Servidores Web de Altas Prestaciones (2021-2022)
Grado en Ingeniería Informática
Universidad de Granada

Memoria Práctica 4

Pedro Antonio Mayorgas Parejo

14 de mayo de 2022

Índice

1. Creación de certificados autofirmados de TLS/SSL y firmados por la CA

1.1. Composición de los certificados de TLS/SSL y funcionamiento del protocolo TLS

Los certificados de TLS/SSL son establecidos bajo el **estándar X.509.**
Dicho estándar establece el formato que deben tener los certificados de claves públicas y el algoritmo de validación de dichos certificados consistente en un algoritmo de clave pública.
En el formato se tiene que definir una identificación Distinguished Name DN del proveedor del servicio en Internet:

- Hostname - Domain Name.
- Organización.
- Persona Individual responsable.
- Nombre de la CA que emite el certificado.
- Fecha de emisión y de expiración del certificado.
- Clave pública del Host o Domain Name.
- Firma digital o clave pública de la CA.

NOTA: Los nombres de la CA son en forma de árbol, es decir, a un CA lo puede autentificar otro CA conocido como Root CA. En cuyo caso el cliente al obtener un certificado autentificado por una CA intermedia, suele seguir autenticando la CA(s) intermedia(s) hasta llegar a una CA root de confianza como Digi Cert u otros.

Ejemplo: Google tiene su certificado CA propio conocido como: *Google Trust Services LLC - GTS CA 1C3* utilizado para validar sus propios certificados y servidores. Luego ese certificado de CA para google.com se valida con otro *Google Trust Services - GTS Root R1*, y este a su vez se valida con el CA global de *GlobalSign Root CA*.
Dichos certificados CA están establecidos de manera predeterminada por Firefox en su gestor de certificados, probando su autentificación.
Los algoritmos de par de claves admitidos por TLS/SSL son:

- **RSA - Rivest-Shamir-Adleman.**
- **DSA - Digital Signature Algorithm.**
- **ECDSA - Elliptic Curve Digital Signature Algorithm variante de DSA.**

En dicho estándar se define la figura de **CA Certificate Authority**. Ofrece un certificado raíz confianza en público disponible en programas como los Navegadores Web, o cualquier programa conectado a la red.
Su cometido principal es para lo siguiente: Una compañía o empresa particular quiere ofrecer un servicio en Internet en un servidor público de ellos. Para asegurar la conexión con sus clientes es decir indicar que la conexión es segura, confiable y totalmente privada necesita un certificado de TLS/SSL.
Pasos que debe seguir el protocolo TLS/SSL, conocido como *TLS HANDSHAKE* para que el cliente establezca una conexión con el servidor segura:

1. **CLIENT HELLO:** el cliente indica al servidor, su versión de TLS/SSL, el algoritmo de cifrado simétrico (mejor conocido como Suite de Cifrado) soportados para que se configure una clave de sesión y el algoritmo hash de suma de comprobación soportado para la integridad de los datos.

2. **SERVER HELLO:** el servidor indica la versión TLS/SSL que mantienen en toda la sesión, la configuración del cifrado asimétrico, datos específicos de la sesión (cifrados por su clave privada) y el certificado TLS/SSL. Envía la suma de comprobación HASH DIGEST, cifrada con la clave privada a modo de autenticación como firma y de integridad con la suma HASH.

3. **AUTHENTICATION:** El cliente verifica la autenticidad y integridad del certificado TLS/SSL de la siguiente manera:
   a) **DESCIFRADO y comprobación de la autenticidad del servidor con su firma:** El cliente usa la clave pública del servidor obtenida a través del certificado TLS/SSL (que solo puede cifrar/descifrar en un sentido) la firma del Servidor. Obtendrá descifrado un HASH conocido como Message DIGEST. **Con esto consigue la autenticación de la clave del Servidor y que es de él.** En caso contrario, al no poder descifrar con la clave pública o obtiene un resultado erróneo con el siguiente paso, se sabe que ha ocurrido un problema con el envío de esta información.
   b) **DIGEST:** Luego con el DIGEST en mano, el cliente hace la suma de comprobación del certificado con su DN en texto plano, bajo el mismo algoritmo HASH. Para a continuación compararlo con el DIGEST del servidor, si ve alguna diferencia por pequeña que sea. Lo descarta automáticamente ya que ha sido manipulado y según que programas. Le indican al cliente que ha sido manipulado y es de riesgo.

4. **Creación de la clave de sesión:** Finalmente el cliente confirma que ha llegado el Certificado y ha sido comprobado que es correcto al servidor. Para que luego el cliente mediante los datos de la sesión como semilla (SEED), genera una *clave de la sesión*, que es simétrica. Para su envío la cifra con la clave pública del servidor y se la envía.

5. **Rececpción de la clave de sesión en el servidor y ACK finalizador del TLS HANDSHAKE:** El servidor recibe la clave simétrica de sesión. La descifra con su clave privada y le indica al cliente que está listo para mantener una sesión de TLS/SSL de todos los protocolos superiores de manera segura.

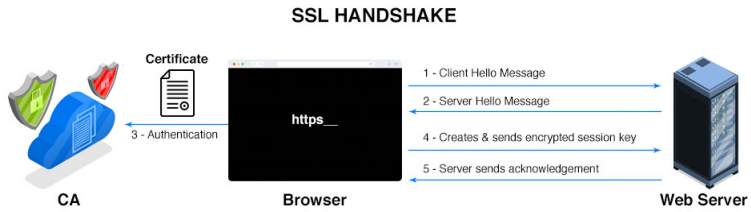

Figura 1: Ilustración de todo el proceso de TLS/SSL obviando TCP para poder establecer una conexión HTTPs.

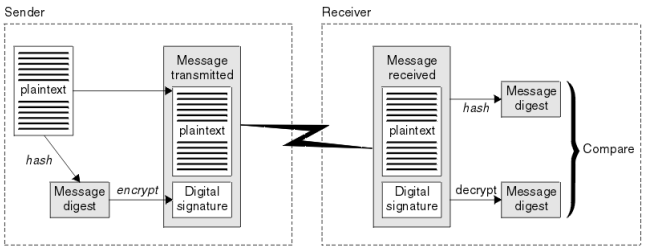

Figura 2: Ilustración del proceso de autenticación y integridad de un certificado con la firma digital.

1.2. Proceso de creado del par de claves público/privadas de la CA.

En mi caso las claves las genero desde ***m2-pedroamp***.
En el proceso de creación primero tenemos que crear las siguientes rutas de directorios:

1. */etc/ssl*: Directorio SSL raíz, utilizado para almacenar todas las claves y certificados.

2. */etc/ssl/private/CApedroamp.key*: Subdirectorio que contendrá la clave privada de la CA.

3. */etc/ssl/private/mswap-pedroamp.key*: Clave privada del dominio de mswap-pedroamp bajo el mismo private.

4. */etc/ssl/certs*: Subdirectorio que contiene los certificados públicos de la CA y de mswap-pedroamp.

Primero nos metemos en el directorio de CAPpedroamp para generar la clave privada de la CA:

```
1    sudo openssl genrsa -aes256 -out /etc/ssl/private/CApedroamp.key 4096
```

- La primera parte del comando es genrsa que consiste en el algoritmo de clave privada que SSL debe usar. Pero puede ser sustituido por los siguientes parámetros:
  - **Generar una clave RSA - Rivest-Shamir-Adleman:** *genrsa*
  - **Generar una clave DSA - Digital Signature Algorithm:** *gendsa*
  - **ECDSA - Elliptic Curve Digital Signature Algorithm variante de DSA requiere de más ajustes:**
    - Para crear una clave ECDSA debemos poner: *ecparam -name ECDSA_ Algorithm*
    - Para saber que algoritmo de curva elíptica escoger se debe consultar: *openssl ecparam -list_ curves*
- La segunda parte nos permite cifrar la clave privada en un algoritmo simétrico para disponer de ella cuando se necesite de manera segura.
- La tercera parte es la ruta de salida de la clave privada de la CA.
- La cuarta y última parte es el tamaño de la clave simétrica en bits. En este caso es 4096.

