UNIVERSIDAD DE GRANADA

Práctica 4: Asegurar una granja web

Autor
Juan Manuel Castillo Nievas

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 30 de julio de 2020

Índice

1. Crear e instalar un certificado SSL autofirmado para configurar el acceso HTTPS al servidor

Para crear el certificado SSL primero hay que activar el módulo SSL de Apache. En la máquina M1 he ejecutado los comandos que se muestran en la Figura 1

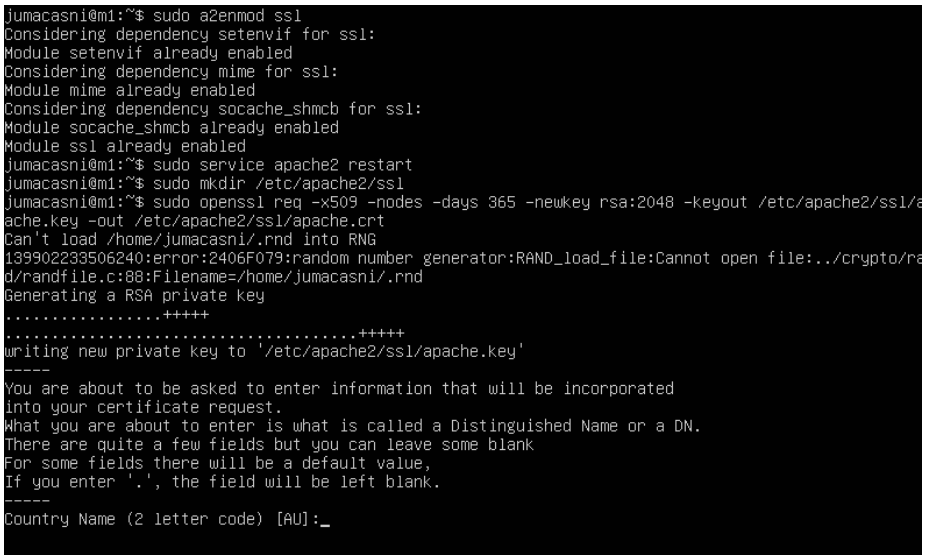

Figura 1: Activar módulo SSL de Apache

A continuación se nos pedirán unos datos para configurar el certificado del dominio. Los datos que he puesto yo se muestran en la Figura 2.

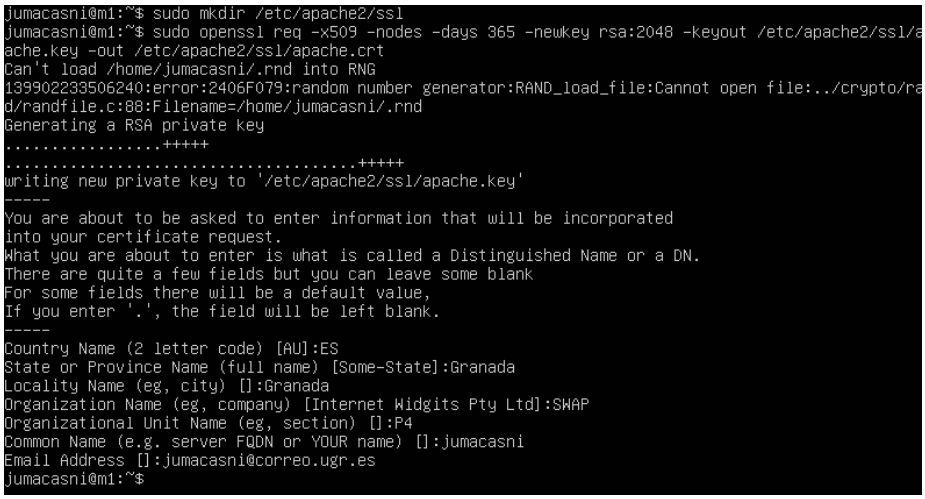

Figura 2: Configurar certificado del dominio

Ahor hay que indicar la ruta a los certificados en la configuración. Para ello, hay que editar el archivo de configuración de Apache **default.ssl**. Tal y como muestro en la Figura 3, hay que añadir las siguientes rutas debajo de **SSLEngine on**:

- SSLCertificateFile /etc/apache2/ssl/apache.crt
- SSLCertificateKeyFile /etc/apache2/ssl/apache.key

Y una vez añadidas las rutas se activa **default-ssl** y se reinicia apache (Figura 4).

