

# PRÁCTICA 4: ASEGURAR LA GRANJA WEB

**Iñaki Melguizo Marcos**



## 1. INTRODUCCIÓN

En esta práctica, nos centraremos en la seguridad de la granja web que hemos estado conformando durante las prácticas de la asignatura. A modo de recordatorio, tenemos instaladas tres máquinas: m1-imm98 y m2-imm98 que contienen la misma información, es decir, podríamos decir que m2-imm98 es un servidor de respaldo. m3-imm98 es la máquina que actúa como balanceador de carga software.

Primeramente instalaremos un certificado SSL para que podamos acceder mediante el protocolo HTTPS a los servidores y en segundo lugar configuraremos las reglas del cortafuegos para proteger a la granja web.

## 2. CERTIFICADO AUTOFIRMADO SSL

Primero vamos a generar un certificado SSL que sirve para dar seguridad a los usuarios que accedan a una página web, es decir una forma de decirles que el sitio es auténtico para que puedan introducir datos sensibles.

Hay varias formas de obtener certificados SSL pero nosotros lo haremos creando nuestros propios certificados SSL auto-firmados usando la herramienta openssl. Para generar estos certificados SSL autofirmados primero debemos activar el módulo SSL de Apache. Por lo tanto ejecutaremos en la máquina m1-imm98:

*sudo a2enmod ssl*

```
imm98@m1-imm98:~$ sudo a2enmod
[sudo] password for imm98:
Your choices are: access_compat actions alias allowmethods asis auth_basic auth_
digest auth_form authn_anon authn_core authn_dbd authn_dbm authn_file authn_soca
che authnz_fcgi authnz_ldap authz_core authz_dbd authz_dbm authz_groupfile authz
_host authz_owner authz_user autoindex buffer cache cache_disk cache_socache cer
n_meta cgi cgid charset_lite data dav dav_fs dav_lock dbd deflate dialup dir dum
p_io echo env expires ext_filter file_cache filter headers heartbeat heartmonito
r http2 ident imagemap include info lbmethod_bybusyness lbmethod_byrequests lbme
thod_bytraffic lbmethod_heartbeat ldap log_debug log_forensic lua macro mime mim
e_magic mpm_event mpm_prefork mpm_worker negotiation proxy proxy_ajp proxy_balan
cer proxy_connect proxy_express proxy_fcgi proxy_fdpass proxy_ftp proxy_hcheck p
roxy_html proxy_http proxy_http2 proxy_scgi proxy_wstunnel ratelimit reflector r
emoteip reqtimeout request rewrite sed session session_cookie session_crypto ses
sion_dbd setenvif slotmem_plain slotmem_shm socache_dbm socache_memcache socache
_shmcb spelling ssl status substitute suexec unique_id userdir usertrack vhost_al
ias xml2enc
Which module(s) do you want to enable (wildcards ok)?
```

Reiniciamos el servicio de apache con la orden:

*sudo service apache2 restart*

```
imm98@m1-imm98:~$ sudo service apache2 restart
```

Creamos el directorio donde se ubicarán nuestros certificados ssl que será el **/etc/apache2/ssl**

```
imm98@m1-imm98:~$ sudo mkdir /etc/apache2/ssl
```

Generamos el certificado y la clave del certificado con la orden:

*sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout  
/etc/apache2/ssl/apache\_imm98.key -out /etc/apache2/ssl/apache\_imm98.crt*

```
imm98@m1-imm98:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/ssl/apache_imm98.key -out /etc/apache2/ssl/apache_imm98.crt
```

E introducimos los valores correspondientes en los diferentes campos del certificado

```
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) []:imm98
Email Address []:imm98@correo.ugr.es
```

