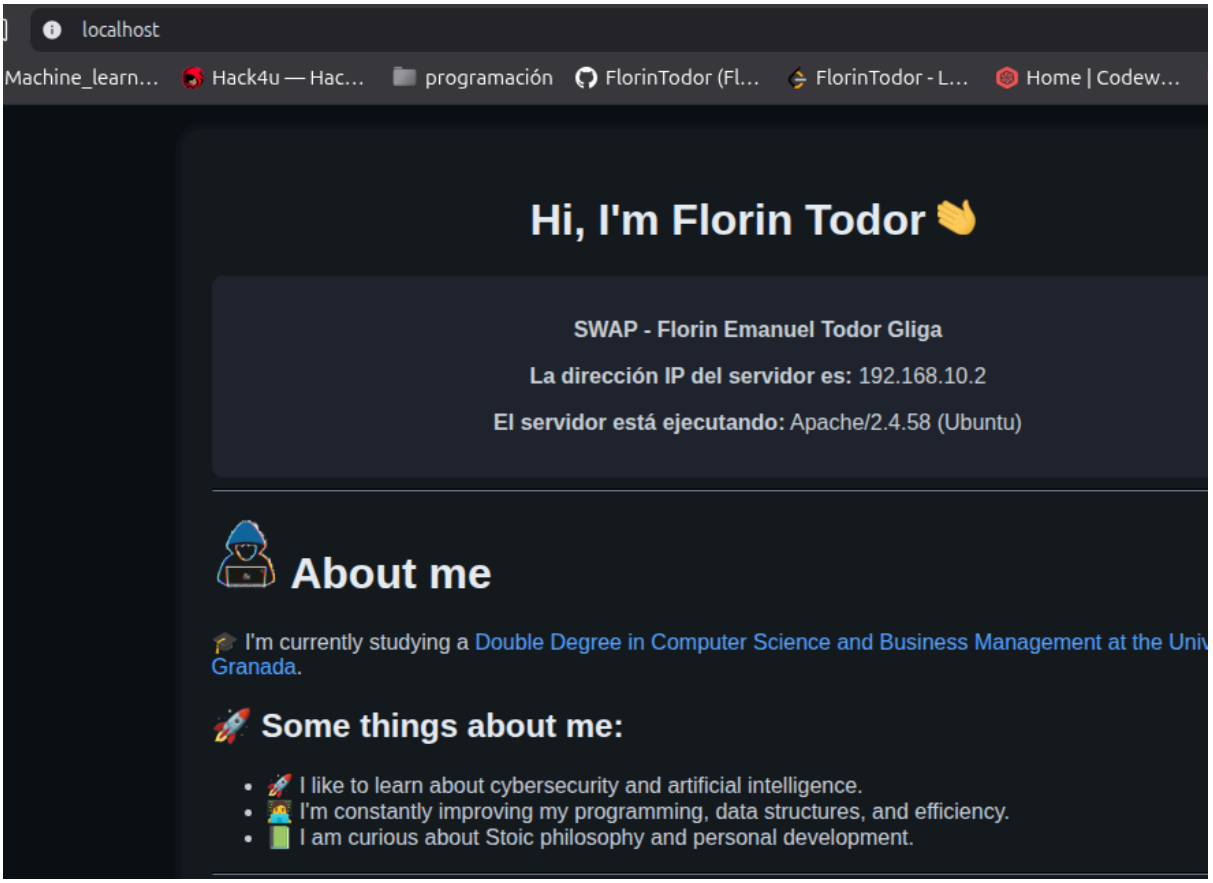


Vemos que acceder directamente al contenedor web no nos permite.

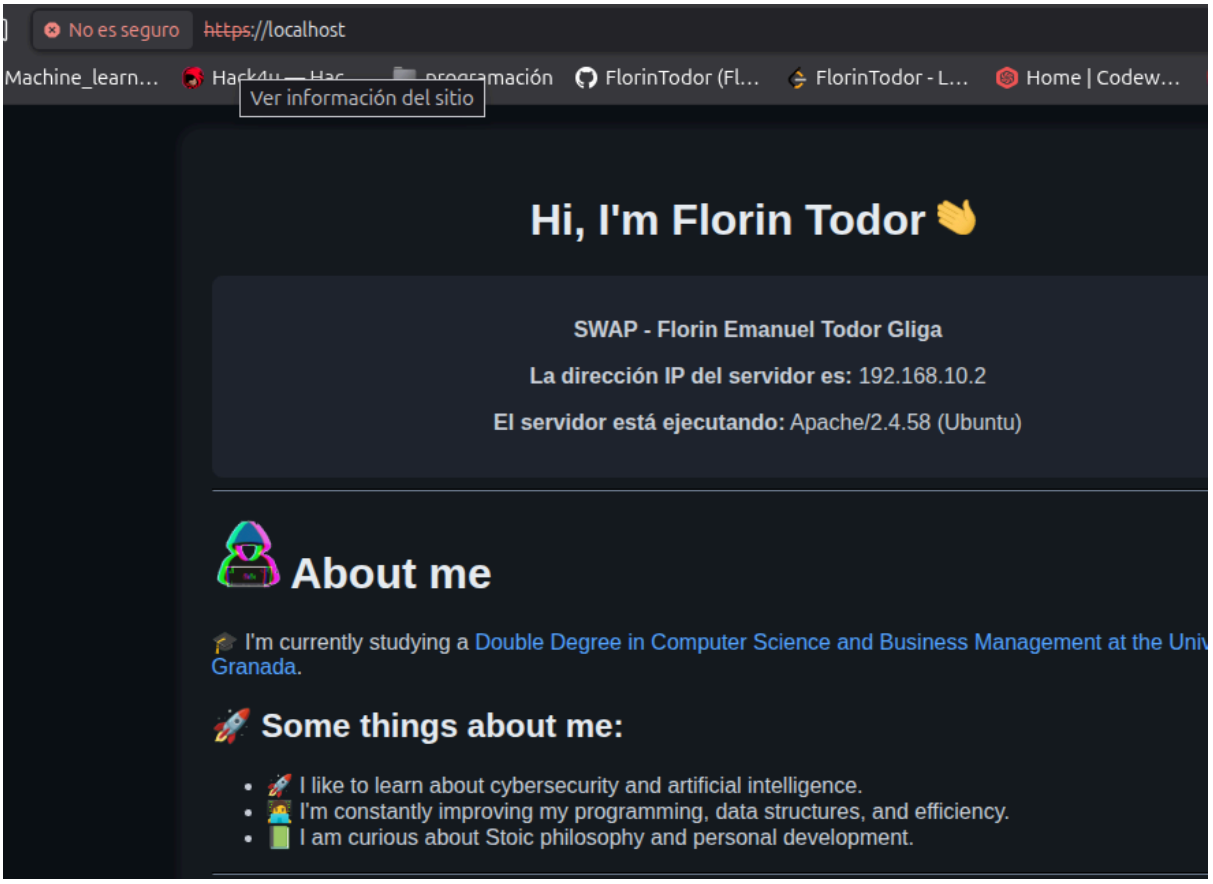


desde el propio balanceador tampoco nos deja

Ejecutar el balanceador desde http si nos permite:



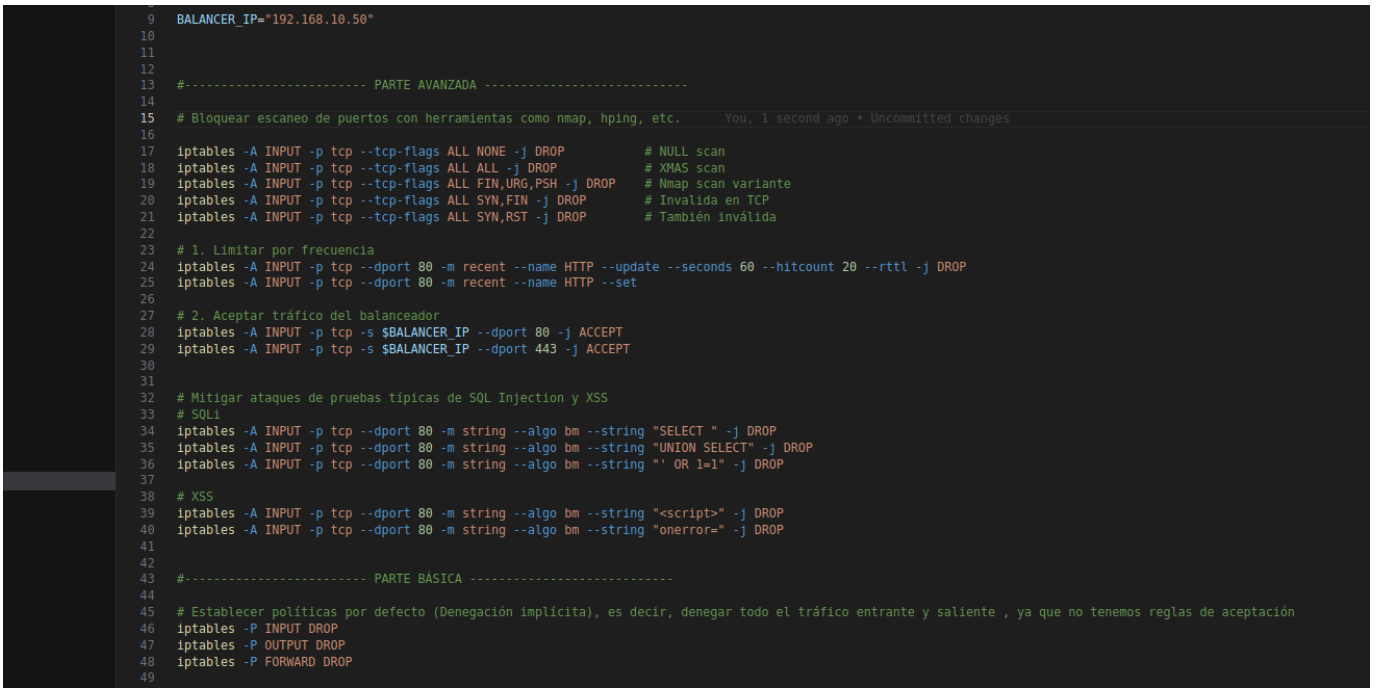
Y desde https también nos permiten:


No es seguro
https://localhost
Machine_learn...
Ver información del sitio
programación
FlorinTodor (Fl...
FlorinTodor - L...
Home | Codew...
Hi, I'm Florin Todor 👋
SWAP - Florin Emanuel Todor Gliga
La dirección IP del servidor es: 192.168.10.2
El servidor está ejecutando: Apache/2.4.58 (Ubuntu)
About me
🎓 I'm currently studying a Double Degree in Computer Science and Business Management at the Univ
Granada.
🚀 Some things about me:
🚀 I like to learn about cybersecurity and artificial intelligence.
👨‍💻 I'm constantly improving my programming, data structures, and efficiency.
📗 I am curious about Stoic philosophy and personal development.

Tareas avanzadas

A1. Definir e implementar políticas de seguridad en el balanceador de carga

Para esta parte, he implementado diferentes políticas, debido a que hay varios temas que ya conocía previamente de hacer explotaciones en algunos servidores webs, tanto con SQLi como XSS.



```
BALANCER_IP="192.168.10.50"



#------------------------- PARTE AVANZADA ---------------------------

# Bloquear escaneo de puertos con herramientas como nmap, hping, etc.      You, 1 second ago • Uncommitted changes

iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP           # NULL scan
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP            # XMAS scan
iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP    # Nmap scan variante
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN -j DROP        # Invalida en TCP
iptables -A INPUT -p tcp --tcp-flags ALL SYN,RST -j DROP        # También inválida

# 1. Limitar por frecuencia
iptables -A INPUT -p tcp --dport 80 -m recent --name HTTP --update --seconds 60 --hitcount 20 --rttl -j DROP
iptables -A INPUT -p tcp --dport 80 -m recent --name HTTP --set

# 2. Aceptar tráfico del balanceador
iptables -A INPUT -p tcp -s $BALANCER_IP --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -s $BALANCER_IP --dport 443 -j ACCEPT


# Mitigar ataques de pruebas típicas de SQL Injection y XSS
# SQLi
iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "SELECT " -j DROP
iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "UNION SELECT" -j DROP
iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "' OR 1=1" -j DROP

# XSS
iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "<script>" -j DROP
iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "onerror=" -j DROP


#------------------------- PARTE BÁSICA ---------------------------

# Establecer políticas por defecto (Denegación implícita), es decir, denegar todo el tráfico entrante y saliente , ya que no tenemos reglas de aceptación
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

```

He implementado el bloqueo de escaneo de puertos con diferentes herramientas y protocolos.
Por otro lado he limitado el acceso continuo al balanceador de carga, es decir, si una misma ip intenta acceder en menos de 1 minuto 20 veces al balanceador, se le bloquea la ip durante un tiempo hasta que posteriormente se le restablece. Esto viene bien para intentos de ataques de DDOS.

Además, he limitado las típicas pruebas de inserción de SQL Inyection y de XSS.
Vídeo con la prueba del bloqueo de IP:
https://drive.google.com/file/d/1BiVGTvWAQnq52Y4eKA2Omx1HwNkfOxkc/view?usp=sharing

En este video podemos ver que con el acceso continuo llega un momento en el que se bloquea la IP.

Tras un minuto aproximadamente, se vuelve a permitir el acceso con esa ip.

A2. Configuración Avanzada de IPTABLES para DDoS

En el ejercicio A1, ya he implementado una limitación de tasa de conexiones por IP, es decir, justamente lo que he mostrado en el video de arriba.