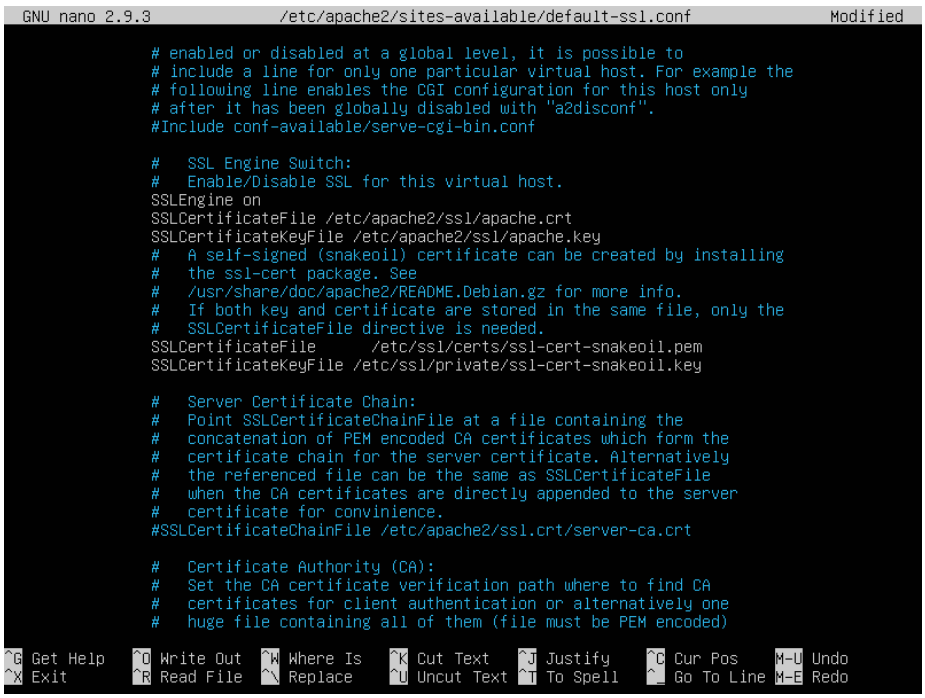

Figura 3: Indicación de la ruta a los certificados

```
jumacasni@m1:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
jumacasni@m1:~$ sudo service apache2 reload
jumacasni@m1:~$ _
```

Figura 4: Activación de **default-ssl**

Desde **Google Chrome** he accedido a `https://192.168.56.102` que es la IP de M1, y me muestra el certificado de la Figura **??**, que sólo pone el nombre de la máquina virtual. Sin embargo, una vez hecha la tarea opcional, al acceder al balanceador (`https://192.168.56.105`) me muestra el certificado que hemos creado e instalado (Figura 6)

Desde M1 he hecho una petición con *curl -k http://192.168.56.102* y *curl -k https://192.168.56.102* (Figura 7), para asegurar que acepta el tráfico tanto de **HTTP** como **HTTPS**.