Figura 3: Generación de la clave privada de la CA cifrada.

Una vez obtenida la clave privada. Tenemos que obtener el certificado y la clave pública:

```
1    sudo openssl req -x509 -sha512 -new -key /etc/ssl/private/CApedroamp.key
       -days 365 -out /etc/ssl/certs/CertCApedroamp.pem
```

- *req:* es el parámetro que indica X.509 Certificate Signing Request. Implica:
  - La generación del DN Distinguished Names del certificado.
  - La generación y inclusión de la clave pública en el certificado.
  - Información sobre la clave y la longitud (en bits).
  - Los argumentos siguientes son directamente dependientes de req:

- *-new* Genera una nueva solicitud de certificado. Generando una salida preguntando por los valores del DN.
- *-key* Entrada de la clave privada de este certificado público X.509. Si es omitido OpenSSL generará una clave privada por defecto, con la configuración de clave privada por defecto si no se indica cual es el algoritmo de clave asimétrica. **Para evitarlo se utiliza -newkey rsa:4096, por utilizar el mismo caso concreto que el mío.**

- *-x509* Indica la generación de un certificado autofirmado en el formato de X.509.
- *-[digest] -sha512* Es el algoritmo Hash que genera el Digest del certificado. Hay muchos otros que son importantes:
  - -md5: Message-Digest 5,
  - -sha256: Mismo algoritmo Secure Hash Algorithms pero con peor precisión.
- *-days [ndays]* Indicamos los días de validez del certificado y la clave pública.
- *-out [path]* Indicamos la ruta de salida del certificado.

```
debian@m2-pedroamp:~$ sudo openssl req -x509 -sha512 -new -key /etc/ssl/private/CApedroamp.key -days 365 -out /etc/ssl/certs/CertCApedroamp.pem
Enter pass phrase for /etc/ssl/private/CApedroamp.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SWAP
Organizational Unit Name (eg, section) []:P4
Common Name (e.g. server FQDN or YOUR name) []:pedroamp
Email Address []:pedroamp@correo.ugr.es
```

Figura 4: Generación del certificado de la CA.

```
debian@m2-pedroamp:~$ sudo cat /etc/ssl/certs/CertCApedroamp.pem
-----BEGIN CERTIFICATE-----
MIIF8TCCA9mgAwIBAgIUWvtU/ZYE57/Nk+6tgeaPvkadVC0wDQYJKoZIhvcNAQEN
BQAwgYcxCzAJBgNVBAYTAkVTMRAwDgYDVQQIDAdHcmFuYWRhMRAwDgYDVQQHDAdH
```

Figura 5: Fichero del certificado de la CA.

1.3. Creación del certificado de mswap-pedroamp y preparación para la firma por la CA

Ahora debemos crear el certificado del Dominio mswap-pedroamp. Hay una variación de pasos que implica el uso de la clave privada de la CA para firmar el certificado de mswap-pedroamp.
Primero creamos la clave privada del dominio mswap-pedroamp:

```
1    sudo openssl genrsa -out /etc/ssl/private/mswap-pedroamp.key 4096
```

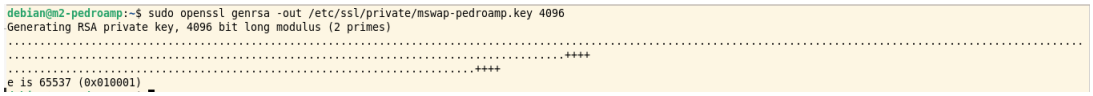

Figura 6: Creando la clave de mswap-pedroamp.

Luego creamos su propio CSR para ser firmado por la CA. *Note que no se incluye el -x509 que indica que es un certificado final autofirmado con la clave privada esto hace que la CA no pueda firmarlo correctamente. Si que genera un Certificate Signed Request, que es un certificado intermedio que es una petición de firma para la CA.*

```
1    sudo openssl req -sha512 -new -key /etc/ssl/private/mswap-pedroamp.key -
         days 365 -out /etc/ssl/certs/Certmswap-pedroamp.csr
```

Figura 7: Creando el CSR de mswap-pedroamp.

1.4. Usando la CA como PKI y firma del certificado de mswap-pedroamp

La CA que tenemos creada la vamos a utilizar como **Public Key Infraestructure**. Es decir, el ordenador m2-pedroamp será el centro emisor de todos los certificados de Clave Pública que verificará todos los dominios privados. Permitiendo la confidencialidad, integridad y autenticidad doble de los certificados autofirmados de mswap-pedroamp y de la CApedroamp.
Para ello utilizo la utilidad de openssl ca que permite de manera portátil generar una PKI a través de comandos. Primero debemos crear un fichero en */etc/ssl/ca.conf*. Que aloja la siguiente configuración:

```
  GNU nano 5.4                                                    /etc/ssl/ca.conf
# we use 'ca' as the default section because we're usign the ca command
[ ca ]
default_ca = CApedroamp

[ CApedroamp ]
DIR = /etc/ssl

#  a text file containing the next serial number to use in hex. Mandatory.
#  This file must be present and contain a valid serial number.
serial = $DIR/serial

# the text database file to use. Mandatory. This file must be present though
# initially it will be empty.
database = $DIR/index.txt

# specifies the directory where new certificates will be placed. Mandatory.
new_certs_dir = $DIR/certs

# the file containing the CA certificate. Mandatory
certificate = $DIR/certs/CertCApedroamp.pem

# the file contaning the CA private key. Mandatory
private_key = $DIR/private/CApedroamp.key

# the message digest algorithm. Remember to not use MD5
default_md = sha512

# for how many days will the signed certificate be valid
default_days = 365

# a section with a set of variables corresponding to DN fields
policy = my_policy

[ my_policy ]
# if the value is "match" then the field value must match the same field in the
# CA certificate. If the value is "supplied" then it must be present.
# Optional means it may be present. Any fields not mentioned are silently
# deleted.
countryName = match
stateOrProvinceName = supplied
organizationName = supplied
commonName = supplied
organizationalUnitName = optional

[ req ]
default_bits = 4096
```

Figura 8: Fichero de ca.conf.

En dicho fichero defino como la ca por defecto a CApedroamp. Que es una directiva que recoge lo siguiente:

- **serial:** Es un fichero de texto que contiene los números de serie actuales que debe utilizar la CA para enumerar los certificados firmados y verificados.

- **database:** Es un fichero de texto que contiene los certificados ya procesados por la CA.
- **new_cer_dir:** Directorio de salida de los certificados ya firmados por la CA. Suele ser distinto a donde se obtienen los certificados sin firmar. Pero por motivos prácticos va a ser el mismo donde se original el certificado que debemos firmar.
- **certificate y private_key:** Ficheros autofirmados de la CA que son utilizados para firmar los otros certificados que quieran usarse en la PKI.
- **defaul_md:** Message-Digest por defecto que debe usarse en la PKI de la CA. En mi caso pongo SHA512
- **default_days = 365:** Indicador de los días por defecto que tienen de validez el certificado firmado.
- **policy:** Política que indica y exige los Distinguished Name de los certificados. En mi caso deben coincidir con los nombres de País y se requiere el resto de información como Provincias, organización, nombre común (Dominio), unidad organizaciónal.
- **req:** Es lo mismo que cuando hacemos el req manual que indica la petición de crear el certificado. En este caso concreto solo especificamos la longitud de bits del certificado.

Los ficheros de serial y el index de la base de datos deben ser creados con touch, de lo contrario openssl ca no fucionará:

```
1 sudo touch /etc/ssl/serial
2 sudo touch /etc/ssl/index.txt
```

Luego el comando necesario para firmar con la CA es como el siguiente:

```
1 sudo openssl ca -create_serial -config /etc/ssl/ca.conf -out
    /etc/ssl/certs/Certmswap-pedroamp_signed.pem -infiles
    /etc/ssl/certs/Certmswap-pedroamp.csr
```

```
debian@m2-pedroamp:~$ sudo openssl ca -rand_serial -config /etc/ssl/ca.conf -out /etc/ssl/certs/Certmswap-pedroamp_signed.pem -infiles /etc/ssl/certs/Certmswap-pedroamp.csr
Using configuration from /etc/ssl/ca.conf
Enter pass phrase for /etc/ssl/private/CApedroamp.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'ES'
stateOrProvinceName   :ASN.1 12:'Granada'
localityName          :ASN.1 12:'Granada'
organizationName      :ASN.1 12:'SWAP'
organizationalUnitName:ASN.1 12:'P4'
commonName            :ASN.1 12:'mswap-pedroamp'
emailAddress          :IA5STRING:'pedroamp@correo.ugr.es'
Certificate is to be certified until May  8 12:17:07 2023 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Figura 9: Firma del Certificado mswap-pedroamp por la CA.

