



SERVIDORES WEB DE ALTAS PRESTACIONES

PRÁCTICA 4: Asegurar la granja web

Autor

Miguel Ángel Pérez Díaz



Escuela Técnica Superior de Ingenierías Informática y
de Telecomunicación

—
Granada, 2020

1. Crear e instalar en la máquina M1 un certificado SSL autofirmado para configurar el acceso HTTPS al servidor. Se debe comprobar que el servidor acepta tanto el tráfico HTTP como el HTTPS.

Para llevar a cabo la tarea propuesta debemos seguir una serie de pasos que voy a detallar a continuación:

- **Activamos el módulo SSL de Apache con el comando:**

```
sudo a2enmod ssl
```

```
miguel444@m1:~$ sudo a2enmod ssl
[sudo] password for miguel444:
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 certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

- Como bien indica la salida de ejecutar el comando anterior: **reiniciamos Apache:**

```
sudo service apache2 restart
```

```
miguel444@m1:~$ sudo service apache2 restart
```

- **Creamos el directorio donde vamos a generar el correspondiente certificado SSL:**

```
sudo mkdir /etc/apache2/ssl
```

```
miguel444@m1:~$ sudo mkdir /etc/apache2/ssl
```

- Finalmente **generamos el certificado SSL** indicándole los siguientes parámetros:

```
sudo openssl req -x509 -nodes -days 365 -newkey
rsa:2048 -keyout /etc/apache2/ssl/apache.key -out
/etc/apache2/ssl/apache.crt
```

- **openssl:** es la herramienta de línea de comandos básica para crear y administrar certificados, claves, y otros archivos de OpenSSL.
- **req:** este subcomando especifica que deseamos usar la administración de la solicitud de firma de certificados (CSR) X.509. El “X.509” es un estándar de infraestructura de claves públicas al que se adecuan SSL y TLS para la administración de claves y certificados a través de él. Queremos crear un nuevo certificado X.509, por lo que usaremos este subcomando.
- **-x509:** modifica aún más el subcomando anterior al indicar a la utilidad que deseamos crear un certificado autofirmado en lugar de generar una solicitud de firma de certificados, como normalmente sucede.
- **-nodes:** indica a OpenSSL que omita la opción para proteger nuestro certificado con una frase de contraseña. Necesitamos que Apache pueda leer el archivo, sin intervención del usuario, cuando se inicie el servidor. Una frase de contraseña evitaría que esto suceda porque tendríamos que ingresarla tras cada reinicio.
- **-days 365:** esta opción establece el tiempo durante el cual el certificado se considerará válido. En este caso, lo configuramos por un año.
- **-newkey rsa:2048:** especifica que deseamos generar un nuevo certificado y una nueva clave al mismo tiempo. No creamos la clave que se requiere para firmar el certificado en un paso anterior, por lo que debemos crearla junto con el certificado. La parte rsa:2048 le indica que cree una clave RSA de 2048 bits de extensión.
- **-keyout:** esta línea indica a OpenSSL dónde colocar el archivo de clave privada generado que estamos creando.
- **-out:** indica a OpenSSL dónde colocar el certificado que creamos.