Como **opciones avanzadas** he introducido la orden:

```
openssl -x509 -text -noout -in apache_imm98.crt
```

que te permite ver el contenido de un certificado, en este caso `apache_imm98.crt` en texto plano

```
imm98@m1-imm98:/etc/apache2/ssl$ openssl x509 -text -noout -in apache_imm98.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            6c:af:1f:78:1b:54:d6:be:3e:24:25:08:2e:87:30:d1:96:be:6b:ea
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = ES, ST = Granada, L = Granada, O = SWAP, OU = P4, CN = imm98, emailAddress = imm98@correo.ugr.es
        Validity
            Not Before: Apr 27 14:59:53 2021 GMT
            Not After : Apr 27 14:59:53 2022 GMT
        Subject: C = ES, ST = Granada, L = Granada, O = SWAP, OU = P4, CN = imm98, emailAddress = imm98@correo.ugr.es
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public-Key: (2048 bit)
            Modulus:
                00:b6:02:50:be:59:e2:54:ed:b7:79:26:cb:d2:cf:
                b2:af:f2:84:66:19:97:18:cf:8d:33:9b:52:36:c0:
                71:09:93:bd:36:13:e5:49:20:59:02:5b:cf:b1:59:
                b6:ed:df:9c:c7:11:43:0a:78:5d:4d:3c:eb:c8:dc:
                e6:ff:30:ef:ae:0c:e2:7d:cc:16:ff:d2:b8:cf:73:
                9f:b0:bc:9f:e2:4d:7a:36:c0:b5:90:b2:0b:43:8a:
                ab:9d:c0:fc:a2:ef:87:84:70:cc:eb:5a:67:ab:c4:
                3d:36:90:c2:04:11:3c:34:17:10:0c:4d:42:ad:4b:
                a1:f6:6c:cb:bc:da:b8:83:93:75:5d:38:78:76:b7:
                37:86:12:13:c8:3c:b2:f4:c2:5f:b8:be:9d:f1:54:
                3f:eb:ed:e9:7f:6a:4c:ed:32:4c:4a:3b:e7:db:25:
                88:f7:92:57:e9:41:88:59:52:58:54:a1:8a:bb:30:
                f2:bd:84:96:10:d6:be:aa:38:3c:03:ff:5c:23:d8:
                ff:62:b2:6a:fb:25:ac:3a:fe:58:6b:ea:34:62:a3:
                34:cd:4b:1b:46:d9:03:9e:ea:05:33:18:5b:ed:4b:
                b2:15:c2:b8:2f:b1:15:21:d5:6c:df:79:f0:f6:f0:
                3d:47:34:45:2f:87:8c:9a:99:c4:d8:76:32:ed:91:
                ea:c1
            Exponent: 65537 (0x10001)
        X509v3 extensions:
```

La otra opción avanzada que he introducido es

```
openssl req -new -newkey rsa:2048 -nodes -out certificado_imm98.csr -keyout
certificado_imm98.key
```

Este comando genera dos nuevos archivos, el archivo de la clave privada necesaria para el cifrado del certificado SSL y lo que se llama un archivo de Solicitud de Firma de Certificado (con extensión `.csr`) que se aplica al certificado SSL.

```
imm98@m1-imm98:/etc/apache2/ssl$ openssl req -new -newkey rsa:2048 -nodes -out certificado_imm98.csr -keyout certificado_imm98.key
```

### 3. APACHE CON CERTIFICADO SSL

Primero vamos a editar el fichero de configuración de apache del sitio **default-ssl.conf**:

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

```
imm98@m1-imm98:~$ sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Le vamos a indicar la ubicación tanto de nuestro certificado SSL como de nuestra clave privada del certificado:

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache_imm98.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache_imm98.key
```

Por último vamos a activar el sitio default-ssl con la orden

```
sudo a2ensite default-ssl.conf
```

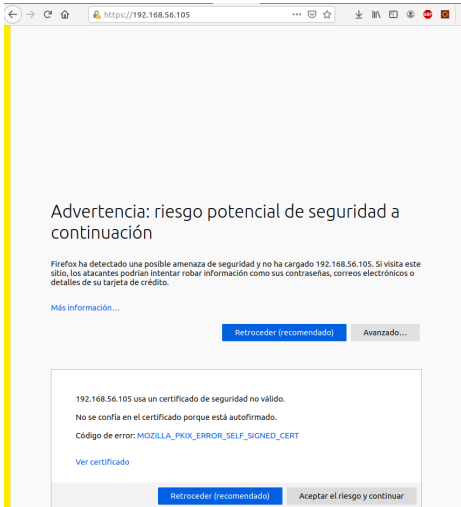
```
imm98@m1-imm98:/etc/apache2/sites-available$ sudo a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
systemctl reload apache2
```