```
debian@m2-pedroamp:~$ cat /etc/ssl/index.txt
V       230508121707Z           46DA59B83A4DF7ABF529B9338B1D7C3257F89CDF        unknown /C=ES/ST=Granada/O=SWAP/CN=mswap-pedroamp/OU=P4
debian@m2-pedroamp:~$ cat /etc/ssl/serial
46DA59B83A4DF7ABF529B9338B1D7C3257F89CE0
debian@m2-pedroamp:~$
```

Figura 10: Registros de la CA

2. Configuración de Nginx con el certificado y clave SSL de mswap-pedroamp

2.1. Servidores HTTPs de Nginx

Ya tenemos el certificado preparado y firmado por la CA. Ahora debemos configurar Nginx para que admita dicho certificado, en /etc/nginx/nginx.conf.

```
1   sudo nano /etc/nginx/nginx.conf
2   sudo systemctl restart nginx.service
3   sudo systemctl status nginx.service
```

En el fichero de configuración de abajo, podemos ver lo siguiente:

- **listen 443 ssl** Indica a Nginx que se configure para usar el HTTP con SSL o HTTPs.
- **server_name** Importante ya que con la inclusión del certificado SSL, su FDQN incluido en el DN debería coincidir con el servidor que está solicitando.
- **ssl_protocols** Aquí por defecto Nginx en su documentación ofrece la posibilidad de usar los protocolos desde el TLSv1 hasta el TLSv1.3. Los TLSv1 y TLSv1.1 no son nada recomendables usar en la actualidad.
- **ssl_certificate y ssl_key** Indicamos las rutas que contienen los ficheros de Certificado y Clave privada para la conexión TLS.

```
##
# SSL Settings
##
server {
        # Puerto de escucha SSL
        listen 443 ssl;
        server_name m2-pedroamp;

        # Protocolos se ha desactivado los TLSv1 y TLSv1.1 por ser arriesgados y deprecated.
        ssl_protocols TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        # Rutas de clave privada y el certificado de SSL de mswap-pedroamp
        ssl_certificate /etc/ssl/certs/Certmswap-pedroamp_signed.pem;
        ssl_certificate_key /etc/ssl/private/mswap-pedroamp.key;
}
```

Figura 11: Fichero de configuración listo para m2-pedroamp

En el servidor m2-pedroamp he habilitado temporalmente el uso del root para poder copiar los ficheros restringidos por SCP. Los cuales son los del certificado, su clave y la configuración para Nginx. Que son la misma.

```
debian@m1-pedroamp:~$ sudo scp root@192.168.122.181:/etc/nginx/nginx.conf /etc/nginx/nginx.conf
root@192.168.122.181's password:
nginx.conf                                                                                                    100% 2268     2.7MB/s   0
debian@m1-pedroamp:~$ sudo scp root@192.168.122.181:/etc/ssl/certs/Certmswap-pedroamp_signed.pem /etc/ssl/certs/Certmswap-pedroamp_signed.pem
root@192.168.122.181's password:
Certmswap-pedroamp_signed.pem                                                                                 100% 6752     6.7MB/s   0
debian@m1-pedroamp:~$ sudo scp root@192.168.122.181:/etc/ssl/private/mswap-pedroamp.key /etc/ssl/private/mswap-pedroamp.key
root@192.168.122.181's password:
mswap-pedroamp.key                                                                                            100% 3243     4.4MB/s   0
debian@m1-pedroamp:~$ sudo systemctl restart nginx.service
debian@m1-pedroamp:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-05-08 16:53:11 UTC; 7s ago
       Docs: man:nginx(8)
    Process: 851 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 853 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 854 (nginx)
      Tasks: 3 (limit: 4678)
```

Figura 12: Copia a m1-pedroamp de los certificados.

```
##
# SSL Settings
###
server {
        # Puerto de escucha SSL
        listen 443 ssl;
        server_name m1-pedroamp;

        # Protocolos se ha desactivado los TLSv1 y TLSv1.1 por ser arriesgados y deprecated.
        ssl_protocols TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        # Rutas de clave privada y el certificado de SSL de mswap-pedroamp
        ssl_certificate /etc/ssl/certs/Certmswap-pedroamp_signed.pem;
        ssl_certificate_key /etc/ssl/private/mswap-pedroamp.key;
}
```

Figura 13: Fichero de configuración listo para m1-pedroamp

Finalmente para que en HTTPs no nos sirva el sitio por defecto debemos ir a la ruta */etc/nginx/sites-avaiable/default*. Donde debemos poner en el fichero de configuración del VirtualHost la siguiente información para que la página sea servida por HTTPs. Esto debe ser configurado en ambos servidores web **m1-pedroamp m2-pedroamp**.

```
# Default server configuration
#
server {
        #listen 80 default_server;
        #listen [::]:80 default_server;

        # SSL configuration
        #
        # Protocolos se ha desactivado los TLSv1 y TLSv1.1 por ser arriesgados y deprecated.
        ssl_protocols TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        # Rutas de clave privada y el certificado de SSL de mswap-pedroamp
        ssl_certificate /etc/ssl/certs/Certmswap-pedroamp_signed.pem;
        ssl_certificate_key /etc/ssl/private/mswap-pedroamp.key;

        listen 443 ssl default_server;
        listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
```

Figura 14: Fichero de VirtualHosts por defecto.

Podemos ver que se ha configurado correctamente con la siguiente visualización de puertos.

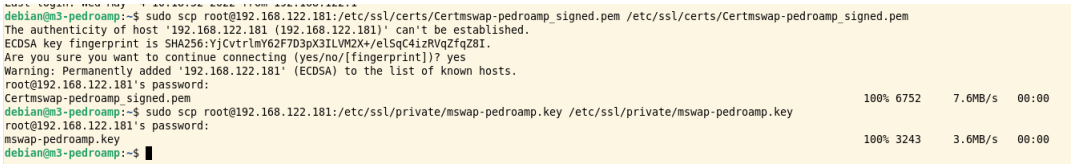

Figura 15: Copia a m1-pedroamp.

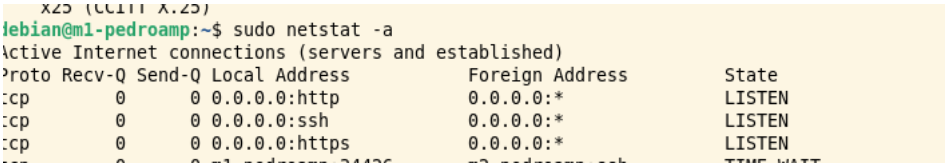

Figura 16: Copia a m1-pedroamp y visualización de puertos.

Finalmente reiniciamos el servicio y ya está en activo.

```
1    sudo systemctl restart nginx.service
2    sudo systemctl status nginx.service
```

2.2. Balanceador HTTPs de Nginx

Finalmente tenemos que activar el balanceador con el certificado SSL, como se ha dicho con el usuario Root copiamos los ficheros de Key y el certificado .pem.