```
miguel444@m1:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/ap
ache.key -out /etc/apache2/ssl/apache.crt
Can't load /home/miguel444/.rnd into RNG
139988017807808:error:2406F079:random number generator:RAND_load_file:Cannot open file:../crypto/ran
d/randfile.c:88:Filename=/home/miguel444/.rnd
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/etc/apache2/ssl/apache.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) []:mapd0004
Email Address []:mapd0004@correo.ugr.es
miguel444@m1:~$ _
```

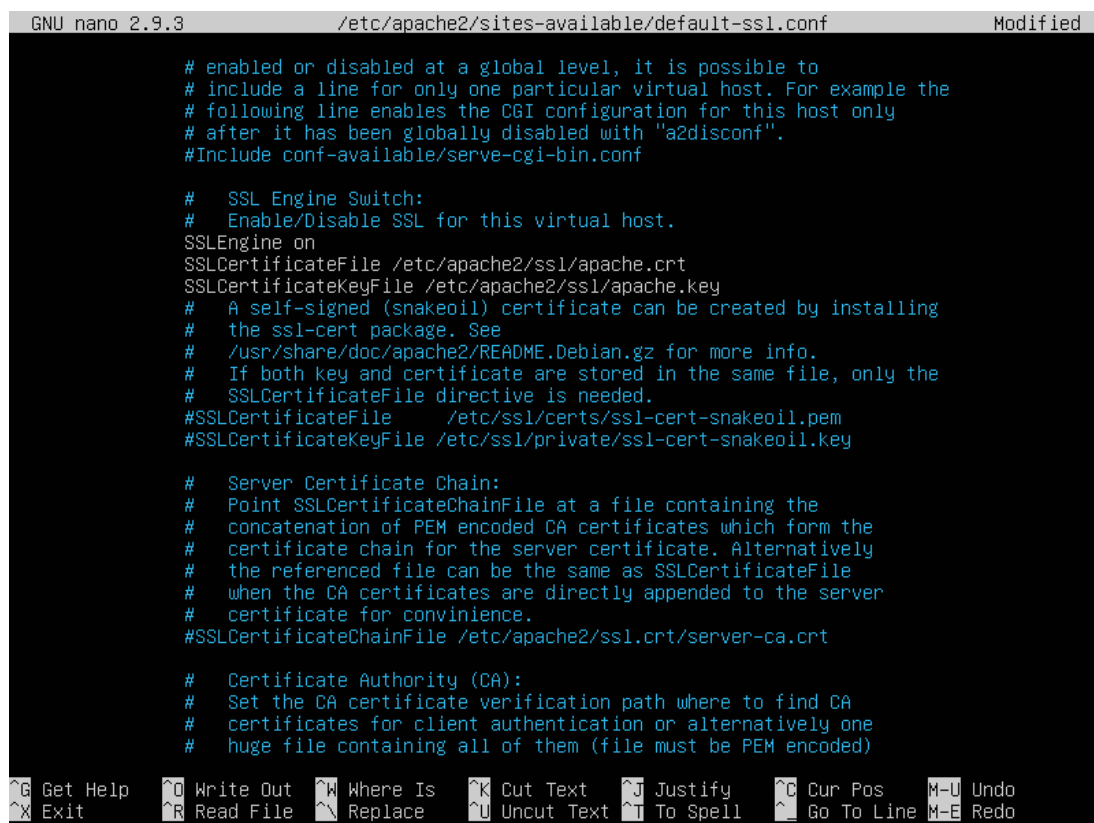
Se puede observar que se me ha pedido los datos citados en el guion de prácticas para configurar el certificado del dominio y se han rellenado con mis pertinentes datos:

- Nombre de país: ES
- Provincia: Granada
- Localidad: Granada
- Organización: SWAP
- Organización sección: P4
- Nombre: mapd0004
- Email: mapd0004@correo.ugr.es

- El siguiente paso sería **modificar el fichero `/etc/apache2/sites-available/default-ssl.conf` y agregar la ruta de los certificados anteriormente generados**, para ello se añade:

```
SSLCertificateFile /etc/apache2/ssl/apache.crt
```

```
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```



```
GNU nano 2.9.3 /etc/apache2/sites-available/default-ssl.conf Modified

# 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.
#SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
#SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
```

Se puede observar que ha sido necesario comentar las líneas de más abajo donde se especificaba otra ruta para los certificados (También podría haberse modificado la ruta existente por la nueva).

- Por último, solamente nos **faltaría activar el servicio *default-ssl* y reiniciar Apache:**

```
sudo a2ensite default-ssl  
  
service apache2 reload
```

```
miguel444@m1:~$ sudo a2ensite default-ssl  
Enabling site default-ssl.  
To activate the new configuration, you need to run:  
    systemctl reload apache2  
miguel444@m1:~$ service apache2 reload  
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====  
Authentication is required to reload 'apache2.service'.  
Authenticating as: Miguel Angel Perez Diaz (miguel444)  
Password:  
==== AUTHENTICATION COMPLETE ====  
miguel444@m1:~$ _
```

Con esto último ya tendríamos todo lo necesario para hacer peticiones a nuestro servidor tanto por HTTP como por HTTPS, tras haber generado nuestro certificado autofirmado.