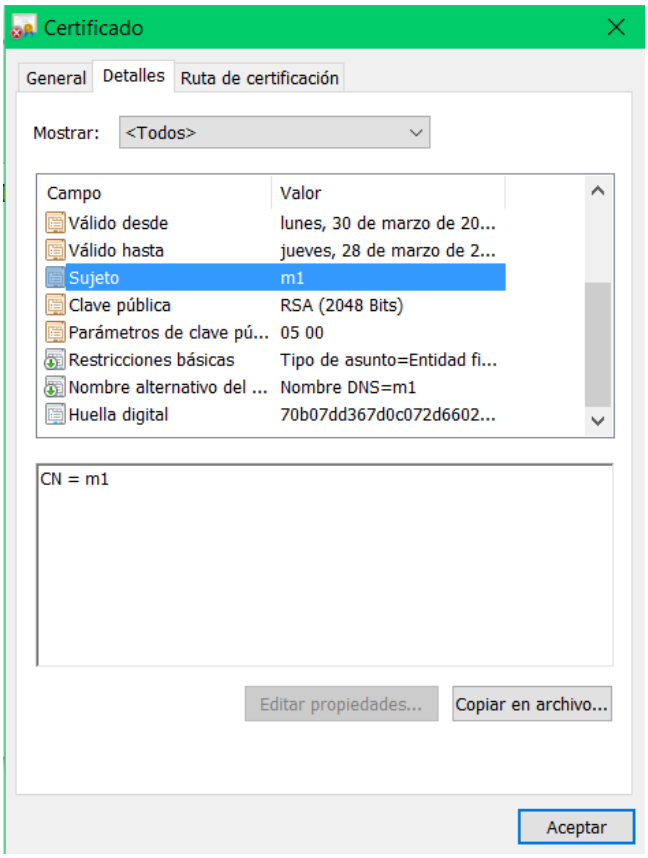

Figura 5: Certificado desde **Google Chrome** accediendo a M1

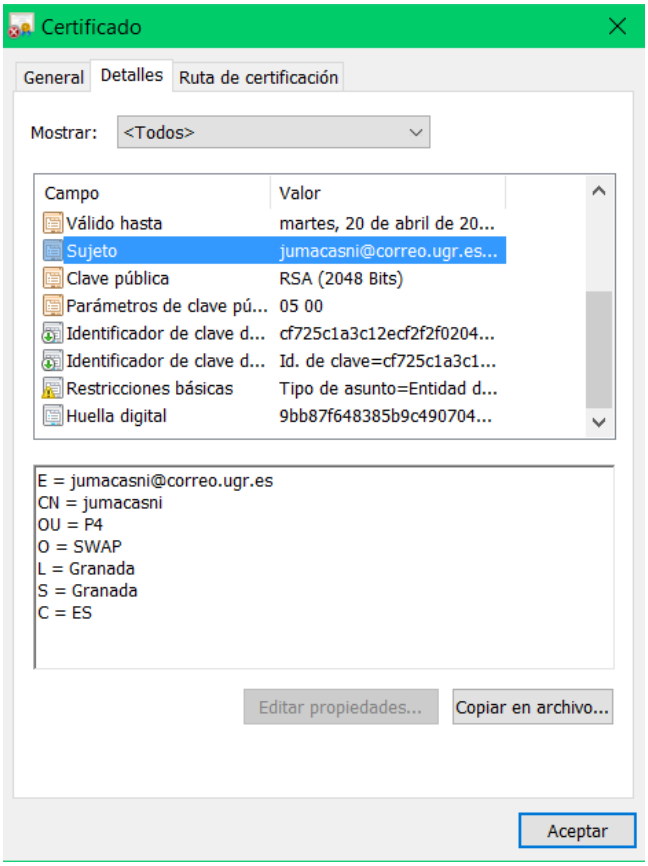

Figura 6: Certificado desde **Google Chrome** accediendo al balanceador

Figura 7: Peticiones **HTTP** y **HTTPS**

2. Primera tarea opcional: configurar nginx para aceptar y balancear correctamente el tráfico HTTP y HTTPS

Para configurar el balanceador para que acepte el tráfico, primero voy a copiar los certificados desde M1 a M2 y M3 con **scp**, tal y como muestro en la Figura 8.

```
jumacasni@m1:~$ cd /etc/apache2/ssl/
jumacasni@m1:/etc/apache2/ssl$ sudo scp apache.crt jumacasni@192.168.56.103:/home/jumacasni/apache.
rt
jumacasni@192.168.56.103's password:
apache.crt                                    100% 1432    83.0KB/s   00:00
jumacasni@m1:/etc/apache2/ssl$ sudo scp apache.key jumacasni@192.168.56.103:/home/jumacasni/apache.
ey
jumacasni@192.168.56.103's password:
apache.key                                    100% 1708     2.5MB/s   00:00
jumacasni@m1:/etc/apache2/ssl$ sudo scp apache.crt jumacasni@192.168.56.105:/home/jumacasni/apache.
rt
The authenticity of host '192.168.56.105 (192.168.56.105)' can't be established.
ECDSA key fingerprint is SHA256:SgwBcL8HIR/4VlgDZUlPXweTabFn7E1L8wJ8z/Uomi4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.105' (ECDSA) to the list of known hosts.
jumacasni@192.168.56.105's password:
apache.crt                                    100% 1432     1.8MB/s   00:00
jumacasni@m1:/etc/apache2/ssl$ sudo scp apache.key jumacasni@192.168.56.105:/home/jumacasni/apache.
ey
jumacasni@192.168.56.105's password:
apache.key                                    100% 1708     2.1MB/s   00:00
jumacasni@m1:/etc/apache2/ssl$
```

Figura 8: Copia de certificados desde M1 a M2 y M3

En segundo lugar, en M2 voy a hacer lo mismo que hice en M1: crear el directorio **/etc/apache2/ssl**, copiar los certificados, configurar **default-ssl**, activarlo y reiniciar apache (Figuras 9 y 10).

```
jumacasni@m2:~$ sudo mkdir /etc/apache2/ssl
[sudo] password for jumacasni:
jumacasni@m2:~$ sudo mv /home/jumacasni/apache* /etc/apache2/ssl
jumacasni@m2:~$ sudo a2enmod ssl & sudo service apache2 restart
[1] 14001
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certific
tes.
To activate the new configuration, you need to run:
  systemctl restart apache2
[1]+  Done                    sudo a2enmod ssl
jumacasni@m2:~$ _
```