```
debian@m3-pedroamp:~$ sudo scp root@192.168.122.181:/etc/ssl/certs/Certmswap-pedroamp_signed.pem /etc/ssl/certs/Certmswap-pedroamp_signed.pem
The authenticity of host '192.168.122.181 (192.168.122.181)' can't be established.
ECDSA key fingerprint is SHA256:YjCvtrlmY62F7D3pX3ILVM2X+/elSqC4izRVqZfqZ8I.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.122.181' (ECDSA) to the list of known hosts.
root@192.168.122.181's password:
Certmswap-pedroamp_signed.pem                                                                    100% 6752     7.6MB/s   00:00
debian@m3-pedroamp:~$ sudo scp root@192.168.122.181:/etc/ssl/private/mswap-pedroamp.key /etc/ssl/private/mswap-pedroamp.key
root@192.168.122.181's password:
mswap-pedroamp.key                                                                               100% 3243     3.6MB/s   00:00
debian@m3-pedroamp:~$
```

Figura 17: Copia a m3-pedroamp del certificado y la clave privada.

La configuración es como la que se puede visualizar en la siguiente foto:

```
log_format upstreamlog 'Client_addr: $remote_addr -Server_addr: $upstream_a

upstream upstreampedroamp {
        server m1-pedroamp:443;
        server m2-pedroamp:443;
}

server {
        listen 80;
        server_name mswap-pedroamp;

        return 301 https://mswap-pedroamp;
}

##
# SSL Settings
##
server {
        listen 443 ssl;

        # Rutas de clave privada y el certificado de SSL de mswap-pedroamp
        ssl_certificate /etc/ssl/certs/Certmswap-pedroamp_signed.pem;
        ssl_certificate_key /etc/ssl/private/mswap-pedroamp.key;

        ssl_protocols TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        location / {
                proxy_pass https://upstreampedroamp;
        }
}
##
```

Figura 18: Configuración del balanceador de carga Nginx.

- Está la redirección ya explicada en el punto anterior, donde genera un código HTTP 301 para que el navegador (o otro cliente web). Que solicite una petición de HTTP, se redirija dicha petición al HTTPs.
- Luego indicamos como en los servidores ñormales; las rutas de los cerrtificados de SSL. Además de su versión del protocolo TLS utilizada con el certificado.
- Finalmente indicamos que cuando escuche una petición en el puerto 443, se redirija como proxy por el protocolo HTTPs(AKA balanceador de carga). Hacia los servidores upstream de upstreampedroamp.
- En el upstream debemos tener cuidado con no olvidarnos de poner HOSTNAME:443. Ya que en otro caso nos dará un error de 502 Bad Gateway en el cual el proxy no sabe como procesar las peticiones HTTPs con su upstream ya que no se le indica en ningún caso que trabaje con HTTPs por su puerto.

Finalmente reiniciamos el servicio y ya está en activo.

```
1    sudo systemctl restart nginx.service
2    sudo systemctl status nginx.service
```

2.2.1. Configuración de redirección automática de HTTP a HTTPs

En los servidores de Nginx tenemos la opción de que podemos redirigir el tráfico del puerto 80 indicándole al cliente que vuelva a hacer la petición al puerto 443 y que lo haga de manera segura.
Esto se logra mediante la respuesta de un código HTTP 301 que indica movido permanentemente **HTTP 301 Moved Permanently**. Esto le indica al navegador que se ha logrado contactar con un servidor correcto de HTTP, pero que atiende las peticiones en la URL que le devuelve bajo el código de 301. El cliente debe tomar las medidas para redirigir la solicitud al tráfico HTTPs.

```
##
# SSL Settings
##
server {
        listen 80;
        server_name mswap-pedroamp;

        return 301 https://mswap-pedroamp;
}
```

Figura 19: Redirección automática con el código HTTP 301.

3. Configuración de CA y HTTPs en el cliente.

Los certificados autofirmados y con una CA no identificada. Suelen tener el problema de que no son reconocidos automáticamente como *"confiables"* por los navegadores. Saliendo lo siguiente si mandas una petición a **https://mswap-pedroamp**.

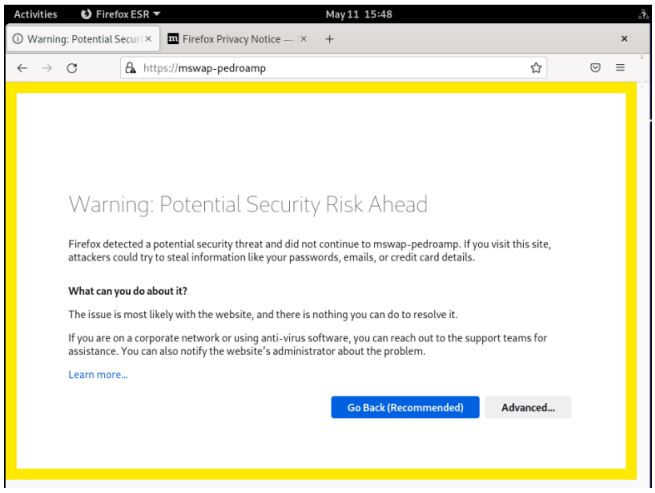

Figura 20: El navegador marca el certificado SSL como potencialmente arriesgado.

Le damos a avanzado y vemos el certificado que nos saldría como sigue:

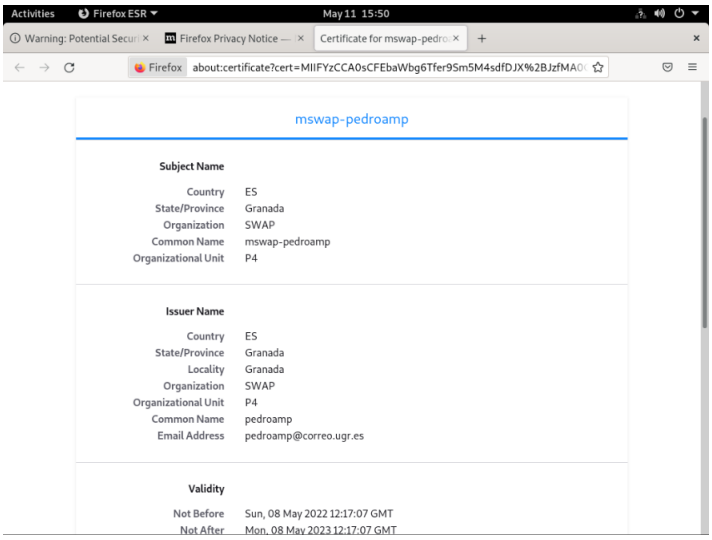

Figura 21: Información de DN de mswap-pedroamp.

La página a pesar de aceptar que es como confiable el navegador sigue marcándola como que **NO** es confiable. Para evitar eso necesitamos adjuntar a la configuración del Navegador el certificado de la CA para poder marcarlo como confiable.

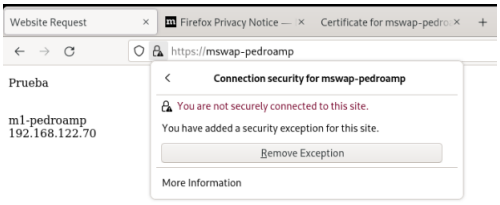

Figura 22: Sigue marcando como inseguro a mswap-pedroamp.

En Firefox nos vamos a *about:preferences* y en Privacidad y Seguridad buscamos el Gestor de Certificados. Previamente tenemos que tener descargado el certificado de la CA.