Y reiniciamos el servicio de apache para guardar los cambios con la orden:

```
sudo systemctl reload apache2.service:
```

```
imm98@m1-imm98:/etc/apache2/sites-available$ sudo systemctl reload apache2.service
```

Ahora accedemos al sitio web mediante el protocolo HTTPS y vemos que se nos muestra nuestro certificado como era de prever:



Certificado

imm98

---

**Nombre del asunto**

País	ES
Estado/Provincia	Granada
Localidad	Granada
Organización	SWAP
Unidad organizativa	P4
Nombre común	imm98
Dirección de correo electrónico	imm98@correo.ugr.es

---

**Nombre del emisor**

País	ES
Estado/Provincia	Granada
Localidad	Granada
Organización	SWAP
Unidad organizativa	P4
Nombre común	imm98
Dirección de correo electrónico	imm98@correo.ugr.es

Para hacer las peticiones mediante la herramienta curl ejecutamos:

```
curl -k https://192.168.56.105 /index.html
```

```
inaki@inaki-N501VW:~$ curl -k https://192.168.56.105/index.html
<HTML>
  <BODY>
    Web de la pagina m1
  </BODY>
</HTML>
```

Y vemos que os muestra correctamente la página.

Nuestro objetivo es que la granja nos permita usar HTTPS, por lo que debemos configurar el balanceador para que también acepte tráfico HTTPS. Por ello no generaremos más certificados sino que copiaremos el certificado y la clave que hemos generado en la máquina m1-imm98 a las máquinas m2-imm98 y m3-imm98. Para realizar esta función de copia usaremos scp. Primeramente lo copiaremos a la máquina m2-imm98. Para ello ejecutaremos el comando:

```
sudo scp apache_imm98.crt imm98@192.168.56.104:/home/imm98/apache_imm98.crt
```

```
imm98@m1-imm98:/etc/apache2/ssl$ sudo scp apache_imm98.crt imm98@192.168.56.104:
/home/imm98/apache_imm98.crt
imm98@192.168.56.104's password:
apache_imm98.crt                                100% 1411      1.8MB/s   00:00
```

Vemos que el proceso de copia se ha realizado correctamente y ahora en la máquina m2-imm98 crearemos el directorio /etc/apache2/ssl que será donde se alojarán tanto la clave privada como el certificado. Lo crearemos con la orden:

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

```
imm98@m2-imm98:~$ sudo mkdir /etc/apache2/ssl
[sudo] password for imm98:
```

Ahora copiamos también la clave privada de la máquina m1-imm98 a la máquina m2-imm98 mediante la orden:

```
sudo scp apache_imm98.key imm98@192.168.56.104:/home/imm98/apache_imm98.key
```

```
imm98@m1-imm98:/etc/apache2/ssl$ sudo scp apache_imm98.key imm98@192.168.56.104:
/home/imm98/apache_imm98.key
imm98@192.168.56.104's password:
apache_imm98.key                                100% 1704      948.1KB/s   00:00
```

Una vez que tenemos ya tanto el certificado en el directorio home de la máquina m2-imm98 los movemos al directorio que habíamos creado, el directorio /etc/apache2/ssl con la orden:

```
sudo mv apache_imm98.* /etc/apache2/ssl
```

```
imm98@m2-imm98:~$ sudo mv apache_imm98.* /etc/apache2/ssl/
```

Y hacemos lo que tuvimos que hacer en la máquina m1-imm98, habilitar el módulo SSL de Apache y reiniciar su servicio con la orden:

```
sudo a2enmod ssl
```

```
imm98@m2-imm98:~$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
```

Vamos a editar el fichero de configuración de apache del sitio **default-ssl.conf**:

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

```
imm98@m2-imm98:~$ sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Le vamos a indicar la ubicación tanto de nuestro certificado SSL como de nuestra clave privada del certificado:

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache_imm98.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache_imm98.key
```