Para ello vamos a realizar peticiones a la máquina M1 desde M4, siendo esta última mi máquina anfitriona. Para llevar a cabo las peticiones por HTTPS es necesario añadir el parámetro *-k* a la llamada de *cURL* como se observa en la siguiente imagen:

```
curl -k https://192.168.56.12/index.html  
  
curl http://192.168.56.12/index.html
```

```
mapd0004@LAPTOP-9FUP9E1A: ~  
mapd0004@LAPTOP-9FUP9E1A:~$ curl -k https://192.168.56.12/index.html  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.12/index.html  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ _
```

Por seguir con la dinámica actual se va a continuar con el primero de los ejercicios opcionales propuestos, por tanto para este apartado debemos copiar el certificado generado tanto a la máquina M2 como al balanceador M3 y comprobar que M3 puede balancear correctamente tanto el tráfico HTTP como HTTPS.

Para copiar el certificado generado a las dos máquinas vamos a utilizar el comando *scp* desde la máquina M1, por lo que bastaría con:

Para la máquina M2

```
sudo scp /home/miguel444/apache.crt
miguel444@192.168.56.102:/home/miguel444/apache.crt

sudo scp /home/miguel444/apache.key
miguel444@192.168.56.102:/home/miguel444/apache.key
```

```
miguel444@m1:~$ sudo scp /etc/apache2/ssl/apache.crt miguel444@192.168.56.102:/home/miguel444/apache
.crt
miguel444@192.168.56.102's password:
apache.crt                                100% 1428    77.8KB/s   00:00
miguel444@m1:~$ sudo scp /etc/apache2/ssl/apache.key miguel444@192.168.56.102:/home/miguel444/apache
.key
miguel444@192.168.56.102's password:
apache.key                                100% 1704    1.1MB/s   00:00
miguel444@m1:~$
```

Para el balanceador M3

```
sudo scp /home/miguel444/apache.crt
miguel444@192.168.56.104:/home/miguel444/apache.crt

sudo scp /home/miguel444/apache.key
miguel444@192.168.56.104:/home/miguel444/apache.key
```

```
miguel444@m1:~$ sudo scp /etc/apache2/ssl/apache.crt miguel444@192.168.56.104:/home/miguel444/apache
.crt
The authenticity of host '192.168.56.104 (192.168.56.104)' can't be established.
ECDSA key fingerprint is SHA256:YX9qB+ArIob67/mq2tiVRjgp7MBQPzRjLWUraDFNlpc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.104' (ECDSA) to the list of known hosts.
miguel444@192.168.56.104's password:
apache.crt                                100% 1428    2.0MB/s   00:00
miguel444@m1:~$ sudo scp /etc/apache2/ssl/apache.key miguel444@192.168.56.104:/home/miguel444/apache
.key
miguel444@192.168.56.104's password:
apache.key                                100% 1704    2.4MB/s   00:00
miguel444@m1:~$ _
```

Tras copiar el certificado tanto a la máquina M2 como al balanceador M3 debemos seguir lo mismos pasos que para M1 en M2:

```
sudo mkdir /etc/apache2/ssl

sudo mv /home/miguel444/apache* /etc/apache2/ssl

sudo a2enmod ssl & sudo service apache2 restart

sudo nano /etc/apache2/sites-available/default-ssl.conf

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

sudo a2ensite default.ssl

Sudo service apache2 reload
```