```
debian@m4-pedroamp:~$ sudo scp root@192.168.122.181:/etc/ssl/certs/CertCApedroamp.pem /etc/ssl/certs
root@192.168.122.181's password:
CertCApedroamp.pem                            100% 2122     1.9MB/s   00:00
debian@m4-pedroamp:~$
```

Figura 23: Descarga del certificado de CA.

Luego en el gestor de certificados de Firefox en la pestaña de Autoridades. Debemos darle a import para buscar en la ruta de */etc/ssl/certs*. El certificado de CA correspondiente.

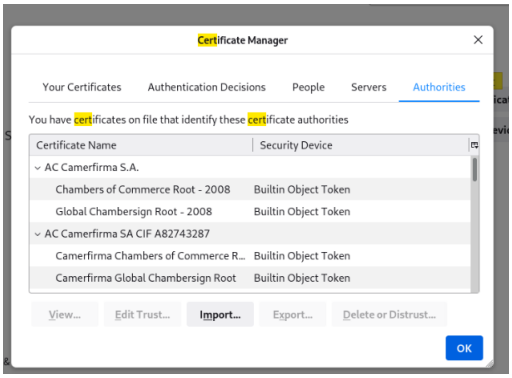

Figura 24: Importando la CA.

Luego de buscar la ruta seleccionamos nuestro CA.pem para importarlo.

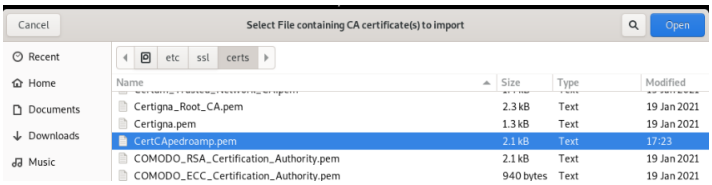

Figura 25: Seleccionando para la importación el certificado de la CA.

Luego Firefox nos hará una pregunta de si queremos confiar en esta CA (trusted). Le damos a que verifique nuestros dominios y mails asociados.

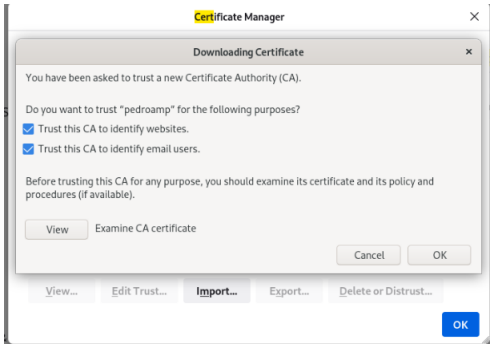

Figura 26: Seleccionando qué queremos verificar con la CA.

Finalmente se incluye en el Gestor de Certificados.

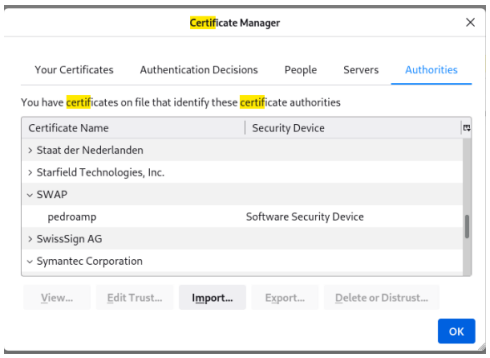

Figura 27: CA incluida en el Gestor de Certificados.

Finalmente el balanceador de carga y sus servidores están verificados por mi CA. Marcados como confiables.

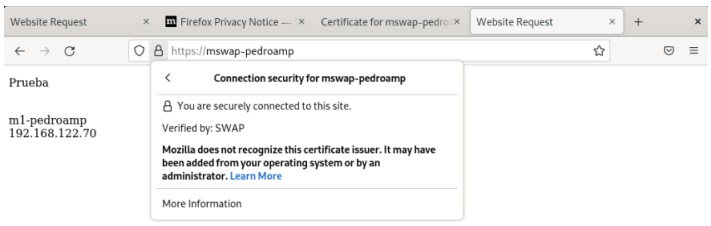

Figura 28: m1-pedroamp bajo dominio verificado de mswap-pedroamp

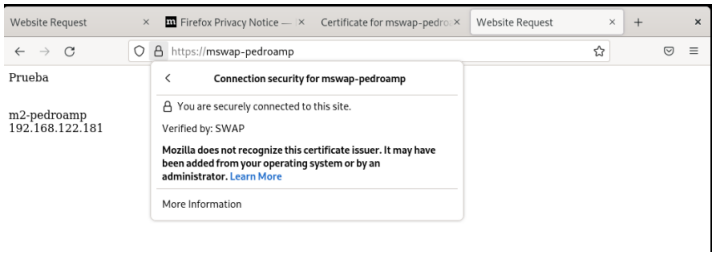

Figura 29: m2-pedroamp bajo dominio verificado de mswap-pedroamp

4. Firewall en GNU/Linux iptables

Es un cortafuegos que utiliza netfilter debajo para poder realizar sus tareas. Las reglas exigidas para esta práctica son:

- Bloqueo de todas los tipos de conexiones entrantes, salientes y que atraviesen el sistema (FORWARD).
- Permitir en las conexiones en localhost todo tipo de conexiones.
- Permitir la conexión SSH externa y trazarla de manera que las conexiones salientes del puerto SSH. Son aquellas que se hayan permitido como nuevas y establecidas de la entrada. En todos los servidores
- Permitir el icmp echo ping en todos los servidores para poder comprobar su estado.
- Conexión HTTPs
  - Las conexiones de cliente-servidor solo deben llegar al m3-pedroamp conocido externamente como mswap-pedroamp.
  - Las conexiones de servidor a servidor, deben ser de m3-pedroamp a sus servidores web backend. Como: m1-pedroamp m2-pedroamp. Iniciandose siempre desde m3-pedroamp y trazando esa conexión (conntrack) en los servidores de m1 y m2.

4.1. Disección de las reglas de m3-pedroamp

4.1.1. Variables de servidores web backend y del cliente. Limpieza de las reglas anteriores

Aquí establezco dos variables que me permiten saber qué direcciones IPv4 son las de los servidores web backend que están relacionadas con el balanceador de carga y la del cliente.
Además en la última línea de esta parte se limpia las reglas anteriores antes de introducir las políticas generales del Firewall.

```
1 # VARIABLES for IP
2 m1="192.168.122.70"
3 m2="192.168.122.181"
4 m4="192.168.122.254"
5
6 # Flushing iptables
7 iptables --flush
```

4.1.2. White-listing Stateful Firewall

Aquí establezco las políticas por defecto de White-listing (con el parámetro -P). Es decir estamos configurando un Firewall, que al bloquear todo tipo de tráfico, luego indicamos las conexiónes con un grano fino. Es decir qué tipos y modos de conexiones se permiten

a continuación de las reglas de bloqueo total (con el argumento -A append rules). Los tipos y modos son: (TCP, UDP, puertos, IPs origen y destino,...)

Es un Firewall de White-listing, **permitimos solo lo que le indicamos. El caso contrario son los Firewall de Black-listing, donde bloqueamos solo lo que le indicamos, todo lo contrario.**

NOTA: La política DROP es similar a REJECT. Solo que REJECT emite una respuesta indicando que ha sido denegada la petición. DROP por el contrario desecha la petición de innmediato y el cliente o el emisor de peticiones que han sido procesadas por el Firewall solo detectan un Destination Hosts Unreachable. Ya que no reciben una respuesta y no identifican qué ocurre.

```
1 # White Listing Firewall policy rules
2 iptables -P INPUT DROP
3 iptables -P OUTPUT DROP
4 iptables -P FORWARD DROP
```

4.1.3. Reglas Loopback

Aquí empiezan las reglas de permitir las conexiones Loopback. Las indico por la interfaz directamente por estar asociado a dicha dirección IP Loopback.