GPT ya me ha comentado que tengo implementado la mayoría de medidas: Denegación implicita, Bloqueo de escaneos, Límites de conexiones simultáneas, Protección básica contra SQLi/XSS, Permisos de loopback y conexiones válidas (balanceador).

Aunque nos propone mejoras para el DDoS.

Entre ellas está mejorar el recent para que el bloqueo no sea permanente, Detectar SYN floods (que son ataques típicos de DDos), prevenir los ataques de fragmentación (como se nos pide en el documento), ajustar la tasa de conexiones que hemos creado antes de 20 a 25 peticiones y proteger contra spoofing (muy importante en mi opinión).

Por lo que, GPT me ha reestructurado un poco el fichero del script de ip tables.

```
5
6   BALANCER_IP="192.168.10.50"
7
8   # ---- POLÍTICAS POR DEFECTO ----
9   iptables -P INPUT DROP
10  iptables -P OUTPUT DROP
11  iptables -P FORWARD DROP
12
13  # ---- LOOPBACK Y CONEXIONES ESTABLECIDAS ----
14  iptables -A INPUT -i lo -j ACCEPT
15  iptables -A OUTPUT -o lo -j ACCEPT
16  iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
17  iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT      You, 2 days ago • P4 SWAP basics finished …
18
19  # ---- BLOQUEO ESCANEOS / TCP INVÁLIDO ----
20  iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
21  iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
22  iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
23  iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN -j DROP
24  iptables -A INPUT -p tcp --tcp-flags ALL SYN,RST -j DROP
25
26  # ---- BLOQUEAR IP SPOOFING (IPs privadas falsas) ----
27  iptables -A INPUT -s 10.0.0.0/8 -j DROP
28  iptables -A INPUT -s 172.16.0.0/12 -j DROP
29  iptables -A INPUT -s 192.168.0.0/16 -j DROP
30  iptables -A INPUT -s 127.0.0.0/8 ! -i lo -j DROP
31  iptables -A INPUT -s 169.254.0.0/16 -j DROP
32
33  # ---- PROTECCIÓN FRAGMENTACIÓN IP ----
34  iptables -A INPUT -f -j DROP
35
36  # ---- LIMITAR NUEVAS CONEXIONES ----
37  # Max 25 nuevas conexiones por IP cada 60s
38  iptables -A INPUT -p tcp --syn --dport 80 -m recent --name NEWCONN --set
39  iptables -A INPUT -p tcp --syn --dport 80 -m recent --name NEWCONN --update --seconds 60 --hitcount 25 --rttl -j DROP
40
41  # ---- CONNLIMIT: máximo 10 conexiones simultáneas por IP ----
42  iptables -A INPUT -p tcp --dport 80 -m connlimit --connlimit-above 10 -j REJECT
43
44  # ---- PROTECCIÓN DOS SIMPLE por frecuencia ----
45  iptables -A INPUT -p tcp --dport 80 -m recent --name HTTP --update --seconds 60 --hitcount 20 --rttl -j DROP
46  iptables -A INPUT -p tcp --dport 80 -m recent --name HTTP --set
47
48  # ---- FILTRADO DE SQLi y XSS ----
49  iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "SELECT " -j DROP
50  iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "UNION SELECT" -j DROP
51  iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "' OR 1=1" -j DROP
52  iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "<script>" -j DROP
53  iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "onerror=" -j DROP
54
55  # ---- PERMITIR TRÁFICO DEL BALANCEADOR ----
56  iptables -A INPUT -p tcp -s $BALANCER_IP --dport 80 -j ACCEPT
57  iptables -A INPUT -p tcp -s $BALANCER_IP --dport 443 -j ACCEPT
58
```

A3: Simular ataques a la granja web y configuraciones de seguridad realizadas

Me parece interesante dejar toda esta parte para que se plasme la información de las pruebas/cambios que voy realizando para dejar todo funcionando correctamente.

Bueno, pues para comenzar con esta parte, le he pedido a GPT que me cree un script básico con diferentes pruebas. Al leer el código y verificar la utilidad he comenzado a realizar el ataque de DDoS más simple y la web se ha saturado. Por lo que tengo que seguir modificando el iptables.

Vale, he situado los errores, no era por saturación sino que se estaba bloqueando por spoofing las ip privadas de los propios contenedores de docker (no me di cuenta de esa línea).

Vale, ahora al realizar los cambios (que posteriormente mostraré) ya no se cae la web, las pruebas de XSS y SQLI se ejecutan pero no tienen sentido (no tengo ninguna base con SQL, pero bueno, realizo la prueba).

Sin embargo, he visto que sí permite escanear los puertos (tengo que seguir modificando entonces el iptables). Esto se debe a que solamente estoy limitando algunos ejemplos básicos de escaneos, pero voy a añadir más (con ayuda de la IA).

Vale, entiendo que al tener los puertos abiertos, siempre se pueden escanear porque permiten SYN, ACK. Pero podemos limitar si detectamos un escaneo masivo de muchos puertos a la vez (uso de nmap de forma agresiva) o engañar en sí a los escáneres enviandolo a otro puerto. Para esto último, el uso de Tarpit, hace falta tener instalado xtables-addons. Lo voy a añadir a los docker files (no veo necesario también subir una imagen).