Máquina M2:

```
miguel444@m2:~$ sudo mkdir /etc/apache2/ssl
[sudo] password for miguel444:
miguel444@m2:~$ sudo mv apache* /etc/ap
apache2/      apm/      apparmor/      apparmor.d/  apport/      apt/
miguel444@m2:~$ sudo mv apache* /etc/apache2/ssl/
miguel444@m2:~$ sudo a2enmod ssl & sudo service apache2 restart
[1] 1870
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 certificates.
```

```
# 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.
#SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
#SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convenience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)

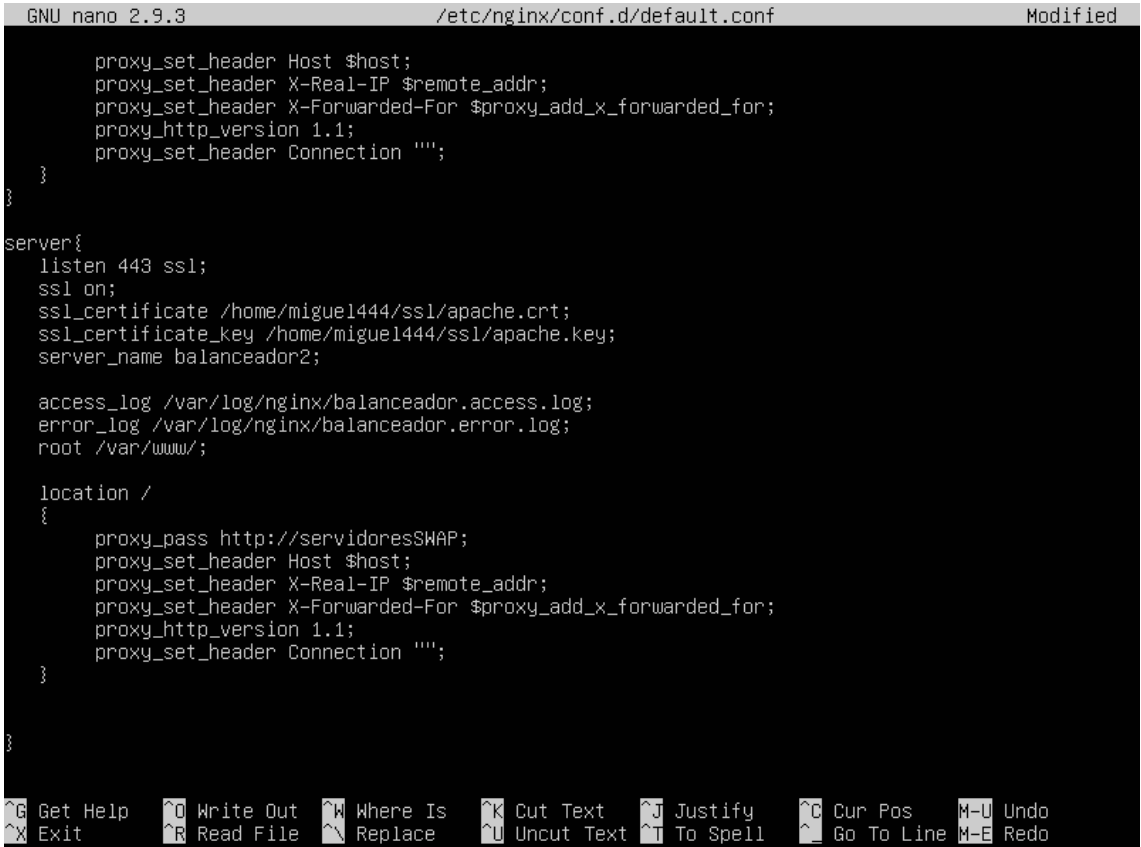
miguel444@m2:~$ sudo a2ensite default.ssl
ERROR: Site default.ssl does not exist!
miguel444@m2:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
    systemctl reload apache2
miguel444@m2:~$ sudo service apache2 reload
miguel444@m2:~$
```