```
1 # Permitir conexiones en localhost
2 iptables -A INPUT -i lo -j ACCEPT
3 iptables -A OUTPUT -o lo -j ACCEPT
```

4.1.4. Reglas SSH. Explicación del Stateful Firewall y su módulo connection tracking conntrack

Aquí empiezan las reglas de SSH, tal y como explico en los comentarios. Iptables, es un *Stateful Firewall*, es decir que tiene un registro del estado de las conexiones. Para ello se usa TCP, ya que se puede trazar cuáles son conexiones nuevas (NEW), conexiones establecidas (ESTABLISHED) y conexiones relacioneadas (RELATED). Según los flags de TCP, que son la relación siguiente:

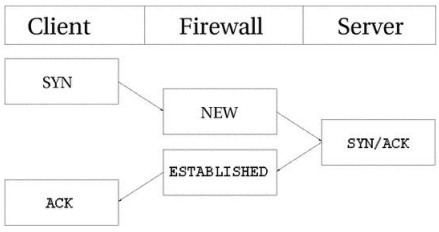

Figura 30: Relación de conntrack de iptables y flags TCP.

Cuando llega un paquete TCP iniciando el *Three-Way Handshake* con el flag en **SYN** donde negocia un establecimiento de una conexión nueva.
Iptables establece un registro en su tabla de connection tracking */proc/net/ip_ conntrack*. Con la traza de SYN_SENT, SYN enviado. Esto es el HOOK de estado NEW, donde marca que se permite una conexión nueva, registrando la ip de origen, puerto efímero de origen, dirección IP de destino y puerto de destino del servicio.

Luego cuando se envía a través de la pila (stack) TCP/IP a la capa de aplicación al socket del puerto en el que se solicita el servicio. Este si detecta que, la petición es correcta, antes de responder a dicha primero manda un SYN/ACK de respuesta como parte del TCP Three-Way Handshake al cliente de la petición. Al bajar por la stack TCP/IP, iptables atrapa la respuesta con su HOOK y detecta que es una respuesta de un SYN registrado, va a la tabla y cambia el estado a ESTABLISHED y permite bajar a las capas inferiores para que se transforme en trama (Capa 2).

Ahora con el estado de esa conexión en ESTABLISHED. El servicio ha terminado de preparar la respuesta a esa petición, la envía y baja por la pila TCP/IP. Entonces esa petición saliente es un OUTPUT, que contiene como dport el puerto efímero del cliente, iptables atrapa esa respuesta y comprueba si está permitida la salida de esa respuesta. Va a detectar que esa salida existe (tiene un HOOK). Con la adición de una conexión ESTABLISHED ya que la regla implica el uso de conntrack ESTABLISHED. Finalmente iptables busca en su tabla de conexiones, donde debe existir una conexión ESTABLISHED con el sport efímero original == dport efímero saliente y permite que el paquete TCP se envíe a las capas inferiores y que el cliente reciba su respuesta.

Esto permite un filtrado que en un **Firewall sin estado ocurriría el siguiente caso:**

Tenemos una conexión entrante en SSH por su puerto 22. El Firewall busca en su tabla de filtros un hook de si esa conexión con el puerto TCP 22 está permitida. Cuando coincide con un HOOK que haga hit con ese puerto y el valor es ACCEPT. La conexión se permitirá y subirá a la capa de aplicación a buscar el socket de escucha de SSH.

Pero, aquí es donde surge el problema, cuando queramos sacar una conexión SSH legítima desde el servidor, en el firewall sin estado, su regla de salida ÖUTPUTïndicaría que está permitido tanto las salidas de conexiones establecidas como las salidas nuevas con origen en el servidor.

Esto plantea un problema de seguridad, ya que nadie que trabaje con el servidor usará una terminal física que se conecte con SSH hacia fuera del servidor. Entonces es un atacante con un software malicioso que aprovecha este deficit de seguridad, hace pasar la conexión nueva de SSH como válida con una conexión inversa permitida desde dicho software malicioso a la terminal que controle el atacante, permitiendo ejecutar comandos y ejecutar un ataque en toda regla, sin tener que autenticarse con SSH.

Es mucho más complejo el ataque de lo que explico, es decir para empezar que hace un proceso malicioso ejecutado en el sistema, cómo se ha instalado... Pero ilustro la manera en la que se pueden aprovechar esos ẗuecosën un firewall sin estado para que el atacante pueda obtener información confidencial.

En el firewall con estado le indicamos con aún mas especifidad **que las conexiones salientes permitidas son aquellas que han sido entrantes, se hayan aprobado y estén trazadas (track) por el firewall.** Aquí es donde entra el módulo de connection tracking (conntrack) de Iptables que tiene varios modos de funcionamiento

```
1 # SSH rules
2 # La cadena de INPUT siguiendo el TCP Handshake se establece un registro en
      la tabla de estados de IPTABLES (cortafuegos con estado STATEFUL)
3 # en la que toda la conexion nueva *NEW debe ser registrada en dicha tabla
    y pasa a ser como conexion
4 # establecida. Dicho registro en el estado ESTABLISHED permite la siguiente
      cadena en la cual sale una conexion de
5 # SSH permitida a traves del puerto 22.
6
7 iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j
    ACCEPT
8 iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

4.1.5. Reglas ICMP ping ECHO

Aquí permito el ICMP echo PING donde indico que se permite las echo-request (PING) y las respuestas echo-reply. Sin otros campos de ICMP.

```
1 # ICMP echo PING RULES ALLOW
2 iptables -A INPUT -p icmp --icmp-type echo-request -m state --state NEW,
    ESTABLISHED,RELATED -j ACCEPT
3 iptables -A OUTPUT -p icmp --icmp-type echo-reply -m state --state
    ESTABLISHED,RELATED -j ACCEPT
```

4.1.6. Reglas HTTPs

Aquí nos enfrentamos a que primero debemos permitir el tráfico de los clientes. En nuestro caso concreto por limitación de la práctica, voy a poner la dirección IPv4 de m4-pedroamp como si fueran todos los posibles clientes. Pero se pueden indicar direcciones de subred enteras, direcciones de IP pública, si estamos en una NAT, indicamos que solo permitimos el tráfico HTTPs entrante desde el router.

Aparte, debemos considerar que el cliente tenga algún cliente web que por defecto mande peticiones al puerto 80 HTTP, donde nuestro balanceador está capacitado para responder con una respuesta HTTP 301 que indica al cliente web que vuelva a realizar la petición al puerto 443 y se configure para SSL.

```
# Conexiones HTTPs permitidas
# Mensaje HTTP 301 de reconfiguracion del navegador o cliente
iptables -A INPUT -p tcp -s ${m4} --dport 80 -m state --state NEW,
    ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -d ${m4} --sport 80 -m state --state ESTABLISHED
    -j ACCEPT

# Regla que permite la entrada del cliente al balanceador
iptables -A INPUT -p tcp -s ${m4} --dport 443 -m state --state NEW,
    ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -d ${m4} --sport 443 -m state --state ESTABLISHED
     -j ACCEPT
```

Luego las reglas de conexión con los servidores son muy sencillas, deben establecerse que la salida desde el balanceador de carga se registre y permita la entrada de la respuesta de los servidores.

```
# Regla de contacto con el balanceador a sus servidores
iptables -A OUTPUT -p tcp -d ${m1},${m2} --dport 443 -m state --state NEW,
    ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -s ${m1},${m2} --sport 443 -m state --state
    ESTABLISHED -j ACCEPT
```

4.1.7. DNS y APT

Tenemos además que añadir una regla que permita el uso del DNS y de APT para motivos de mantenimiento. De lo contrario no podremos ni actualizar o instalar paquetería necesaria para nuestro sistema. Ejemplo: iptables-persistent.

```
# Regla para permitir el uso de un repositorio de APT
iptables -A OUTPUT -p tcp -d  199.232.182.132 --dport 80 -m state --state
    NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -s 199.232.182.132 --sport 80 -m state --state
    ESTABLISHED -j ACCEPT

# Regla de DNS
iptables -A OUTPUT -p udp --dport 53 -m state --state NEW,ESTABLISHED -j
    ACCEPT
iptables -A INPUT -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
```

4.2. Disección de las reglas de m1-pedroamp y m2-pedroamp

El contenido principal es similar, es decir las políticas por defecto son White-listing, el ping permitido y el SSH también permitido. Además las reglas de APT y DNS son las mismas.