Figura 9: Copia de los certificados en M2

```
                # error, crit, alert, emerg.
                # It is also possible to configure the loglevel for particular
                # modules, e.g.
                #LogLevel info ssl:warn

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                # For most configuration files from conf-available/, which are
                # enabled or disabled at a global level, it is possible to
                # include a line for only one particular virtual host. For example the
                # following line enables the CGI configuration for this host only
                # after it has been globally disabled with "a2disconf".
                #Include conf-available/serve-cgi-bin.conf

                #   SSL Engine Switch:
                #   Enable/Disable SSL for this virtual host.
                SSLEngine on
                SSLCertificateFile /etc/apache2/ssl/apache.crt
                SSLCertificateKeyFile /etc/apache2/ssl/apache.key
                #   A self-signed (snakeoil) certificate can be created by installing
                #   the ssl-cert package. See
                #   /usr/share/doc/apache2/README.Debian.gz for more info.
                #   If both key and certificate are stored in the same file, only the
                #   SSLCertificateFile directive is needed.


jumacasni@m2:~$ sudo a2ensite default.ssl
ERROR: Site default.ssl does not exist!
jumacasni@m2:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
jumacasni@m2:~$ sudo service apache2 reload
```

Figura 10: Activación de **default-ssl** en M2

En último lugar, en el balanceador M3 he añadido un nuevo server en la configuración de **nginx**, tal y como muestro en la Figura 11. Como resultado obtenemos que se pueden hacer peticiones **HTTPS** al balanceador desde M4 y se obtiene el resultado deseado (Figura 12).

```
jumacasni@m3:~$ cat /etc/nginx/conf.d/default.conf
upstream servidoresSWAP {
        server 192.168.56.102;
        server 192.168.56.103;
}

server {
        listen 80;
        listen 443 ssl;
        ssl_on;
        ssl_certificate /home/jumacasni/apache.crt;
        ssl_certificate_key /home/jumacasni/apache.key;
        server_name balanceador;

        access_log /var/log/nginx/balanceador.access.log;
        error_log /var/log/nginx/balanceador.error.log;
        root /var/www/;

        location /
        {
                proxy_pass http://servidoresSWAP;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
        }
}
```

Figura 11: Nuevo server en la configuración del balanceador M3

```
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M1!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M2!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M1!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M2!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
```

Figura 12: Peticiones al balanceador desde M4

3. Configurar las reglas del cortafuegos con IPTABLES para permitir el acceso por puertos HTTP y HTTPS

Para este apartado, he creado el archivo *cortafuegos.sh* en la máquina M1 con la configuración mostrada en la Figura 13. Las reglas 5 y 6 son las que hacen que permiten el acceso por **HTTP** y **HTTPS**.

```
jumacasni@m1:~$ cat cortafuegos.sh
#!/bin/bash

# 1. Eliminar todas las reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# 2. Denegar todo el tráfico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# 3. Permitir cualquier acceso desde localhost (lo)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# 4. Abrir el puerto 22 para permitir acceso por SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# 5. Permitir el tráfico por el puerto 80 (HTTP)
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

# 6. Permitir el tráfico por el puerto 443 (HTTPS)
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT


iptables -L -n -v
jumacasni@m1:~$
```

Figura 13: Cortafuegos en M1

Para hacer que este script se ejecute cada vez que se arranque el sistema, he instalado el paquete **iptables-persistent**. Es una herramienta que sirve para hacer las reglas de iptables persistentes al reiniciar el sistema. Se deben poner las reglas en el archivo **/etc/iptables/rules.v4**, y posteriormente utilizar el comando **iptables-save**, tal y como muestro en la Figura 14.

