UNIVERSIDAD DE GRANADA

Servidores Web de Altas Prestaciones

Práctica 4: Seguridad (Cortafuegos)

Miguel Torres Alonso

Mayo 2024

Índice

1. Tareas Básicas - B1. Preparación del Entorno de Trabajo

1.1. Crear y preparar directorios específicos para los archivos de configuración de IPTABLES y certificados SSL previamente generados

Seguimos los pasos de creación de carpetas como se nos indica en el guión y en la figura 1 se muestra la estructura del entorno de trabajo.

```
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P4$ tree
.
├── P4-mtorres26-apache
│   ├── DockerFileApacheP3_mtorres26
│   ├── mtorres26-apache-ssl.conf
│   └── P4-mtorres26-iptables-web
├── P4-mtorres26-certificados
│   ├── certificado_mtorres26.crt
│   └── certificado_mtorres26.key
├── P4-mtorres26-nginx
│   ├── DockerFileNginxP3_mtorres26
│   └── mtorres26-nginx-ssl.conf
└── web_mtorres26
    └── index.php

5 directories, 7 files
```

Figura 1: Estructura del entorno de trabajo

2. Tareas Básicas - B2. Creación y Configuración de Scripts IPTABLES

2.1. Desarrollar y escribir un script mtorres26-iptables-web.sh que establecerá las reglas de IPTABLES en los servidores web.

Este script debe:

- Establecer políticas por defecto para rechazar todo tráfico no explícitamente permitido.
- Permitir conexiones entrantes y salientes específicas necesarias para la operación normal de los servidores web.
- Asegurar que las conexiones entre el balanceador de carga y los servidores web estén adecuadamente configuradas para permitir solo tráfico HTTP y HTTPS.

En el guión se nos proporcionan los comandos necesarios para la realización de este apartado, por lo tanto el script *mtorres26-iptables-web.sh* se nos queda de la siguiente manera:

```
1 # Denegacion implicita de todo el trafico
2 iptables -P INPUT DROP
3 iptables -P OUTPUT DROP
4 iptables -P FORWARD DROP
```

```
5
6  # Aceptar paquetes que pertenezcan a una conexion existente o relacionada
7  iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
8
9  # Permitir conexiones nuevas o ya establecidas al trafico saliente
10 iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
11
12 # Permitir conexiones loopback
13 iptables -A INPUT -i lo -j ACCEPT
14 iptables -A OUTPUT -o lo -j ACCEPT
15
16 # Permitir el trafico HTTP y HTTPS
17 iptables -A INPUT -p tcp -s 192.168.10.50 --dport 80 -j ACCEPT
18 iptables -A INPUT -p tcp -s 192.168.10.50 --dport 443 -j ACCEPT
```

Listing 1: Script con los comandos de iptables.

3. Tareas Básicas - B3. Implementación de Scripts IPTABLES en Docker

3.1. Integrar el script IPTABLES en la configuración de Docker de los servidores web Apache

Esto implica:

- Modificar los Dockerfiles para incluir y ejecutar el script IPTABLES al iniciar los contenedores.
- Asegurar que los scripts tienen los permisos adecuados para ejecutarse y modificar las reglas de IPTABLES dentro de los contenedores.

De nuevo seguimos los pasos que nos vienen en el guión, sin olvidarnos de darle permisos de ejecución al script a continuación.

```
1 #!/bin/bash
2 # Ejecuta el script de iptables
3 ./mtorres26-iptables-web.sh
4 # Luego, ejecuta el comando principal del contenedor
5 exec "$@"
```

Listing 2: Script que se ejecutará en cada servicio web.