Además, para el balanceador debemos crear un nuevo 'server' en el archivo de configuración de Nginx pero en este caso debemos introducir los datos correspondientes a HTTPS y el certificado:

```
listen 443 ssl;

ssl on;

ssl_certificate /home/usuario/ssl/apache.crt;
ssl_certificate_key /home/usuario/ssl/apache.key;
```



```
GNU nano 2.9.3 /etc/nginx/conf.d/default.conf Modified

    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 "";
}
}
server{
    listen 443 ssl;
    ssl on;
    ssl_certificate /home/miguel444/ssl/apache.crt;
    ssl_certificate_key /home/miguel444/ssl/apache.key;
    server_name balanceador2;

    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 "";
    }
}

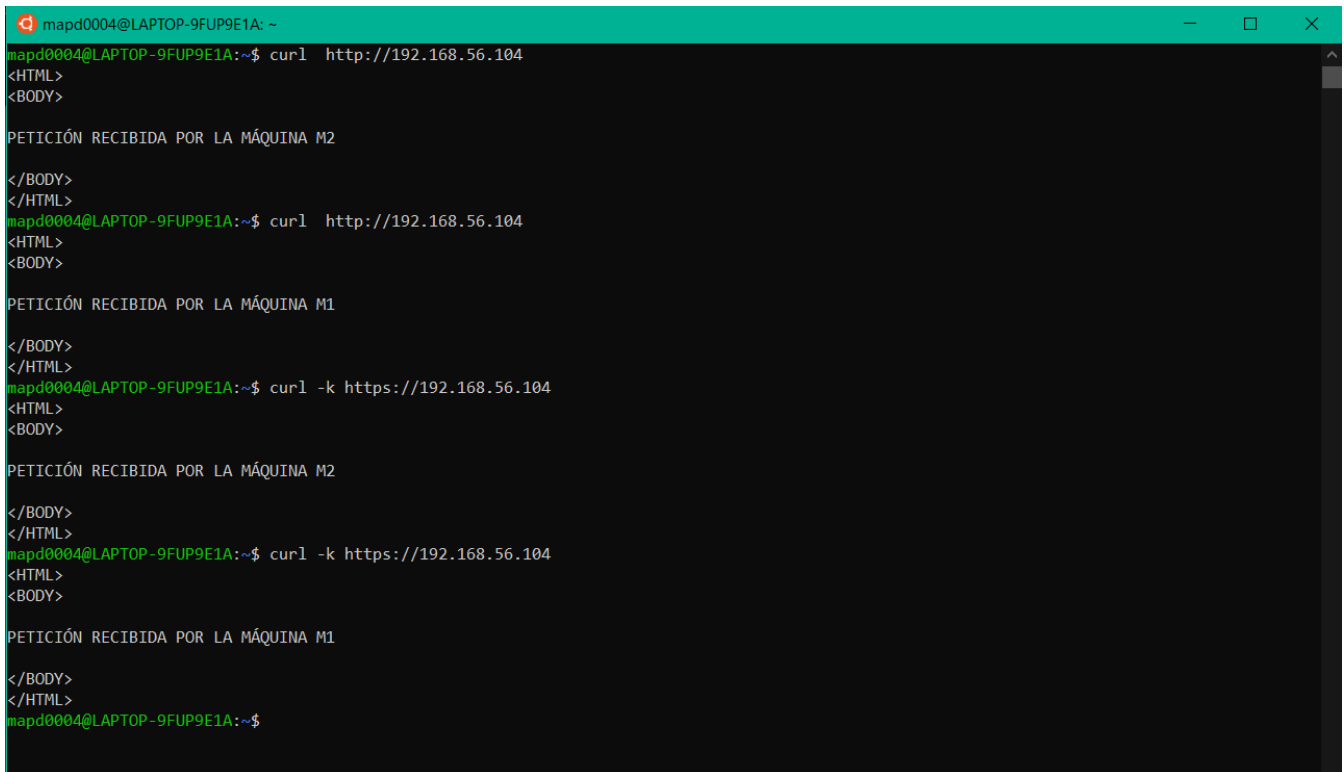
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^_ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line M-E Redo
```

Una vez disponemos del certificado autofirmado tanto en los servidores web (M1 y M2) como en la máquina balanceadora (M3) vamos a comprobar el correcto funcionamiento del balanceador haciéndole peticiones tanto HTTP como HTTPS y observar si se obtienen las respuestas deseadas a dichas peticiones.