Por último vamos a activar el sitio default-ssl con la orden

```
sudo a2ensite default-ssl.conf
```

y reiniciamos el servicio de apache con la orden:

```
sudo a2ensite default-ssl.conf
```

```
imm98@m2-imm98:~$ sudo a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
    systemctl reload apache2
imm98@m2-imm98:~$ sudo systemctl reload apache2.service
```

Ahora, copiamos tanto la clave como el propio certificado en la máquina m3-imm98

```
imm98@m1-imm98: /etc/apache2/ssl$ sudo scp apache_imm98.crt imm98@192.168.56.107: /home/imm98/apache_1
imm98.crt
The authenticity of host '192.168.56.107 (192.168.56.107)' can't be established.
ECDSA key fingerprint is SHA256:vcyuZkbKwp2FMC2ytIAWpbDADktMzRDjRJSnG26Ps2w.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.107' (ECDSA) to the list of known hosts.
imm98@192.168.56.107's password:
apache_imm98.crt          100% 1411    580.2KB/s   00:00
```

[illegible]

Como opción avanzada he introducido que el **SSLCertificateChainFile** es el `/etc/apache2/ssl/apache_imm98_intermedio.crt`

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache_imm98.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache_imm98.key
SSLCertificateChainFile /etc/apache2/ssl/apache_imm98_intermedio.crt
```

La cadena `SSLCertificateChainFile` le permite a Apache mostrar al cliente exactamente cómo se ve la relación de confianza, lo que puede ayudar a un cliente a completar los espacios en blanco entre su certificado, una raíz en la que confían y el intermedio que desconocen.



## 4. NGINX COMO BALANCEADOR PARA PETICIONES HTTPS

Lo primero que debemos hacer es modificar el archivo de configuración de Nginx, el `/etc/nginx/conf.d/default.conf`:

```
imm98@m3-imm98:~$ sudo nano /etc/nginx/conf.d/default.conf
```

y le vamos a añadir un nuevo server:

```
server{
    listen 443 ssl;
    ssl on;
    ssl_certificate /home/imm98/apache_imm98.crt;
    ssl_certificate_key /home/imm98/apache_imm98.key;

    server_name balanceador_imm98;
    access_log /var/log/nginx/balanceador_imm98.access.log;
    error_log /var/log/nginx/balanceador_imm98.error.log;
    root /var/www/;

    location /
    {
        proxy_pass http://balanceo_imm98;
        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 "";
    }
}
```

Y reiniciamos el servicio de NginX para que se actualicen los cambios realizados en el fichero de configuración con la orden:

```
sudo systemctl restart nginx.service
```

```
imm98@m3-imm98:~$ sudo systemctl restart nginx.service
```

y con tan solo esto ya tendríamos configurado nuestro balanceador de carga para HTTPS. Lo comprobamos:

```
inaki@inaki-N501VW:~$ curl -k https://192.168.56.107
<HTML>
  <BODY>
    Web de la pagina m1
  </BODY>
</HTML>
inaki@inaki-N501VW:~$
```



Como primera opción avanzada tenemos la inclusión en el nuevo servidor de dos opciones, **ssl\_protocols** y **ssl\_ciphers** que lo que hacen es limitar para incluir solo las versiones fuertes y los cifrados de SSL / TLS. Nginx por defecto utiliza:

- ssl\_protocols TLSv1 TLSv1.1 TLSv1.2
- ssl\_ciphers HIGH:!aNULL:!MD5;

por lo que vamos a modificar este server para que restrinja las conexiones:

```
server{
    listen 443 ssl;
    ssl on;
    ssl_certificate /home/imm98/apache_imm98.crt;
    ssl_certificate_key /home/imm98/apache_imm98.key;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!MD5;
```

Como segunda opción avanzada he añadido otras dos opciones a nuestro servidor en el fichero **/etc/nginx/conf.d/default.conf** Vamos a utilizar las opciones **ssl\_session\_cache** y **ssl\_session\_timeout** con el objetivo de compartir cache 20 MB durante 10 minutos:

```
server{
    listen 443 ssl;
    ssl on;
    ssl_certificate /home/imm98/apache_imm98.crt;
    ssl_certificate_key /home/imm98/apache_imm98.key;
    ssl_session_cache shared:SSL:20m;
    ssl_session_timeout 10m;
```

## 5. IPTABLES

Para proteger nuestra granja web de accesos indebidos debemos configurar el cortafuegos que es el que permite el tráfico autorizado y deniega el resto. Para configurar el cortafuegos utilizaremos **iptables** que es una herramienta de espacio de usuario con la que se definen las reglas de filtrado de paquetes.

Para ver el estado del cortafuegos tenemos el comando:

*sudo iptables -L -n -v*

```
imm98@m1-imm98:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 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 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
```

Para eliminar todas las reglas existentes en el cortafuegos tendríamos que ejecutar todos estos comandos:

```
imm98@m1-imm98:~$ sudo iptables -F
imm98@m1-imm98:~$ sudo iptables -X
imm98@m1-imm98:~$ sudo iptables -t nat -F
imm98@m1-imm98:~$ sudo iptables -t nat -X
imm98@m1-imm98:~$ sudo iptables -t mangle -F
imm98@m1-imm98:~$ sudo iptables -t mangle -X
imm98@m1-imm98:~$ sudo iptables -P INPUT ACCEPT
imm98@m1-imm98:~$ sudo iptables -P OUTPUT ACCEPT
imm98@m1-imm98:~$ sudo iptables -L -n -v
iptables v1.6.1: can't initialize iptables table `filter': Permission denied (you must be root)
Perhaps iptables or your kernel needs to be upgraded.
imm98@m1-imm98:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 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 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
```

Si queremos denegar todo tráfico de información debemos poner que deniegue todo el INPUT, OUTPUT y FORWARD

```
imm98@m1-imm98:~$ sudo iptables -P INPUT DROP
imm98@m1-imm98:~$ sudo iptables -P OUTPUT DROP
imm98@m1-imm98:~$ sudo iptables -P FORWARD DROP
imm98@m1-imm98:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination

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
```

Ahora que ya sabemos las reglas básicas vamos a denegar todo el tráfico entrante a las máquinas M1, M2 y M3 a excepción del tráfico HTTP y HTTPS. Para ello vamos a crear un script llamado `cortafuegos_imm98.sh`

```
imm98@m1-imm98:~$ nano cortafuegos_imm98.sh
```

En este script primero vamos a borrar todas las reglas para establecer una configuración limpia, después vamos a denegar todo el tráfico entrante y posteriormente vamos a permitir el tráfico HTTP (80, 8080) y el tráfico HTTPS (443)

```
#(1) Se eliminan todas las reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

#(2) Denegar todo el trafico entrante
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP

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

#(4) Permitir el trafico por el puerto 8080 (HTTP)
iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 8080 -j ACCEPT

#(5) Permitir el trafico por el puerto 443
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT

iptables -L -n -v
```

Vamos a ejecutar el script siempre con la opción `sudo` delante y como se puede ver en la imagen de abajo se crean bien las reglas del cortafuegos para que deniegue todo el tráfico de entrada salvo el HTTP y el HTTPS:

```
imm98@m1-imm98:~$ sudo ./cortafuegos_imm98.sh
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination
    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:8080
    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 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination
    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:8080
    0     0 ACCEPT    tcp  --  *      *       0.0.0.0/0         0.0.0.0/0         tcp spt:443
```



Ejecutamos el script en m2-imm98 y vemos que, como era de prever, las reglas del cortafuegos que se crean son correctas.

```
imm98@m2-imm98:~$ sudo ./cortafuegos_imm98.sh
[sudo] password for imm98:
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    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:808
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 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    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:808
0      0 ACCEPT     tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp spt:443
```

En la máquina m3-imm98 ocurre lo mismo:

```
imm98@m3-imm98:~$ sudo ./cortafuegos_imm98.sh
[sudo] password for imm98:
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    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:808
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 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    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:808
0      0 ACCEPT     tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp spt:443
```

Ahora, como tarea avanzada vamos a permitir en las máquinas m1-imm98, m2-imm98 y m3-imm98 tanto SSH, PING y DNS así como el tráfico consigo misma, denegando el resto de servicios. Para ello tendremos que establecer las siguientes reglas que las introduciremos en un script llamado **cort\_dns\_imm98.sh**:

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

# (2) Denegar todo el trafico entrante
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP

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

# (4) Abrir el puerto 53 para permitir el acceso a DNS
iptables -A INPUT -m state --state NEW -p udp --dport 53 -j ACCEPT
iptables -A INPUT -m state --state NEW -p tcp --dport 53 -j ACCEPT

# (5) Abrir el puerto 22 para permitir el acceso por ssh
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p udp --dport 22 -j ACCEPT

#(6) Abrir el puerto 2222
iptables -A INPUT -p tcp --dport 2222 -j ACCEPT
iptables -A OUTPUT -p udp --sport 2222 -j ACCEPT

# (7) Abrir el puerto de ping
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

iptables -L -n -v
```

Para comprobar su correcto funcionamiento vamos a probar a probar a hacer una petición http, https y a hacer ping desde la máquina m2-imm98 a la máquina m1-imm98 que es donde hemos ejecutado el script de arriba.

```
imm98@m2-imm98:~$ curl http://imm98@192.168.56.105:8080
^C
imm98@m2-imm98:~$ curl -k https://imm98@192.168.56.105:8080
^C
imm98@m2-imm98:~$ ping 192.168.56.105
PING 192.168.56.105 (192.168.56.105) 56(84) bytes of data.
64 bytes from 192.168.56.105: icmp_seq=1 ttl=64 time=0.889 ms
64 bytes from 192.168.56.105: icmp_seq=2 ttl=64 time=0.798 ms
^C
--- 192.168.56.105 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.798/0.843/0.889/0.053 ms
imm98@m2-imm98:~$ _
```

Como era de prever el cortafuegos evita las peticiones http y https. En cambio si deja que se haga ping a la máquina m1-imm98. Además también nos podemos conectar mediante ssh como podemos ver en la imagen de abajo lo que nos demuestra que el cortafuegos está realizando su labor correctamente.

```
imm98@m2-imm98:~$ ssh -p 2222 imm98@192.168.56.105
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-142-generic x86_64)

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

System information as of Wed Apr 28 09:47:16 UTC 2021

System load:  0.0               Processes:           99
Usage of /:   54.4% of 8.79GB    Users logged in:    1
Memory usage: 32%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%                IP address for enp0s8: 192.168.56.105

 * Pure upstream Kubernetes 1.21, smallest, simplest cluster ops!

https://microk8s.io/

65 packages can be updated.
0 updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Apr 28 09:46:07 2021 from 192.168.56.104
imm98@m1-imm98:~$
```

Dado que también habíamos permitido el tráfico consigo misma (localhost) vamos a probar que podemos hacer una petición HTTP y HTTPS de m1-imm98 a sí misma

```
imm98@m1-imm98:~$ curl http://192.168.56.105:8080
<HTML>
  <BODY>
    Web de la pagina m1
  </BODY>
</HTML>
imm98@m1-imm98:~$ curl -k https://192.168.56.105
<HTML>
  <BODY>
    Web de la pagina m1
  </BODY>
</HTML>
```

Y vemos que se realiza correctamente la petición.

Ahora, dado que queremos tener la misma configuración que en m1-imm98 en las máquinas m2-imm98 y m3-imm98 vamos a copiar mediante el comando scp ya usado anteriormente el script usado para la configuración anterior a las máquinas m2-imm98 y m3-imm98. Para ello ejecutamos los siguientes comandos:

```
sudo scp cort_dns_imm98.sh imm98@192.168.56.104:/home/imm98/cort_dns_imm98.sh
sudo scp cort_dns_imm98.sh imm98@192.168.56.107:/home/imm98/cort_dns_imm98.sh
```

```
imm98@m1-imm98:~$ sudo scp cort_dns_imm98.sh imm98@192.168.56.104:/home/imm98/cort_dns_imm98.sh
imm98@192.168.56.104's password:
cort_dns_imm98.sh                                100% 896   629.6KB/s   00:00
imm98@m1-imm98:~$ sudo scp cort_dns_imm98.sh imm98@192.168.56.107:/home/imm98/cort_dns_imm98.sh
imm98@192.168.56.107's password:
cort_dns_imm98.sh                                100% 896   552.1KB/s   00:00
```