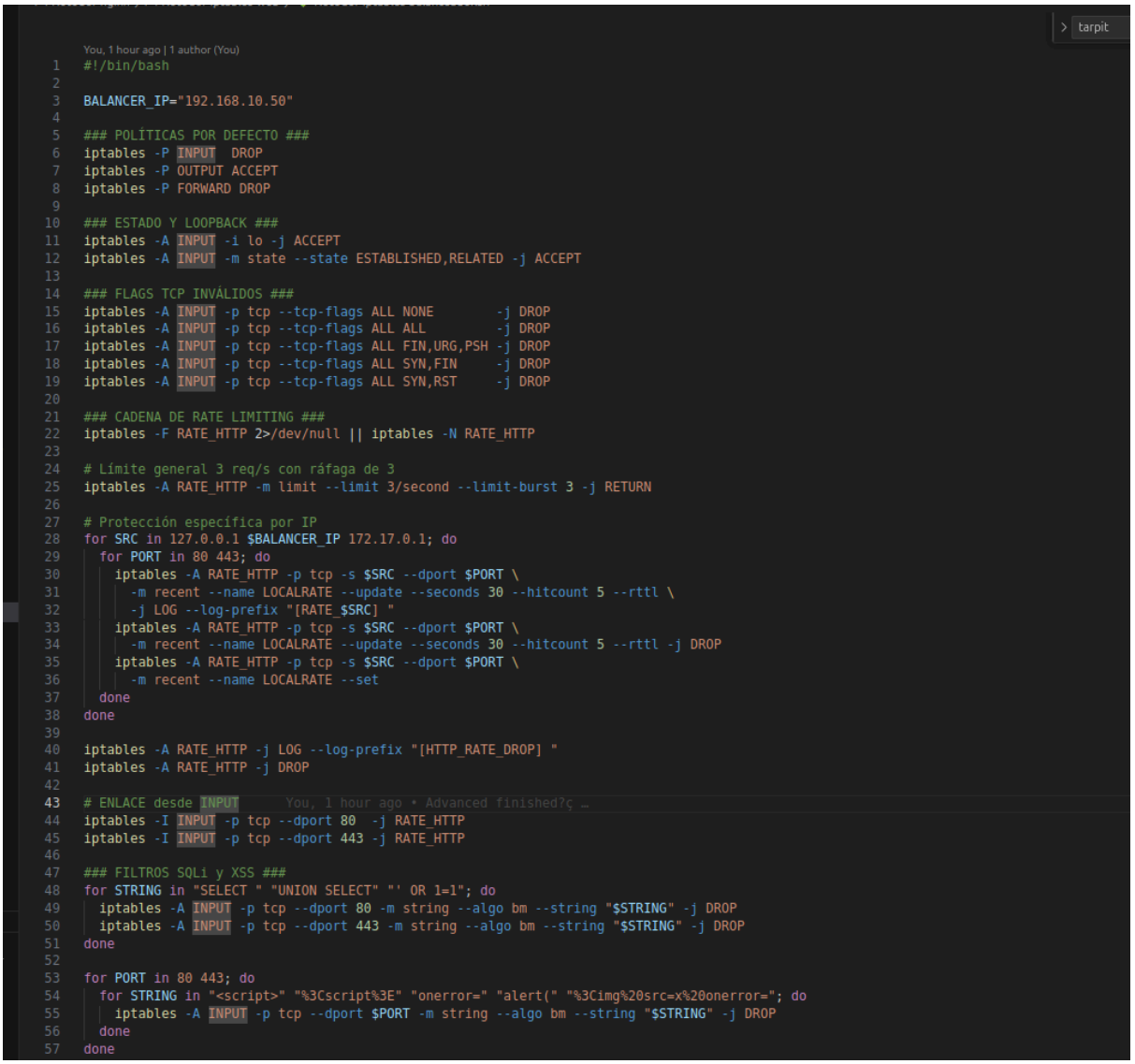
Aunque este último, Tarpit, no tiene mucho sentido aplicarlo porque el puerto que estoy implementando está cerrado y los demás puertos están en uso. Por lo que lo voy a dejar, pero no tiene sentido, pero es interesante aplicarlo. Si tuviera el puerto abierto lo podría probar.

Vale creo que todos los problemas que me está ocurriendo se encuentra en usar el mismo script para los servidores webs que para el balanceador, por lo tanto, voy a crear un script nuevo para el balanceador y enviarlo a su contenedor para ejecutarlo.
Es decir, no estoy usando un script con los IP tables específicos para el balanceador, estoy usando todas las reglas para los servidores webs. Fallo mio por no darme cuenta previamente. Paso a la crear el nuevo script y enviarlo al contenedor del balanceador.

Como he comentado anteriormente, me parece interesante dejar esta parte de comprobaciones que voy realizando durante la creación de la práctica.

Por lo tanto, ahora ya tenemos 2 scripts distintos, el script con la parte básica de IPTABLES, que es para los servidores webs y ahora el script para el balanceador de carga para las distintas pruebas que vamos a ejecutar.

Adjunto la captura del script del balanceador de carga:



Como vemos, ha variado respecto al A1 Y A2, por eso insisto en tener en cuenta todo lo que he comentado. Por ejemplo he eliminado el spoofing (realmente es útil, pero teniendo en cuenta que esto es una prueba en local y que solamente tengo las ips de los servidores webs y del propio balanceador, no es útil bloquear lo demás).

```
### LÍMITE BÁSICO DE SYN FLOOD ###
iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 4 -j ACCEPT
iptables -A INPUT -p tcp --syn -j DROP

### MITIGACIÓN DE CONEXIONES EXCESIVAS POR IP ###
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -m connlimit --connlimit-above 20 --connlimit-mask 32 -j DROP
iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW -m connlimit --connlimit-above 20 --connlimit-mask 32 -j DROP

### LIMITAR FRECUENCIA DE NUEVAS CONEXIONES POR IP (hashlimit) ###
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -m hashlimit --hashlimit 3/sec --hashlimit-burst 5 --hashlimit-mode srcip --hashlimit-name http_limit -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j DROP

iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW -m hashlimit --hashlimit 3/sec --hashlimit-burst 5 --hashlimit-mode srcip --hashlimit-name https_limit -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW -j DROP

### PUERTOS PERMITIDOS ###
for PORT in 80 443 9100 2049 111; do
  iptables -A INPUT -p tcp --dport $PORT -j ACCEPT
done

iptables -A INPUT -p tcp --dport 8081:8088 -j ACCEPT

### BLOQUEO DE RANGOS NO USADOS ###
iptables -A INPUT -p tcp --dport 0:79       -j DROP
iptables -A INPUT -p tcp --dport 81:109     -j DROP
iptables -A INPUT -p tcp --dport 113:442    -j DROP
iptables -A INPUT -p tcp --dport 444:1023   -j DROP
iptables -A INPUT -p tcp --dport 1025:65535 -j DROP

echo "[✓] IPTABLES cargado con refuerzo hashlimit y connlimit por IP."
a       You, 1 hour ago • Uncommitted changes
```

Antes de comenzar a explicar lo que he implementado y que he ido cambiando respecto al A1 Y A2 realizado previamente (esto debido a que no lo estaba haciendo bien porque estaba cambiando los iptables de las webs y no del balanceador), debo de comentar que es importante el orden de las implementaciones de los iptables.

En este script, primero declaro la ip del balanceador, posteriormente se implementa:

- **Políticas por defecto:**
  - El tráfico entrante queda bloqueado por defecto, todo lo que salga del balanceador está permitido y no se permite redirección de paquetes.
- **Permitir tráfico esencial:**
  - Permitimos el uso del propio localhost y evitamos cortar tráfico legítimo.
- **Bloqueo de flags:**
  - Bloqueamos paquetes ilegítimos usados en escaneos o ataques DDoS.
- **Creamos una cadena personalizada para la parte de rate limiting (Uso de RATE_HTTP):**
  - Límite general de 3 solicitudes por segundo con ráfaga maxima de 3, además de implementar protección específica por ip conocidas (el bucle for de SRC), en el que si se hace más de 5 solicitudes en 30 segundos se loguea (se pasa al log) y se bloquea temporalmente.
  - En el caso de no pasar el límite se hace DROP Y LOG, posteriormente de estas comprobaciones para la parte de DDoS, todos los accesos HTTP/HTTPS pasan por la lógica de RATE_HTTP que hemos creado.
- **Filtros contra SQLi y XSS:**
  - Aunque se bloquea correctamente, habría que realizar más modificaciones en los propios ficheros de configuración no solo usar IPTABLES, sobretodo para la parte de XSS.
- **Protección contra SYN flood (es decir, muchas conexiones nuevas):**
  - Solo se permiten 1 nueva conexión con un pequeño margen.
- **Bloqueo de excesivas conexiones nuevas por IP (so de connlimit):**
  - Si una IP abre más de 20 conexiones nuevas simuntánemanete, se bloquea.
- **Limitación de frecuencia de nuevas conexiones por IP (haslimit):**
  - Permitimos hasta 3 conexiones nuevas por segundo, con ráfaga de 5. Si se supera, la conexión se rechaza, efectivo contra bots o herramientas de escaneo.
- **Permitimos puertos explícitos que si se usan y bloqueamos los que no**

Tras comentar la implementación realizada en el balanceador, voy a comentar las pruebas realizadas, el script usado y la salida de dicho script.