Para ello como se indicó anteriormente utilizaremos la herramienta cURL pero en este caso hacia el balanceador desde nuestra máquina anfitrión:

```
curl -k https://192.168.56.104
```

```
curl http://192.168.56.104
```



```
mapd0004@LAPTOP-9FUP9E1A: ~  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl -k https://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl -k https://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$
```

2. Configurar las reglas del cortafuegos con IPTABLES en uno de los servidores web finales (M1 o M2) para asegurarlo, permitiendo el acceso por los puertos de HTTP y HTTPS a dicho servidor. Como se indica, esta configuración se hará en una de las máquinas servidoras finales (p.ej. en la máquina M1), y se debe poner en un script con las reglas del cortafuegos.

En esta sección debemos crear un script con reglas IPTABLES para permitir el tráfico HTTP y HTTPS en nuestra máquina M1. A continuación voy a detallar un poco las reglas utilizadas en el script para después comprobar su funcionamiento y ver si se obtiene una respuesta correcta del cortafuegos.

Para crear el script se ha generado un fichero en formato .sh y se ha declarado en la primera línea que sea ejecutable en el terminal:

```
sudo nano script_iptables.sh

#!/bin/bash
```

Las reglas utilizadas en el script son las siguientes:

```
GNU nano 2.9.3 script_iptables.sh Modified
#!/bin/bash

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

#(2) Denegar todo el tráfico entrante
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A INPUT -m state --state NEW,ESTABLISHED -j ACCEPT

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

#(4) Permitir acceso SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

#(5) Permitir acceso a HTTP
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

#(6) Permitir acceso a HTTPS
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT_

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^M Replace ^U Uncut Text ^T To Linter ^_ Go To Line M-E Redo
```

Damos permisos de ejecución al script con *chmod a+x script_iptables.sh* y ejecutamos el script con *sudo ./script_iptables.sh*.

Vamos a comprobar si el script creado aplica realmente las correspondientes reglas del cortafuegos, para ello podemos usar el siguiente comando para ver las reglas actuales:

```
iptables -L -n -v
```

```
root@m1:/home/miguel444# iptables -L -v -n
Chain INPUT (policy ACCEPT 1045 packets, 81157 bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain OUTPUT (policy ACCEPT 1429 packets, 103K bytes)
 pkts bytes target    prot opt in     out     source                   destination
root@m1:/home/miguel444# ./script_iptables.sh
root@m1:/home/miguel444# iptables -L -v -n
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination      state NEW,ESTABLISHED
    2   142 ACCEPT    all  --  *      *        0.0.0.0/0               0.0.0.0/0
    0     0 ACCEPT    all  --  lo     *        0.0.0.0/0               0.0.0.0/0
    0     0 ACCEPT    tcp  --  *      *        0.0.0.0/0               0.0.0.0/0      tcp dpt:22
    0     0 ACCEPT    tcp  --  *      *        0.0.0.0/0               0.0.0.0/0      tcp dpt:80
    0     0 ACCEPT    tcp  --  *      *        0.0.0.0/0               0.0.0.0/0      tcp dpt:443
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain OUTPUT (policy ACCEPT 2 packets, 120 bytes)
 pkts bytes target    prot opt in     out     source                   destination
    2   142 ACCEPT    all  --  *      lo       0.0.0.0/0               0.0.0.0/0
    0     0 ACCEPT    tcp  --  *      *        0.0.0.0/0               0.0.0.0/0      tcp spt:22
    0     0 ACCEPT    tcp  --  *      *        0.0.0.0/0               0.0.0.0/0      tcp spt:80
    0     0 ACCEPT    tcp  --  *      *        0.0.0.0/0               0.0.0.0/0      tcp spt:443
root@m1:/home/miguel444#
```

Tras finalizar la tarea vamos a continuar con el ejercicio opcional correspondiente a esta parte del cortafuegos en la que debemos configurar M1 y M2 para que solo acepte peticiones del balanceador M3 y que el balanceador solo reciba peticiones HTTP y HTTPS. Además habrá que guardar dicha configuración del cortafuegos para que se ejecute en cada máquina durante el arranque del sistema.