4.2.1. Variables

Solo tiene una única variable que es para identificar al balanceador de carga m3-pedroamp.

```
1 # VARIABLES for IP
2 m3="192.168.122.57"
```

4.2.2. Regla de contacto de HTTPs solo con el balanceador

Esta regla indica que se traza y registra una conexión entrante solo desde el m3-pedroamp con los estados NEW y ESTABLISHED. ESTABLISHED, es necesario por si hay persistencia de sesión HTTPs.

```
1 # Regla de contacto con desde el balanceador a sus servidores
2 # Solo se permite el contacto del balanceador para el puerto HTTPs 443
3 iptables -A INPUT -p tcp -s ${m3} --dport 443 -m state --state NEW,
      ESTABLISHED -j ACCEPT
4 iptables -A OUTPUT -p tcp -d ${m3} --sport 443 -m state --state ESTABLISHED
      -j ACCEPT
```

4.3. Fichero Iptables de m3-pedroamp

```
#!/bin/bash

# VARIABLES for IP
m1="192.168.122.70"
m2="192.168.122.181"
m4="192.168.122.254"

# Flushing iptables
iptables --flush

# White Listing Firewall policy rules
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Permitir conexiones en localhost
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# SSH rules
# La cadena de INPUT siguiendo el TCP Handshake se establece un registro en la tabla de estados de IPTABLES (cortafuegos con estado STATEFUL)
# en la que toda la conexion nueva *NEW debe ser registrada en dicha tabla y pasa a ser como conexion
# establecida. Dicho registro en el estado ESTABLISHED permite la siguiente cadena en la cual sale una conexion de
# SSH permitida a traves del puerto 22.

iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

# ICMP echo PING RULES ALLOW
iptables -A INPUT -p icmp --icmp-type echo-request -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -m state --state ESTABLISHED,RELATED -j ACCEPT

# Conexiones HTTPs permitidas
# Mensaje HTTP 301 de reconfiguracion del navegador o cliente
iptables -A INPUT -p tcp -s ${m4} --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -d ${m4} --sport 80 -m state --state ESTABLISHED -j ACCEPT

# Regla que permite la entrada del cliente al balanceador
iptables -A INPUT -p tcp -s ${m4} --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -d ${m4} --sport 443 -m state --state ESTABLISHED -j ACCEPT

# Regla de contacto con el balanceador a sus servidores
```

```
43 iptables -A OUTPUT -p tcp -d ${m1},${m2} --dport 443 -m state --state NEW,
       ESTABLISHED -j ACCEPT
44 iptables -A INPUT -p tcp -s ${m1},${m2} --sport 443 -m state --state
       ESTABLISHED -j ACCEPT
45
46 # Regla para permitir el uso de un repositorio de APT
47 iptables -A OUTPUT -p tcp -d  199.232.182.132 --dport 80 -m state --state
       NEW,ESTABLISHED -j ACCEPT
48 iptables -A INPUT -p tcp -s 199.232.182.132 --sport 80 -m state --state
       ESTABLISHED -j ACCEPT
49
50 # Regla de DNS
51 iptables -A OUTPUT -p udp --dport 53 -m state --state NEW,ESTABLISHED -j
       ACCEPT
52 iptables -A INPUT -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
```

4.4. Fichero de Iptables de m1-pedroamp y m2-pedroamp

```
#!/bin/bash

# VARIABLES for IP
m3="192.168.122.57"

# Flushing iptables
iptables --flush

# White Listing Firewall
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Permitir conexiones en localhost
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# SSH rules
# La cadena de INPUT siguiendo el TCP Handshake se establece un registro en la tabla de estados de IPTABLES (cortafuegos con estado STATEFUL)
# en la que toda la conexion nueva *NEW debe ser registrada en dicha tabla y pasa a ser como conexion
# establecida. Dicho registro en el estado ESTABLISHED permite la siguiente cadena en la cual sale una conexion de
# SSH permitida a traves del puerto 22.

iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

# ICMP echo PING RULES ALLOW
iptables -A INPUT -p icmp --icmp-type echo-request -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -m state --state ESTABLISHED,RELATED -j ACCEPT

# Regla de contacto con desde el balanceador a sus servidores
# Solo se permite el contacto del balanceador para el puerto HTTPs 443
iptables -A INPUT -p tcp -s ${m3} --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -d ${m3} --sport 443 -m state --state ESTABLISHED -j ACCEPT

# Regla para permitir el uso de un repositorio de APT
iptables -A OUTPUT -p tcp -d deb.debian.org --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -s deb.debian.org --dport 80 -m state --state ESTABLISHED -j ACCEPT

# Regla de DNS
iptables -A OUTPUT -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
```

```
42  iptables -A INPUT -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
```

4.5. Pruebas y instalación persistente

Para las pruebas ejecuto simplemente el script hasta comprobar que todo vaya correctamente.

```
1  sudo nano /etc/iptables.sh # Localizacion del fichero de las reglas
2  sudo chmod +x /etc/iptables.sh # Dando permisos de ejecucion
3  sudo /etc/iptables.sh
```

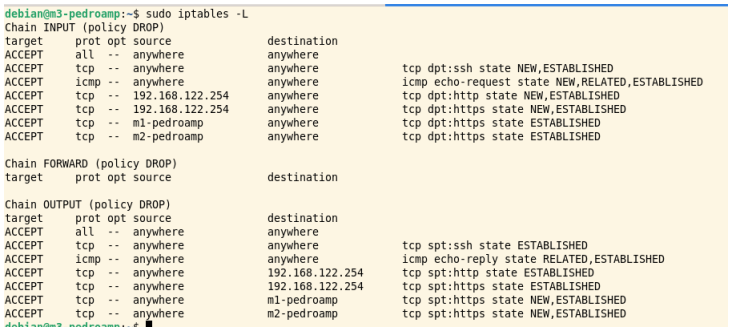

Figura 31: Iptables en m3-pedroamp

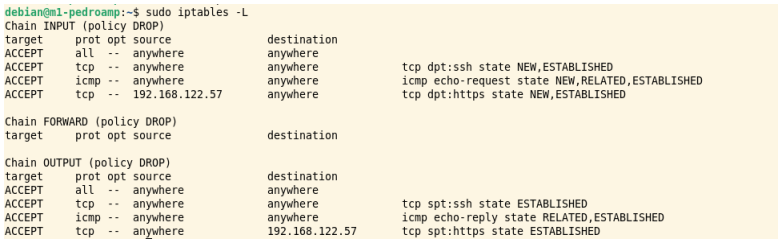

Figura 32: Iptables en m1-pedroamp

Figura 33: Iptables en m2-pedroamp

4.6. Ejecución de pruebas

4.6.1. ICMP echo ping

Probamos desde el cliente si hay respuestas al ping.

```
debian@m4-pedroamp:~$ ping 192.168.122.70
PING 192.168.122.70 (192.168.122.70) 56(84) bytes of data.
64 bytes from 192.168.122.70: icmp_seq=1 ttl=64 time=0.328 ms
64 bytes from 192.168.122.70: icmp_seq=2 ttl=64 time=0.478 ms
^C
--- 192.168.122.70 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.328/0.403/0.478/0.075 ms
debian@m4-pedroamp:~$ ping 192.168.122.181
PING 192.168.122.181 (192.168.122.181) 56(84) bytes of data.
64 bytes from 192.168.122.181: icmp_seq=1 ttl=64 time=0.462 ms
64 bytes from 192.168.122.181: icmp_seq=2 ttl=64 time=0.447 ms
^C
--- 192.168.122.181 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.447/0.454/0.462/0.007 ms
debian@m4-pedroamp:~$ ping mswap-pedroamp
PING mswap-pedroamp (192.168.122.57) 56(84) bytes of data.
64 bytes from mswap-pedroamp (192.168.122.57): icmp_seq=1 ttl=64 time=0.436 ms
64 bytes from mswap-pedroamp (192.168.122.57): icmp_seq=2 ttl=64 time=0.483 ms
64 bytes from mswap-pedroamp (192.168.122.57): icmp_seq=3 ttl=64 time=0.455 ms
^C
--- mswap-pedroamp ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.436/0.458/0.483/0.019 ms
```

Figura 34: Pruebas ICMP en m4-pedroamp de m1-pedroamp a m3-pedroamp

4.6.2. HTTPs

Probamos en cURL si tenemos acceso directo al balanceador y vemos que recibe tanto del puerto 80 con el HTTP 301. Como en el HTTPs.

```
debian@m4-pedroamp:~$ curl -iL  --cacert /etc/ssl/certs/CertCApedroamp.pem http://mswap-pedroamp
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0
Date: Sat, 14 May 2022 16:40:45 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://mswap-pedroamp

HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Sat, 14 May 2022 16:40:46 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

<html>
        <head>
                <title>Website Request</title>
        </head>
        <body>
                <p>Prueba</p>
                <br>m2-pedroamp<br>192.168.122.181<br>  </body>
</html>
debian@m4-pedroamp:~$ curl -iL  --cacert /etc/ssl/certs/CertCApedroamp.pem http://mswap-pedroamp
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0
Date: Sat, 14 May 2022 16:40:51 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://mswap-pedroamp

HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Sat, 14 May 2022 16:40:51 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

<hmtl>
        <head>
                <title>Website Request</title>
        </head>
        <body>
                <p>Prueba</p>
                <br>m1-pedroamp<br>192.168.122.70<br>   </body>
</html>

debian@m4-pedroamp:~$
```

Figura 35: HTTP y HTTPs con cURL al balanceador.