El script acepta como argumento la URL del objetivo (por defecto http://localhost) y un tipo de prueba. Tiene una función para cada ataque o test, y cada una mide el tiempo de ejecución y guarda la salida en un archivo de log. Lo hice así para tenerlo todo bien registrado y poder comparar resultados.

Primero probé que el balanceador respondiera bien a un acceso normal, lo cual fue correcto. Luego lancé un ataque DDoS usando 100 conexiones paralelas con curl y prácticamente todas tuvieron éxito, aunque el tiempo fue muy alto. Eso me hace pensar que el servidor aguanta la carga, pero “sufre” en cuanto a disponibilidad y tiempo de respuesta.

Después usé ApacheBench para lanzar 1000 peticiones con 100 de concurrencia, y ahí sí se notó que el sistema estaba protegiendo la prueba de carga de DDoS: muchas peticiones no llegaron a completarse y aparecieron errores como “Connection reset by peer”. Esto, lejos de ser un fallo, es una buena señal, porque antes de hacer estas pruebas ya habíamos creado los iptables que limitan la cantidad de conexiones nuevas y el número de conexiones activas por IP, usando connlimit, hashlimit y otras protecciones como control de SYN floods. Así que en realidad, el sistema estaba haciendo su trabajo.

También lancé un ataque tipo Slowloris con slowhttptest, simulando muchas conexiones lentas. En ese caso, el servicio se volvió no disponible durante unos segundos, lo que indica que esta técnica sí le afecta. Pero es cierto que si el número de sockets abiertos se dispara, el sistema aún puede llegar a saturarse.

Las pruebas de inyección SQL y XSS también las pasé. En el XSS, el servidor respondió con un código 200, es decir, aceptó la petición aunque contenía un <script>, lo cual podría ser un riesgo si el backend no lo filtra correctament (Para esta parte he visto que tendría que modificar también archivos de configuración del balanceador y de las webs, debido a que iptables no es tan robusto para proteger tanto estas cargas). En cambio, la SQLi devolvió un código 000, lo cual parece indicar que la petición fue bloqueada por la regla de iptables.

Por último, hice escaneos SYN, NULL y XMAS con nmap, y me llevé la sorpresa de que hay bastantes puertos abiertos. Algunos de estos están abiertos a propósito (porque los habilité manualmente para los servicios web),pero otros están abiertos porque la prueba de nmap la hago desde mi propio ordenador host, no desde los contenedores y se muestran los puertos abiertos. Aun así, muchas respuestas aparecen como open|filtered, lo cual indica que las reglas del firewall están filtrando de forma activa este tipo de escaneos sigilosos.

Tras haber explicado las salidas voy a mostrar el script de prueba realizado y la salida:

$ ataques.sh

You, 3 hours ago | 1 author (You)