Una vez elaborado el script *entrypoint.sh*, ajustamos el Dockerfile de la práctica anterior añadiendole lo que se nos indica en el guión.

```
1  #Utilizar una imagen base de debian ligera como en la P1
2  FROM debian:buster-slim
3
4  # Instalar Apache, PHP, herramientas de red e iptables
5  RUN apt-get update && \
6      apt-get install -y apache2 php libapache2-mod-php iptables net-tools iputils-ping && \
7      apt-get clean && rm -rf /var/lib/apt/lists/*
8
9  #Instalar modulo y habilitar sitio SSL por defecto y crear directorio para certificados
10 RUN a2enmod ssl && a2ensite default-ssl && mkdir /etc/apache2/ssl
11
12 # Copiar script de entrada entrypoint.sh y script de reglas iptables
13 COPY ./P4-mtorres26-iptables-web/entrypoint.sh /entrypoint.sh
14 COPY ./P4-mtorres26-iptables-web/mtorres26-iptables-web.sh /mtorres26-iptables-web.sh
15
16 # Configurar los permisos adecuados
17 RUN chmod +x /entrypoint.sh /mtorres26-iptables-web.sh
18
```

```
# Incluir la configuracion SSL
COPY mtorres26-apache-ssl.conf /etc/apache2/sites-available/mtorres26-apache-ssl.conf

# Activamos nuestra configuracion
RUN a2ensite mtorres26-apache-ssl

# Exponer el puerto HTTPS y HTTP
EXPOSE 443
EXPOSE 80

# Configurar el script de entrada
ENTRYPOINT ["/entrypoint.sh", "apachectl", "-D", "FOREGROUND"]
```

Listing 3: Dockerfile de los servicios web Apache.

4. Tareas Básicas - B4. Configuración de Docker Compose

4.1. Modificar y adaptar el archivo docker-compose.yml de la práctica anterior para incluir los cambios necesarios que permitan la ejecución de IPTABLES dentro de los contenedores de Apache y Nginx.

Esto incluirá:

- La configuración para montar los scripts y directorios necesarios dentro de los contenedores.
- Asegurar que los contenedores tienen las capacidades de red necesarias (CAP_NET_ADMIN) para modificar IPTABLES.

Lo único que debemos modificar con respecto del *docker-compose* de la práctica anterior es la capacidad de administración de red de los servicios web, por tanto incluimos lo que se nos proporciona en el guión *'cap_add: - NET_ADMIN'* y sustituimos tags de 'P3' a 'P4'.

Cada servicio web se nos quedaría como se refleja en el siguiente código:

```
web1:
    image: mtorres26-apache-image:p4
    container_name: web1
    build:
      context: ./P4-mtorres26-apache
      dockerfile: DockerFileApacheP4_mtorres26
    volumes:
      - ./web_mtorres26:/var/www/html
      - ./P4-mtorres26-certificados:/etc/apache2/ssl
      - ./P4-mtorres26-apache/mtorres26-apache-ssl.conf:/etc/apache2/sites-available/
    mtorres26-apache-ssl.conf
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2
    cap_add:
      - NET_ADMIN
```

Listing 4: Código de cada servicio web del docker-compose.

5. Tareas Básicas - B5. Verificación y Pruebas

5.1. Ejecutar y verificar el entorno configurado.

Levantamos los servicios y accedemos a la url *https://localhost*, lo cual nos llevará a la página que nos proporciona el balanceador de carga y vemos como en la figura 2 que cumple su función de manera correcta.

Figura 2: Acceso al balanceador de carga.

Sin embargo, si solicitamos acceder a cualquiera de las IP's de los servicios web, por ejemplo, 192.168.10.6 (figura 3), la página no se nos muestra y al cabo de un tiempo devuelve error por tiempo de espera agotado, lo cual significa que funciona correctamente lo que hemos hecho.