Inicialmente comenzaremos configurando tanto M1 como M2 para que solo acepten peticiones provenientes de M3, para ello debemos modificar el script generado para M1 y crear uno nuevo para M2 en donde definiremos un parámetro indicando la dirección IP del balanceador M3.

En este nuevo script hay que especificar tanto la IP de origen como de destino, en nuestro caso la del balanceador. Para especificar la dirección IP se añade un nuevo parámetro a las reglas anteriormente vistas: -s (IP origen) y -d (IP destino).

```
GNU nano 2.9.3 script_iptables.sh Modified
#!/bin/sh

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

#(2) Política por defecto: denegar todo el tráfico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

#(3) Permitimos solo el tráfico con el balanceador por los puertos de SSH,HTTP y HTTPS
iptables -A INPUT -s 192.168.56.104 -p tcp -m multiport --dports 22,80,443 -j ACCEPT
iptables -A OUTPUT -d 192.168.56.104 -p tcp -m multiport --sports 22,80,443 -j ACCEPT
```

Al igual que antes daremos permisos de ejecución al script y lo ejecutaremos para después comprobar si se han añadido las reglas correctamente.

```
miguel444@m2:~$ sudo ./script_iptables.sh
miguel444@m2:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source               destination
  0      0 ACCEPT    tcp  --  *      *       192.168.56.104        0.0.0.0/0             multiport d
ports 22,80,443

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source               destination
  0      0 ACCEPT    tcp  --  *      *       0.0.0.0/0            192.168.56.104        multiport s
ports 22,80,443
miguel444@m2:~$ _
```

Vamos a comprobar si la máquina M2 solo acepta realmente peticiones desde M3 y deniega peticiones desde mi máquina anfitrión.

MÁQUINA ANFITRIÓN

```
mapd0004@LAPTOP-9FUP9E1A: ~  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.102  
curl: (7) Failed to connect to 192.168.56.102 port 80: Connection refused  
mapd0004@LAPTOP-9FUP9E1A:~$
```

MÁQUINA BALANCEADORA M3

```
miguel1444@m3:~$ curl http://192.168.56.102  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
miguel1444@m3:~$ curl -k https://192.168.56.102  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
miguel1444@m3:~$
```

Una vez se ha comprobado que funciona correctamente para M2 podemos aplicar el mismo script para M1 y añadir las mismas reglas, así solo aceptarán peticiones del balanceador M3.

```
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.12  
curl: (7) Failed to connect to 192.168.56.12 port 80: Connection refused  
mapd0004@LAPTOP-9FUP9E1A:~$
```

```
miguel1444@m3:~$ curl http://192.168.56.12  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
miguel1444@m3:~$ curl -k https://192.168.56.12  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
miguel1444@m3:~$
```

Terminada esta parte vamos a proceder a añadir las correspondientes reglas de cortafuegos en el balanceador para que solo acepte peticiones HTTP y HTTPS. Al igual que para los servidores de nuestra granja web, debemos crear un script con las reglas a añadir para luego ejecutarlo:

```
GNU nano 2.9.3 script_iptables.sh

#!/bin/sh

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

#(2) Denegar todo el tráfico entrante
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A INPUT -m state --state NEW,ESTABLISHED -j ACCEPT

#(3) Permitir tráfico HTTP y HTTPS
iptables -I INPUT -p tcp -m multiport --dports 80,443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -I OUTPUT -p tcp -m multiport --sports 80,443 -m state --state NEW,ESTABLISHED -j ACCEPT

[ Read 19 lines ]
Get Help  Write Out  Where Is  Cut Text  Justify  Cur Pos  M-U Undo
Exit      Read File  Replace  Uncut Text To Linter Go To Line M-E Redo
```

Posteriormente le damos permisos de ejecución y lo ejecutamos para observar si se han aplicado las reglas correctamente:

```
miguel444@m3:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination
 43 3690 ACCEPT    tcp  --  *      *       0.0.0.0/0         0.0.0.0/0         multiport d
ports 80,443 state NEW,ESTABLISHED
 98 14534 ACCEPT    all  --  *      *       0.0.0.0/0         0.0.0.0/0         state NEW,E
STABLISHED

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 115 packets, 13893 bytes)
 pkts bytes target    prot opt in     out     source            destination
 34 6915 ACCEPT    tcp  --  *      *       0.0.0.0/0         0.0.0.0/0         multiport s
ports 80,443 state NEW,ESTABLISHED
miguel444@m3:~$ _
```