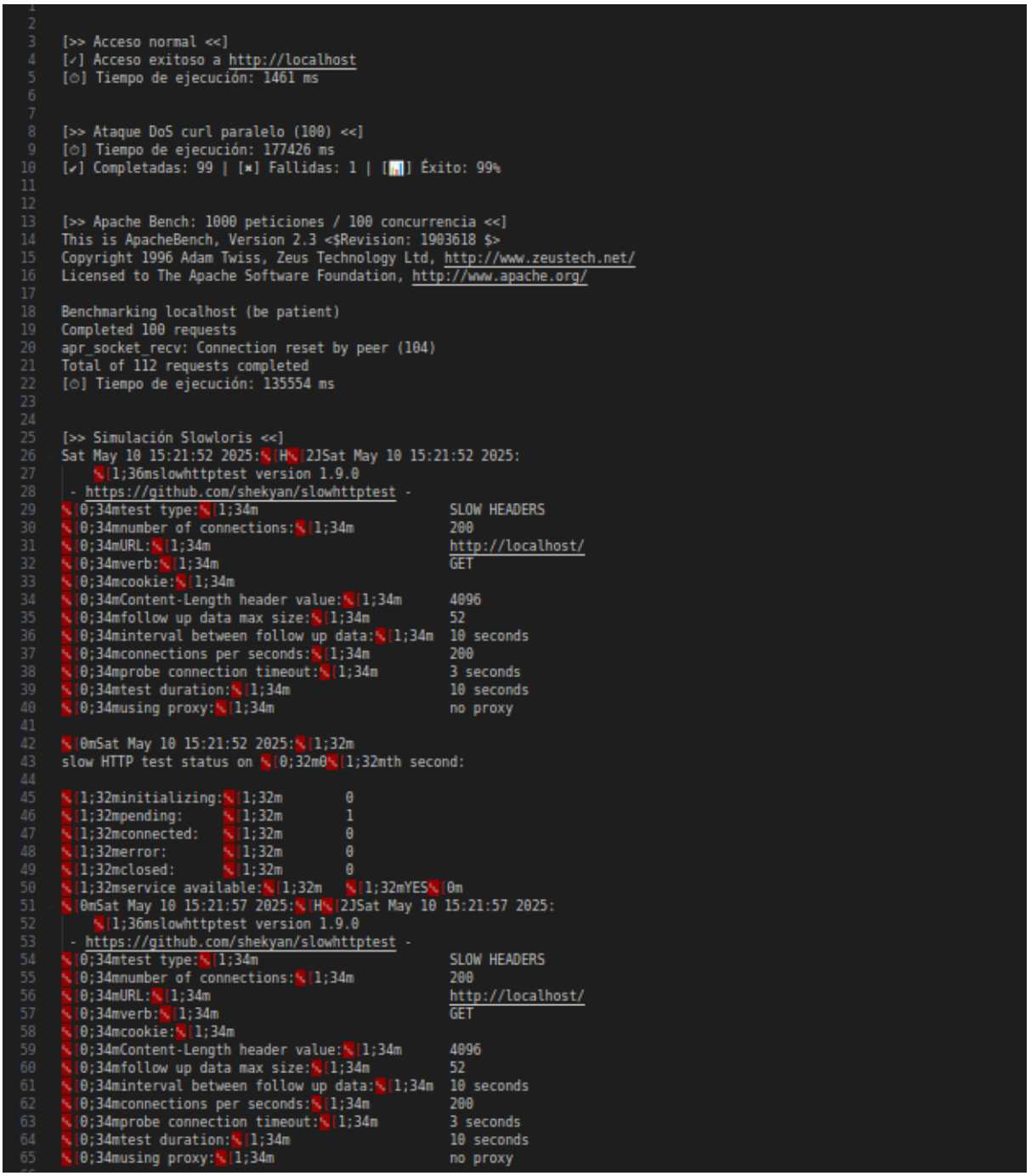
```bash
#!/bin/bash

TARGET=${1:-http://localhost}
TIPO_ATAQUE=$2
LOGFILE="resultados_ataques.txt"

[[ "$TARGET" =~ ^http ]] || TARGET="http://$TARGET"

function log() {
  echo -e "\n\n[>> $1 <<]" | tee -a "$LOGFILE"
}

function medir_tiempo() {
  local start=$(date +%s%3N)
  local cmd="$*"
  local output
  output=$(eval "$cmd" 2>&1)
  local status=$?
  local end=$(date +%s%3N)      You, 5 hours ago • Advanced finished?ç
  local duration=$((end - start))
  echo "$output" | tee -a "$LOGFILE"
  echo "[⏱] Tiempo de ejecución: ${duration} ms" | tee -a "$LOGFILE"
  return $status
}

function ataque_ddos() {
  log "Ataque DoS curl paralelo (100)"
  local completed=0 failed=0
  local start=$(date +%s%3N)

  for i in {1..100}; do
    if curl -s -o /dev/null --max-time 5 "$TARGET"; then
      ((completed++))
    else
      ((failed++))
    fi
  done

  local end=$(date +%s%3N)
  local duration=$((end - start))
  local total=$((completed + failed))
  local success_pct=$(( total > 0 ? 100 * completed / total : 0 ))

  echo "[⏱] Tiempo de ejecución: ${duration} ms" | tee -a "$LOGFILE"
  echo "[✓] Completadas: $completed | [✖] Fallidas: $failed | [📊] Éxito: $success_pct%" | tee -a "$LOGFILE"
}

function ataque_ddos_saturacion() {
  log "Apache Bench: 1000 peticiones / 100 concurrencia"
  medir_tiempo ab -n 1000 -c 100 "$TARGET/"
}

function ataque_ddos_slow() {
  log "Simulación Slowloris"
  if ! command -v slowhttptest &>/dev/null; then
    echo "[!] slowhttptest no está instalado." | tee -a "$LOGFILE"
    return
  fi
  medir_tiempo slowhttptest -c 200 -H -i 10 -r 200 -t GET -u "$TARGET/" -x 24 -p 3 -l 10
}

function ataque_sqli() {
  log "Ataque SQL Injection"
  local code=$(curl -s -o /dev/null -w "%{http_code}" "$TARGET/?id=1 UNION SELECT * FROM users WHERE '1'='1'")
  echo "[HTTP_CODE:$code]" | tee -a "$LOGFILE"
}
```

```bash

function ataque_xss() {
  log "Ataque XSS"
  local code=$(curl -s -o /dev/null -w "%{http_code}" "$TARGET/?search=<script>alert('x')</script>")
  echo "[HTTP_CODE:$code]" | tee -a "$LOGFILE"
}

function escaneo_puertos() {
  log "Escaneo SYN"
  medir_tiempo sudo nmap -sS -Pn -T4 -p- "$(echo "$TARGET" | sed 's|http[s]*://||')"
}

function escaneo_null() {
  log "Escaneo NULL"
  medir_tiempo sudo nmap -sN -Pn "$(echo "$TARGET" | sed 's|http[s]*://||')"
}

function escaneo_xmas() {
  log "Escaneo XMAS"
  medir_tiempo sudo nmap -sX -Pn "$(echo "$TARGET" | sed 's|http[s]*://||')"
}

function prueba_http() {
  log "Acceso normal"
  local start=$(date +%s%3N)
  if curl -s -o /dev/null "$TARGET"; then
    echo "[✓] Acceso exitoso a $TARGET" | tee -a "$LOGFILE"
  else
    echo "[✗] No se pudo acceder a $TARGET" | tee -a "$LOGFILE"
  fi
  local end=$(date +%s%3N)
  echo "[⏱] Tiempo de ejecución: $((end - start)) ms" | tee -a "$LOGFILE"
}

function ejecutar_todo() {
  rm -f "$LOGFILE"
  prueba_http
  ataque_ddos
  ataque_ddos_saturacion
  ataque_ddos_slow
  ataque_sqli
  ataque_xss
  escaneo_puertos
  escaneo_null
  escaneo_xmas
}

function ayuda() {
  echo "Uso: $0 <url> <ataque>"
  echo "Tipos de prueba:"
  echo "  --test         => Acceso normal"
  echo "  --ddos         => Conexiones curl simultáneas"
  echo "  --ddos-full    => ApacheBench con 1000 peticiones"      You, 5 hours ago • Advanced finished?ç
  echo "  --ddos-slow    => Simula conexión lenta tipo Slowloris"
  echo "  --sqli         => Simulación de SQL Injection"
  echo "  --xss          => Simulación de Cross Site Scripting"
  echo "  --scan         => Escaneo SYN con nmap"
  echo "  --scan-null    => Escaneo NULL con nmap"
  echo "  --scan-xmas    => Escaneo XMAS con nmap"
  echo "  --all          => Ejecuta todas las pruebas anteriores"
}

case "$TIPO_ATAQUE" in
  --test) prueba_http ;;
  --ddos) ataque_ddos ;;
  --ddos-full) ataque_ddos_saturacion ;;
  --ddos-slow) ataque_ddos_slow ;;
  --sqli) ataque_sqli ;;
  --xss) ataque_xss ;;
  --scan) escaneo_puertos ;;
  --scan-null) escaneo_null ;;
  --scan-xmas) escaneo_xmas ;;
  --all) ejecutar_todo ;;
  *) ayuda ;;
esac
```

El script se llama ataques.sh, ejecutamos con ./ataques.sh <url> <tipo>.