Probamos con los Servidores

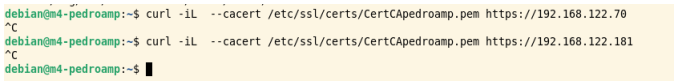

Figura 36: HTTP y HTTPs con cURL al m1-pedroamp y m2-pedroamp.

4.6.3. Daemonize iptables

Tenemos una utilidad llamada iptables-persistent que permite que se almacenen las reglas al reiniciar el sistema o en caso de apagón. Para ello la instalamos y usamos iptables-save en la ruta especificada abajo que es donde lee este demonio las reglas de iptables.

```
1    sudo apt-get install iptables-persistent
2    sudo iptables-save | sudo tee /etc/iptables/rules.v4
```

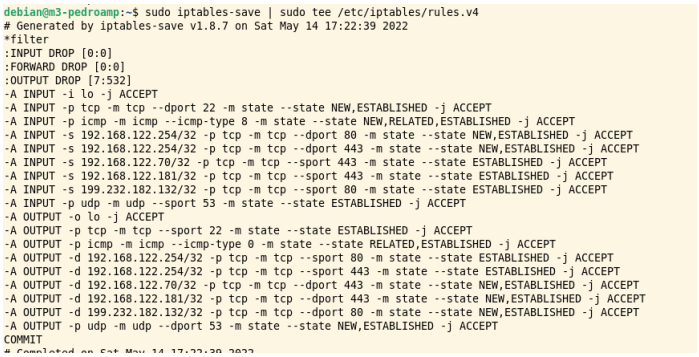

Figura 37: Output generado por iptables-save.

Ahora activamos el demonio y lo cargamos. Y en el estado final podemos visualizar que no hay reglas de IPv6 que no se han guardado.

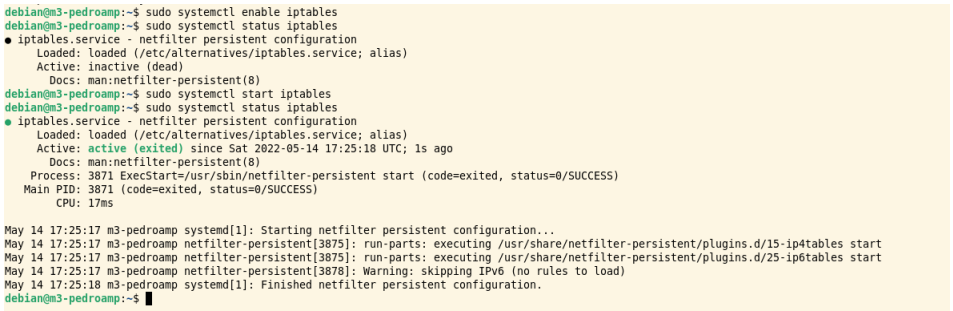

Figura 38: Levantando demonio de iptables

Finalmente reiniciamos el sistema (uso init 6). Y verificamos que las reglas efectivamente se recargan.

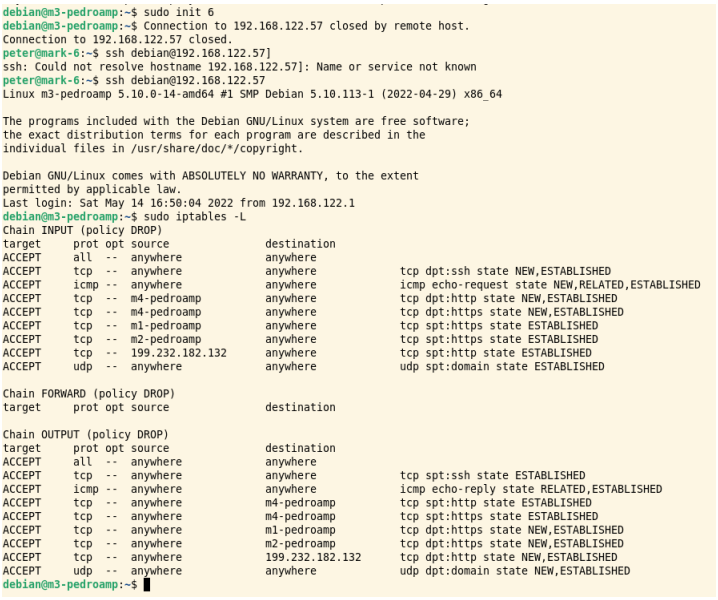

Figura 39: Reglas de iptables cargadas después del reinicio.

5. Bibliografía

5.1. TLS y certificados

[1] https://www.encryptionconsulting.com/education-center/ssl-tls-certificates/
[2] https://en.wikipedia.org/wiki/Transport_Layer_Security
[3] https://en.wikipedia.org/wiki/X.509
[4] https://www.ibm.com/docs/en/ibm-mq/7.5?topic=ssl-digital-signatures
[5] https://www.ibm.com/docs/en/db2/11.1?topic=SSEPGG_11.1.0/com.ibm.db2.luw.admin.sec.doc/doc/c0053515.html
[6] https://www.ibm.com/docs/en/ibm-mq/7.5?topic=certificates-how-certificate-chains-work
[7] https://www.ibm.com/docs/en/i/7.2?topic=concepts-distinguished-names-dns
[8] https://www.ibm.com/docs/en/runbook-automation?topic=certificate-generate-root-ca-key
[9] https://www.ibm.com/docs/en/api-connect/10.0.1.x?topic=overview-generating-self-signed-certificate-using-openssl
[10] https://en.wikipedia.org/wiki/Transport_Layer_Security
[11] https://www.phildev.net/ssl/creating_ca.html
[12] https://www.phildev.net/ssl/opensslconf.html
[13] https://www.phildev.net/ssl/opensslconf.html
[14] https://www.phildev.net/ssl/creating_ca_csrs.html
[15] https://www.openssl.org/docs/manmaster/man1/openssl-ca.html
[16] https://www.openssl.org/docs/man1.1.1/man1/req.html
[17] https://en.wikipedia.org/wiki/Public_key_infrastructure

5.2. Balanceador de Carga y NGINX HTTPs

[18] https://cloud.google.com/community/tutorials/https-load-balancing-nginx

---

[1] TLS HANDSHAKE.
[2] TLS protocolo.
[3] X.509 Certificados (Estándar).
[4] Proceso de verificación de la integridad y autenticidad de las firmas digitales.
[5] Proceso de verificación de Claves públicas de CA.
[6] Cadenas de CA.
[7] DN Distinguished Names definición.
[8] Generación de un certificado CA (en esencia es lo mismo que el siguiente).
[9] Generación de un certificado autofirmado.
[10] TLS protocolo.
[11] Proceso de creación de la CA.
[12] Fichero de configuración de la CA.
[13] Otro ejemplo de creación de CA.
[14] Creación de ficheros CSR Certificate Signing Request.
[15] Manpage de openssl CA.
[16] Manpage de openssl parámetro REQ.
[17] Definición de PKI Public Key Infraestructure.
[18] Configuraciones del balanceador

[19] https://nginx.org/en/docs/http/configuring_https_servers.html
[20] https://nginx.org/en/docs/http/ngx_http_ssl_module.html

5.3. IPTABLES

[21] https://www.linuxtopia.org/Linux_Firewall_iptables/x1692.html
[22] https://w3.ual.es/~vruiz/Docencia/Apuntes/Networking/Security/Filtrado_Paquetes/index.html
[23] https://linuxconfig.org/how-to-make-iptables-rules-persistent-after-reboot-on-linux

[19] Configuraciones de https NGINX
[20] Configuraciones de https NGINX - Módulo SSL
[21] Conntrack de iptables
[22] Reglas de Iptables
[23] Iptables persistent