A continuación, veremos si el balanceador realiza su función correctamente, para ello realizaremos peticiones a él por HTTP y HTTPS.

```
mapd0004@LAPTOP-9FUP9E1A: ~  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl http://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl -k https://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M2  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$ curl -k https://192.168.56.104  
<HTML>  
<BODY>  
  
PETICIÓN RECIBIDA POR LA MÁQUINA M1  
  
</BODY>  
</HTML>  
mapd0004@LAPTOP-9FUP9E1A:~$
```

Para finalizar la tarea solo faltaría configurar las máquinas para que el correspondiente script se ejecute durante el arranque del sistema. Cabe mencionar que se va a describir el proceso para la máquina M1, este mismo proceso se ha repetido para el resto de máquinas: M2 y M3.

Para ello inicialmente debemos crear un nuevo servicio en `/etc/systemd/system/` llamado `script_iptables.service` en mi caso :

```
GNU nano 2.9.3 /etc/systemd/system/script_iptables.service Modified
[Unit]
Description=Configuración de firewall
After=syslog.target

[Service]
Type=forking
ExecStart=/home/miguel444/script_iptables.sh

[Install]
WantedBy=multi-user.target

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line  M-E Redo
```

En la sección `ExecStart=/home/miguel44/script_iptables.sh` se da la ruta en dónde realmente está el script a ejecutar. Y el tipo de demonio que va a ser en:

```
[Service]
Type=forking
```

Por último, la sección `[Install]` nos dice en qué runlevel se ejecutará. En el modo multiusuario o en el clásico sysv, que sería el runlevel 3.

```
[Install]
WantedBy=multi-user.target
```

Con esto ya tenemos todo listo para agregar nuestro script al inicio del sistema.

<https://www.ubuntuleon.com/2016/10/cargar-un-script-al-inicio-del-sistema.html>

Una vez hemos declarado el servicio, debemos darle los permisos necesarios para su ejecución: **rwxr-xr-x**

```
sudo chmod 744 script_iptables.sh
```

Una vez configurado todo, antes de ejecutar la nueva configuración, hay que refrescar systemd para que lea de nuevo los servicios:

```
sudo systemctl daemon-reload
```

Levantamos el servicio:

```
sudo systemctl start script_iptables.service
```

Habilitamos el servicio para que inicie durante el arranque del sistema:

```
sudo systemctl enable script_iptables.service
```

```
miguel444@m1:~$ sudo chmod 744 script_iptables.sh
miguel444@m1:~$ sudo systemctl daemon-reload
miguel444@m1:~$ systemctl start script_iptables.service
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'script_iptables.service'.
Authenticating as: Miguel Angel Perez Diaz (miguel444)
Password:
==== AUTHENTICATION COMPLETE ====
miguel444@m1:~$ sudo systemctl enable script_iptables.service
Created symlink /etc/systemd/system/multi-user.target.wants/script_iptables.service → /etc/systemd/s
ystem/script_iptables.service.
miguel444@m1:~$ _
```

Finalmente comprobamos que está activo al iniciar la máquina virtual:

```
Ubuntu 18.04.4 LTS m1 tty1

m1 login: miguel444
Password:
Last login: Mon Apr 27 17:11:35 UTC 2020 on tty1
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

Pueden actualizarse 14 paquetes.
0 actualizaciones son de seguridad.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection
or proxy settings

miguel444@m1:~$ sudo iptables -L -n -v
[sudo] password for miguel444:
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT    tcp  --  *      *       192.168.56.104        0.0.0.0/0            multiport d
ports 22,80,443

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 40 packets, 2880 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT    tcp  --  *      *       0.0.0.0/0            192.168.56.104      multiport s
ports 22,80,443
miguel444@m1:~$ _
```