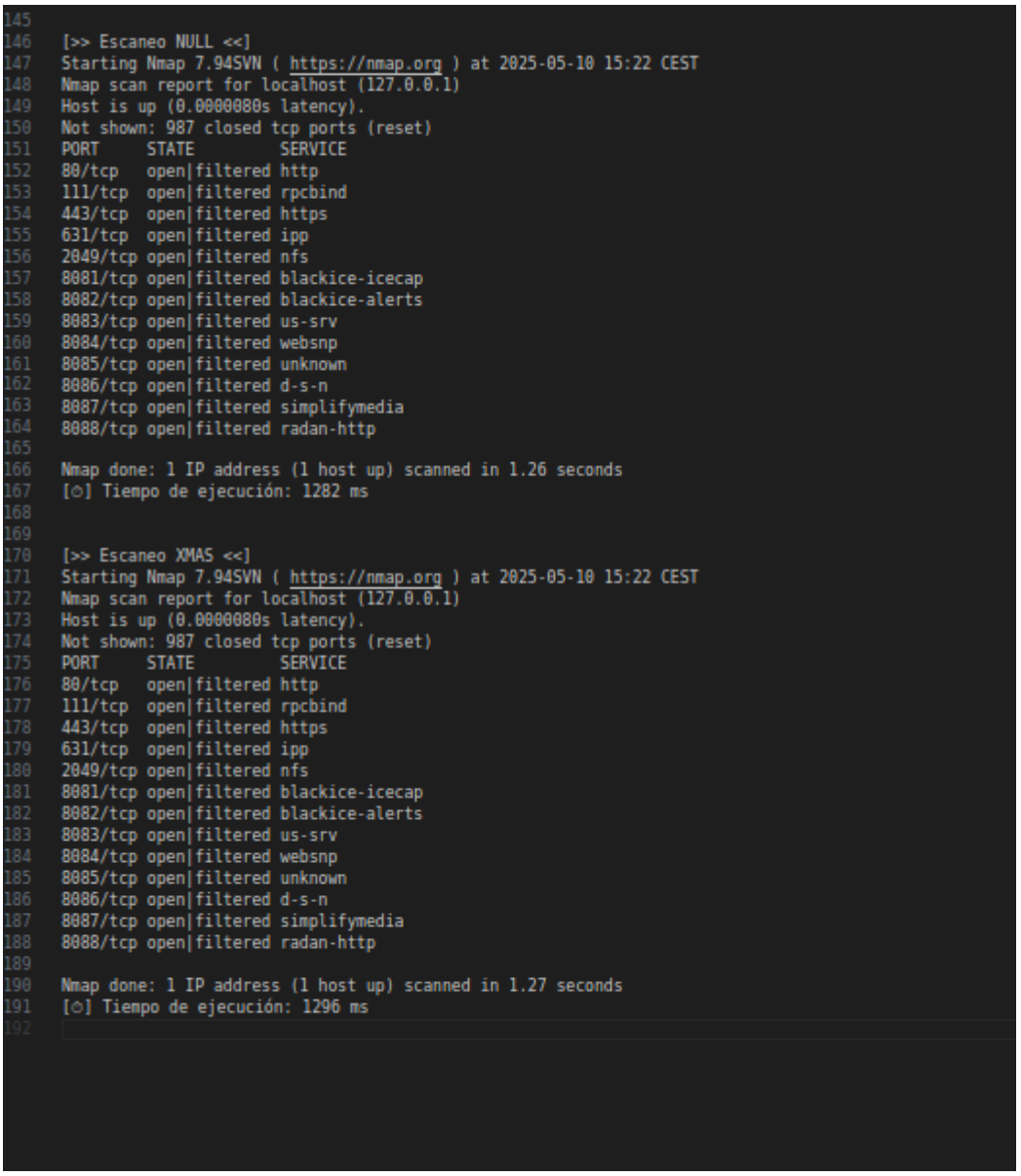
La salida de dicho script, cuando usamos el tipo --all, se crea un fichero llamado resultados_ataques.txt, con los otros tipos solamente se muestra por pantalla.