Como resultado obtenemos que desde M4 podemos hacer peticiones tanto desde **HTTP** como de **HTTPS** a través del balanceador (Figura 15)

```
jumacasni@m1:~$ cat /etc/iptables/rules.v4
# 1. Eliminar todas las reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# 2. Denegar todo el tráfico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# 3. Permitir cualquier acceso desde localhost (lo)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# 4. Abrir el puerto 22 para permitir acceso por SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# 5. Permitir el tráfico por el puerto 80 (HTTP)
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

# 6. Permitir el tráfico por el puerto 443 (HTTPS)
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT

iptables -L -n -v
jumacasni@m1:~$ sudo iptables-save > /etc/iptables/rules.v4
-bash: /etc/iptables/rules.v4: Permission denied
jumacasni@m1:~$ sudo bash -c "iptables-save > /etc/iptables/rules.v4"
jumacasni@m1:~$
```

Figura 14: Aplicar cortafuegos en cada arranque del sistema

```
jumacasni@m4:~$ curl -k http://192.168.56.105
<HTML>
<BODY>
Estás viendo M1!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M2!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k http://192.168.56.105
<HTML>
<BODY>
Estás viendo M1!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M2!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$
```

Figura 15: Peticiones HTTP y HTTPS desde M4

Si, por ejemplo, ahora en M1 configuro el cortafuegos para que no acepte peticiones a través del puerto 80 (Figura 16), podremos acceder a su IP desde **HTTPS** pero no desde **HTTP**, como muestro en la Figura 17.

```
jumacasni@m1:~$ cat cortafuegos.sh
#!/bin/bash

# 1. Eliminar todas las reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# 2. Denegar todo el tráfico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# 3. Permitir cualquier acceso desde localhost (lo)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# 4. Abrir el puerto 22 para permitir acceso por SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# 5. Permitir el tráfico por el puerto 80 (HTTP)
iptables -A INPUT -p tcp --dport 80 -j DROP
iptables -A OUTPUT -p tcp --sport 80 -j DROP

# 6. Permitir el tráfico por el puerto 443 (HTTPS)
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT


iptables -L -n -v
jumacasni@m1:~$ _
```

Figura 16: Configuración de cortafuegos para **NO** aceptar conexiones del puerto 80

```
jumacasni@m2:~$ curl -k https://192.168.56.102
<HTML>
<BODY>
Estás viendo M1!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m2:~$ curl -k http://192.168.56.102
curl: (7) Failed to connect to 192.168.56.102 port 80: Connection timed out
jumacasni@m2:~$
```

Figura 17: No se pueden realizar peticiones a M1 a través de HTTP

4. Segunda tarea opcional: desde M4 sólo M3 servirá las páginas web de M1 y M2 y lo hará de forma balanceada con nginx

Para hacer esto, primero he creado el mismo cortafuegos para M1 y para M2 (Figura 18), denegando todo el tráfico excepto el tráfico de M3, cuya IP es 192.168.56.105.

```
jumacasni@m1:~$ cat cortafuegos_opcional.sh
#!/bin/bash

# 1. Eliminar todas las reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# 2. Denegar todo el tráfico
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP

# 3. Permitir cualquier acceso desde localhost (lo)
iptables -I INPUT -s 192.168.56.105 -j ACCEPT
iptables -I OUTPUT -s 192.168.56.105 -j ACCEPT

iptables -L -n -v
jumacasni@m1:~$ _
```

Figura 18: Cortafuegos de M1 y M2

En M3 he creado un cortafuegos muy similar al que hice en el punto 3 para M1. Este cortafuegos está en la Figura 19.

Para comprobar que funciona correctamente, en la Figura 20 se puede ver como se pueden hacer peticiones **HTTP** y **HTTPS** a M3 (192.168.56.105) pero al intentar conectar con M1 (192.168.56.102) o M2 (192.168.56.103) da error de conexión (ya sea por HTTP o HTTPS).

```
jumacasni@m3:~$ cat cortafuegos.sh
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP

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

iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT

iptables -L -n -v
jumacasni@m3:~$
```

Figura 19: Cortafuegos de M3

```
jumacasni@m4:~$ curl -k http://192.168.56.105
<HTML>
<BODY>
Estás viendo M1!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k http://192.168.56.105
<HTML>
<BODY>
Estás viendo M2!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M1!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.105
<HTML>
<BODY>
Estás viendo M2!
Práctica 4 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl -k https://192.168.56.102
curl: (7) Failed to connect to 192.168.56.102 port 443: Connection timed out
jumacasni@m4:~$ curl -k https://192.168.56.103
curl: (7) Failed to connect to 192.168.56.103 port 443: Connection timed out
jumacasni@m4:~$ curl -k http://192.168.56.102
curl: (7) Failed to connect to 192.168.56.102 port 80: Connection timed out
jumacasni@m4:~$ curl -k http://192.168.56.103
curl: (7) Failed to connect to 192.168.56.103 port 80: Connection timed out
jumacasni@m4:~$
```

Figura 20: Comprobación de que el cortafuegos de la tarea opcional funciona