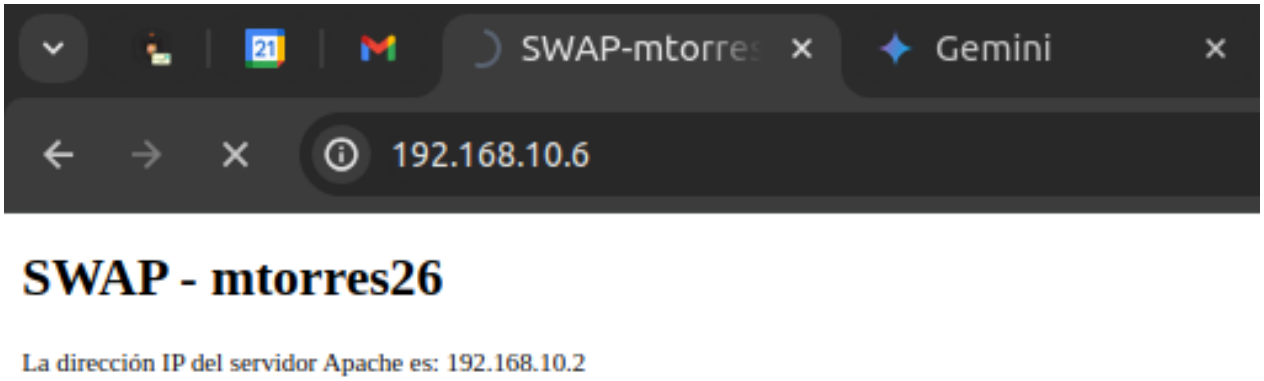

Figura 3: Intento denegado de acceso a un servicio web de manera independiente al balanceador.

6. Tareas Avanzadas - A1. Definir e implementar políticas de seguridad en el balanceador de carga.

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.

6.0.1. Pregunta realizada a la IA:

'¿Cómo puedo limitar el número de conexiones simultáneas en un balanceador de carga Nginx?'

6.0.2. Respuesta de la IA:

```
limit_conn_zone my_limit_zone 10;

server {
  listen 80;
  server_name example.com;

  location / {
    limit_conn my_limit_zone;
  }
}
```

Listing 5: Respuesta de la IA para añadir más políticas de seguridad en el balanceador de carga.

6.0.3. Análisis detallado:

Lo que proporciona la IA está a medias porque le faltan directivas, por lo tanto al montar el servicio nos da error. No nos sirve, debemos buscar en la documentación oficial de Nginx. Esta nos dice que podemos limitar el número de conexiones simultáneas con la variable *limit_conn*.

```
limit_conn_zone $server_name zone=perserver:10m;
limit_conn perserver 10;
```

Donde *$server_name* indica el nombre del servidor que acepta la solicitud. Debemos añadir el comando anterior en el server de HTTPS del archivo de configuración de Nginx. Esa zona del archivo de configuración nos queda como observamos en el código siguiente.

```
limit_conn_zone $server_name zone=perserver:10m;

server{
    # puerto 443 para https
    listen 443 ssl;
    ssl_certificate /etc/nginx/ssl/certificado_mtorres26.crt;
    ssl_certificate_key /etc/nginx/ssl/certificado_mtorres26.key;

    limit_conn perserver 10; # Limite de conexiones en el servidor

    location / {
        proxy_pass http://backend_mtorres26;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
    }
```

Listing 6: Modificación al archivo de configuración de Nginx.

Ahora, pedimos a la IA que intente usar módulo string para mitigar ataques de inyección SQL o XSS a través de las peticiones HTTP.

6.0.4. Pregunta realizada a la IA:

'¿Cómo puedo usar módulo string en Nginx para mitigar ataques de inyección SQL o XSS a través de las peticiones HTTP.?'

6.0.5. Respuesta de la IA:

```
# Bloquear palabras clave comunes de inyeccion SQL
location / {
  if ($request_uri ~* "(INSERT|UPDATE|DELETE|SELECT|EXEC|SCRIPT)") {
    return 400;
  }
}

# Bloquear fragmentos de codigo JavaScript malicioso
location / {
  if ($http_user_agent ~* "(<script|onload|onmouseover|onmouseout|javascript:|eval)") {
    return 400;
  }
}
```

6.0.6. Análisis detallado:

Una manera de realizar este apartado es como nos dice la IA, añadimos los comandos al archivo de configuración y probamos (figura 4 y figura 5) que nos devuelva error 400 al inyectar en la URI código SQL o JavaScript.