Mostramos la salida de dicho fichero:



```
[>> Acceso normal <<]
[✓] Acceso exitoso a http://localhost
[⏱] Tiempo de ejecución: 1461 ms


[>> Ataque DoS curl paralelo (100) <<]
[⏱] Tiempo de ejecución: 177426 ms
[✓] Completadas: 99 | [✖] Fallidas: 1 | [📊] Éxito: 99%


[>> Apache Bench: 1000 peticiones / 100 concurrencia <<]
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
apr_socket_recv: Connection reset by peer (104)
Total of 112 requests completed
[⏱] Tiempo de ejecución: 135554 ms


[>> Simulación Slowloris <<]
Sat May 10 15:21:52 2025:␛[H␛[2JSat May 10 15:21:52 2025:
    ␛[1;36mslowhttptest version 1.9.0
 - https://github.com/shekyan/slowhttptest -
␛[0;34mtest type:␛[1;34m                          SLOW HEADERS
␛[0;34mnumber of connections:␛[1;34m              200
␛[0;34mURL:␛[1;34m                                http://localhost/
␛[0;34mverb:␛[1;34m                               GET
␛[0;34mcookie:␛[1;34m
␛[0;34mContent-Length header value:␛[1;34m        4096
␛[0;34mfollow up data max size:␛[1;34m            52
␛[0;34minterval between follow up data:␛[1;34m    10 seconds
␛[0;34mconnections per seconds:␛[1;34m            200
␛[0;34mprobe connection timeout:␛[1;34m           3 seconds
␛[0;34mtest duration:␛[1;34m                      10 seconds
␛[0;34musing proxy:␛[1;34m                        no proxy

␛[0mSat May 10 15:21:52 2025:␛[1;32m
slow HTTP test status on ␛[0;32m0␛[1;32mth second:

␛[1;32minitializing:␛[1;32m        0
␛[1;32mpending:     ␛[1;32m        1
␛[1;32mconnected:   ␛[1;32m        0
␛[1;32merror:       ␛[1;32m        0
␛[1;32mclosed:      ␛[1;32m        0
␛[1;32mservice available:␛[1;32m   ␛[1;32mYES␛[0m
␛[0mSat May 10 15:21:57 2025:␛[H␛[2JSat May 10 15:21:57 2025:
    ␛[1;36mslowhttptest version 1.9.0
 - https://github.com/shekyan/slowhttptest -
␛[0;34mtest type:␛[1;34m                          SLOW HEADERS
␛[0;34mnumber of connections:␛[1;34m              200
␛[0;34mURL:␛[1;34m                                http://localhost/
␛[0;34mverb:␛[1;34m                               GET
␛[0;34mcookie:␛[1;34m
␛[0;34mContent-Length header value:␛[1;34m        4096
␛[0;34mfollow up data max size:␛[1;34m            52
␛[0;34minterval between follow up data:␛[1;34m    10 seconds
␛[0;34mconnections per seconds:␛[1;34m            200
␛[0;34mprobe connection timeout:␛[1;34m           3 seconds
␛[0;34mtest duration:␛[1;34m                      10 seconds
␛[0;34musing proxy:␛[1;34m                        no proxy
```

```
[0mSat May 10 15:21:57 2025:[1;32m
slow HTTP test status on [0;32m5[1;32mth second:

[1;32minitializing:[1;32m        0
[1;32mpending:     [1;32m        0
[1;32mconnected:   [1;32m        200
[1;32merror:       [1;32m        0
[1;32mclosed:      [1;32m        0
[1;32mservice available:[1;32m   [1;31mNO[0m
[0mSat May 10 15:22:02 2025:[H[2JSat May 10 15:22:02 2025:
    [1;36mslowhttptest version 1.9.0
 - https://github.com/shekyan/slowhttptest -
[0;34mtest type:[1;34m                          SLOW HEADERS
[0;34mnumber of connections:[1;34m              200
[0;34mURL:[1;34m                                http://localhost/
[0;34mverb:[1;34m                               GET
[0;34mcookie:[1;34m
[0;34mContent-Length header value:[1;34m        4096
[0;34mfollow up data max size:[1;34m            52
[0;34minterval between follow up data:[1;34m    10 seconds
[0;34mconnections per seconds:[1;34m            200
[0;34mprobe connection timeout:[1;34m           3 seconds
[0;34mtest duration:[1;34m                      10 seconds
[0;34musing proxy:[1;34m                        no proxy

[0mSat May 10 15:22:02 2025:[1;32m
slow HTTP test status on [0;32m10[1;32mth second:

[1;32minitializing:[1;32m        0
[1;32mpending:     [1;32m        0
[1;32mconnected:   [1;32m        200
[1;32merror:       [1;32m        0
[1;32mclosed:      [1;32m        0
[1;32mservice available:[1;32m   [1;31mNO[0m
[0mSat May 10 15:22:03 2025:[0;36m
Test ended on 11th second
Exit status:[1;36m Hit test time limit
[0m
[⏲] Tiempo de ejecución: 11061 ms


[>> Ataque SQL Injection <<]
[HTTP_CODE:000]


[>> Ataque XSS <<]
[HTTP_CODE:200]


[>> Escaneo SYN <<]
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-10 15:22 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000070s latency).
Not shown: 65516 closed tcp ports (reset)
PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
443/tcp   open  https
631/tcp   open  ipp
2049/tcp  open  nfs
8081/tcp  open  blackice-icecap
8082/tcp  open  blackice-alerts
8083/tcp  open  us-srv
8084/tcp  open  websnp
8085/tcp  open  unknown
8086/tcp  open  d-s-n
8087/tcp  open  simplifymedia
8088/tcp  open  radan-http
29754/tcp open  unknown
41815/tcp open  unknown
43095/tcp open  unknown
44039/tcp open  unknown
54837/tcp open  unknown
59617/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.90 seconds
[⏲] Tiempo de ejecución: 923 ms
```



```
[>> Escaneo NULL <<]
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-10 15:22 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000080s latency).
Not shown: 987 closed tcp ports (reset)
PORT     STATE         SERVICE
80/tcp   open|filtered http
111/tcp  open|filtered rpcbind
443/tcp  open|filtered https
631/tcp  open|filtered ipp
2049/tcp open|filtered nfs
8081/tcp open|filtered blackice-icecap
8082/tcp open|filtered blackice-alerts
8083/tcp open|filtered us-srv
8084/tcp open|filtered websnp
8085/tcp open|filtered unknown
8086/tcp open|filtered d-s-n
8087/tcp open|filtered simplifymedia
8088/tcp open|filtered radan-http

Nmap done: 1 IP address (1 host up) scanned in 1.26 seconds
[⏲] Tiempo de ejecución: 1282 ms


[>> Escaneo XMAS <<]
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-10 15:22 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000080s latency).
Not shown: 987 closed tcp ports (reset)
PORT     STATE         SERVICE
80/tcp   open|filtered http
111/tcp  open|filtered rpcbind
443/tcp  open|filtered https
631/tcp  open|filtered ipp
2049/tcp open|filtered nfs
8081/tcp open|filtered blackice-icecap
8082/tcp open|filtered blackice-alerts
8083/tcp open|filtered us-srv
8084/tcp open|filtered websnp
8085/tcp open|filtered unknown
8086/tcp open|filtered d-s-n
8087/tcp open|filtered simplifymedia
8088/tcp open|filtered radan-http

Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds
[⏲] Tiempo de ejecución: 1296 ms
```

Prueba de ráfagas:
https://drive.google.com/file/d/1fVOX7MpQdsxtdNlri5UMkD6zUdzaANqZ/view?usp=sharing

3. Uso de la IA Generativa

Enlace a la conversación:
https://chatgpt.com/share/681fb897-8538-800d-b287-e05d53a52cbe

Como he comentado en las anteriores prácticas, hay que utilizar la IA como herramienta para facilitar el trabajo y aumentar así la eficiencia. Además, hay que usarlo para aprender, no solo para realizar copia y pega.

La IA la he utilizado durante toda la práctica, simplemente para facilitar la creación de los distintos ficheros o modificaciones de ellos. También preguntarle dudas respecto a conceptos que no comprendía.

Como no pude asistir a la presentación de la práctica hice algunas preguntas más teóricas sobre el funcionamiento e implementación de lo que se nos solicitaba. Entre ellos destaco el funcionamiento de IPTABLES, la denegación implícita (política por defecto), además de las distintas reglas creadas en la parte básica de la práctica. Otra de las preguntas más teoricas fue sobre el funcionamiento del módulo recent para el bloqueo temporal/permanente, por otro lado la parte de implementación de TARPIT (que no lo implementé al final), no solo fue preguntar por recent, también preguntar su diferencia frente a hashlimit y keep-alive en http.

Por otro lado, también tuve varios problemas en la parte de la tarea A3, al comienzo eran problemas debido a que no caí en que realmente estaba implementado reglas propias a los servidores webs en vez del balanceador de carga, por lo que no me funcionaban bien. Por ello, posteriormente los implementé al balanceador de carga.

En dichas reglas para el balanceador, tuve varios problemas, incluso usando la IA, para implementar las diferentes reglas. Principalmente eran errores por el orden de las diferentes reglas de iptables, no por la generación que había implementado la IA. Sin embargo, hubo muchos problemas de contexto para la IA, por más que se le explicaba, aunque también hay que indicar que muchos problemas que me estaban dando (sobretodo respecto a realizar yo pruebas de ataques), era porque no valía solamente con implementar reglas de IPTABLES o porque no eran muy específicas algunas de ellas. Posteriormente me lo implementó correctamente la IA a base de prueba y error.

Por ello es importante recalcar que solamente es una herramienta que nos permite acelerar el proceso del desarrollo y poder comprender conceptos sin tener que estar revisando cientos de páginas web.

Tiempo de desarrollo:

Realmente implementarlo he tardado unas 2-3 horas, donde he dedicado más tiempo ha sido en la A3, para la cual le he echado otras 3 horas solucionando errores. La documentación la he ido haciendo a la par, por ello explico en la A3 que me parecía interesante comentar todo el proceso de pruebas y errores hasta llegar a la implementación correcta.

Por lo tanto, **tiempo en desarrollo : 6 horas.**