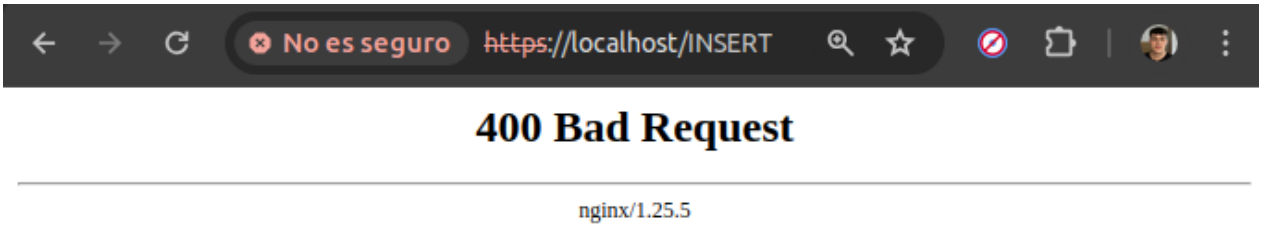

Figura 4: Comprobación de inyección de código SQL.

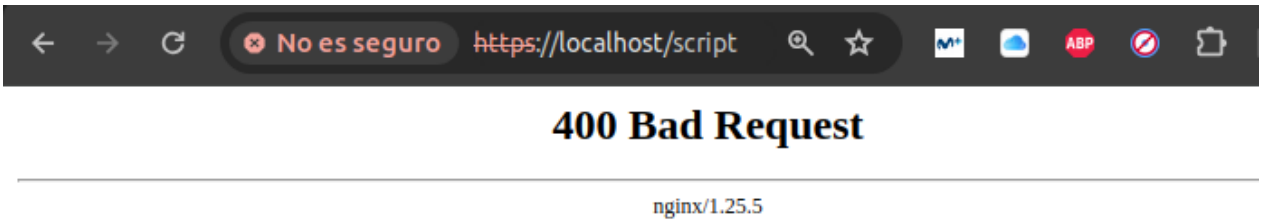

Figura 5: Comprobación de inyección de código JavaScript.

7. Tareas Avanzadas - A2. Configuración Avanzada de IPTABLES para DDoS

7.1. 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.

7.1.1. Pregunta realizada a la IA:

'Genera código iptables para limitar la tasa de conexiones nuevas por IP.'

7.1.2. Respuesta de la IA:

```
# Limitar tasa de nuevas conexiones por IP (ventana deslizante)
iptables -N limit_conn  # Crear una cadena para la limitacion
iptables -A INPUT -m state --state NEW -j limit_conn  # Marcar nuevas conexiones
iptables -A limit_conn -m connlimit --connlimit-above 10 --connlimit-mask 32 --connlimit-
    mode count/s --connlimit-unit 1 -j ACCEPT
iptables -A limit_conn -j DROP  # Bloquear despues de superar el limite
```

7.1.3. Análisis detallado:

Lo que proporciona la IA es válido, pues va marcando las conexiones nuevas por IP y al superar el límite las rechaza. Además, podemos añadir reglas iptables para bloquear ciertos tipos de paquetes para decrementar las posibilidades de un ataque DDoS aún más:

- Bloquear paquetes nuevos que no sean SYN ni pertenezcan a una conexión existente.
- Bloquear valores anormales de MSS.
- Bloquear tráfico por cantidad de conexiones.

Las nuevas reglas iptables quedarían de la siguiente manera:

```
# Limitar tasa de nuevas conexiones por IP (ventana deslizante)
iptables -N limit_conn  # Crear una cadena para la limitacion
iptables -A INPUT -m state --state NEW -j limit_conn  # Marcar nuevas conexiones
iptables -A limit_conn -m connlimit --connlimit-above 10 --connlimit-mask 32 --connlimit-
    mode count/s --connlimit-unit 1 -j ACCEPT
iptables -A limit_conn -j DROP  # Bloquear despues de superar el limite

# Bloquear paquetes nuevos que no sean SYN ni pertenezcan a una conexion existente
iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate NEW -j DROP

# Bloquear valores anormales de MSS
iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW -m tcpmss ! --mss
    536:65535 -j DROP

# Bloquear trafico por cantidad de conexiones
iptables -A INPUT -p tcp -m connlimit --connlimit-above 5 -j REJECT --reject-with tcp-reset
```