Ahora vamos a proceder a configurar m3-imm98 de manera que sólo m3-imm98 sea el que acepte peticiones HTTP y HTTPS mientras que m1-imm98 y m2-imm98 solo acepten peticiones de m3-imm98, el balanceador de carga. Para ello crearemos un script que llamaremos cort\_m3.sh:

```
imm98@m1-imm98:~$ sudo nano cort_m3.sh
```

En las máquinas **m1-imm98** y **m2-imm98** el contenido del script para que el cortafuegos realice las acciones descritas anteriormente debe ser el siguiente:

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

#(2) Denegar todo el trafico entrante
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP

# (3) Permitir solo peticiones de m3
iptables -I INPUT -s 192.168.56.107 -j ACCEPT
iptables -I OUTPUT -s 192.168.56.107 -j ACCEPT
```



Es decir, simplemente denegamos todo el tráfico entrante salvo el que provenga de m3-imm98, es decir, de la máquina con IP 192.168.56.107. Tras ejecutar el script vemos que las reglas del cortafuegos que se han generado son correctas:

```
imm98@m2-imm98:~$ sudo ./cort_m3.sh
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       192.168.56.107       0.0.0.0/0

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

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       192.168.56.107       0.0.0.0/0
```

En el script de configuración del cortafuegos de la máquina m3-imm98 debemos añadir que deniegue todo el tráfico de entrada excepto el tráfico por los puertos 80, 8080 y 443, es decir excepto el tráfico HTTP y HTTPS. Este sería el script con dichas sentencias:

```
m3-imm98 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 2.9.3                                cort_m3.sh

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

#(2) Denegar todo el trafico 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 el trafico por el puerto 80 (HTTP)
iptables -A INPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 8080 -j ACCEPT
# (4) Permitir el trafico por el puerto 443 (HTTPS)
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT

iptables -F -X -Z
```

Al ejecutar el script vemos que se crean las reglas correctas:

```
imm98@m3-imm98:~$ sudo ./cort_m3.sh
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
    0    0 ACCEPT     all  --  *      *        0.0.0.0/0         0.0.0.0/0         state NEW,E
ESTABLISHED
    0    0 ACCEPT     tcp  --  *      *        0.0.0.0/0         0.0.0.0/0         tcp dpt:80
state NEW,ESTABLISHED
    0    0 ACCEPT     tcp  --  *      *        0.0.0.0/0         0.0.0.0/0         tcp dpt:808
0
    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 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
    0    0 ACCEPT     tcp  --  *      *        0.0.0.0/0         0.0.0.0/0         tcp spt:80
state NEW,ESTABLISHED
    0    0 ACCEPT     tcp  --  *      *        0.0.0.0/0         0.0.0.0/0         tcp spt:808
0
    0    0 ACCEPT     tcp  --  *      *        0.0.0.0/0         0.0.0.0/0         tcp spt:443
imm98@m3-imm98:~$
```

Vamos a comprobar el correcto funcionamiento de las reglas del cortafuegos diseñadas anteriormente.

Vemos que desde la máquina m3-imm98 si se pueden hacer peticiones HTTP tanto a m1-imm98 como a m2-imm98

```
imm98@m3-imm98:~$ curl http://imm98@192.168.56.105:8080
<HTML>
    <BODY>
        Web de la pagina m1
    </BODY>
</HTML>
imm98@m3-imm98:~$ curl http://imm98@192.168.56.104
<HTML>
    <BODY>
        Web de la pagina m2
    </BODY>
</HTML>
```

Sin embargo, desde otra máquina que no sea m3-imm98 no se pueden hacer peticiones HTTP a la máquina m1-imm98 ni a la máquina m2-imm98

```
inaki@inaki-N501VW:~$ curl http://imm98@192.168.56.105:8080
^C
inaki@inaki-N501VW:~$ curl http://imm98@192.168.56.104
^C
```

Desde la máquina m3-imm98, como era de prever, se pueden hacer peticiones HTTPS tanto a la máquina m1-imm98 como a la máquina m2-imm98:

```
imm98@m3-imm98:~$ curl -k https://imm98@192.168.56.105
<HTML>
  <BODY>
    Web de la pagina m1
  </BODY>
</HTML>
imm98@m3-imm98:~$ curl -k https://imm98@192.168.56.104
<HTML>
  <BODY>
    Web de la pagina m2
  </BODY>
</HTML>
```

Sin embargo, desde otra máquina distinta a m3-imm98 no se pueden realizar peticiones HTTPS ni a m1-imm98 ni a m2-imm98:

```
inaki@inaki-N501VW:~$ curl -k https://imm98@192.168.56.104
^C
inaki@inaki-N501VW:~$ curl -k https://imm98@192.168.56.105
^C
```

En cambio desde la máquina anfitrión si podemos hacer peticiones HTTP y HTTPS a la máquina m3-imm98 que balanceará la carga a m1-imm98 y a m2-imm98. Esto es justo lo que queríamos.

```
inaki@inaki-N501VW:~$ curl http://192.168.56.107
<HTML>
  <BODY>
    Web de la pagina m1
  </BODY>
</HTML>
inaki@inaki-N501VW:~$ curl -k https://192.168.56.107
<HTML>
  <BODY>
    Web de la pagina m2
  </BODY>
</HTML>
inaki@inaki-N501VW:~$
```

## 6. CONFIGURAR EL CORTAFUEGOS AL ARRANQUE

Una vez que ya conocemos el funcionamiento de las reglas iptables e incluirlas en un script para ejecutar un conjunto de ellas secuencialmente, vamos a proceder a hacer que se inicialicen las reglas del cortafuegos en el arranque del sistema (en este caso de las máquinas). Haremos el proceso de configuración en m2-imm98 y sería idéntico en las máquinas m1-imm98 y m3-imm98.

Vamos a modificar el archivo `/etc/crontab`. Para ello ejecutamos el comando:

```
sudo nano /etc/crontab
```

```
imm98@m2-imm98:/etc$ sudo nano crontab
```

E incluimos la siguiente regla:

```
@reboot root sh /home/imm98/cort_m3.sh
```

```
# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
@reboot root sh /home/imm98/cort_m3.sh
```

Esto quiere decir que cada vez que se reinicie la máquina m2-imm98 el demonio cron ejecutará el script `/home/imm98/cort_m3.sh` que es el script descrito en el último punto de la sección anterior, es decir el que hace que m2-imm98 sólo acepte tráfico de m3-imm98.

Vamos a comprobar cron ejecuta correctamente este script cuando se reinicia la máquina m2-imm98. Para ello reiniciamos la máquina m2-imm98.

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

System information as of Tue May  4 16:45:20 UTC 2021

System load:  0.97      Processes:            94
Usage of /:   49.4% of 8.79GB
Memory usage: 30%      IP address for enp0s3: 10.0.2.15
Swap usage:   0%        IP address for enp0s8: 192.168.56.104

* Pure upstream Kubernetes 1.21, smallest, simplest cluster ops!

https://microk8s.io/

64 packages can be updated.
0 updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

imm98@m2-imm98:~$ iptables -L -n -v
iptables v1.6.1: can't initialize iptables table `filter': Permission denied (you must be root)
Perhaps iptables or your kernel needs to be upgraded.
imm98@m2-imm98:~$ sudo iptables -L -n -v
[sudo] password for imm98:
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *        192.168.56.107        0.0.0.0/0

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

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *        192.168.56.107        0.0.0.0/0
imm98@m2-imm98:~$
```

Y vemos que si tras realizar el reinicio de la máquina m2-imm98 ejecutamos el comando:

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

se nos muestran las reglas del cortafuegos que se generan tras la ejecución del script cort\_m3.sh por lo que el demonio cron ha realizado su trabajo correctamente.

## 7. CREAR, INSTALAR Y CONFIGURAR CERTIFICADO SSL EN APACHE Y NGINX CON CERTBOT

Para la instalación de Certbot primeramente debemos desactivar todas las reglas del cortafuegos para que acepte todo el tráfico entrante. Vamos a probar que se ha ejecutado correctamente con el comando:

*iptables -L -n -v*

```
imm98@m1-imm98:~$ sudo iptables -L -n -v
[sudo] password for imm98:
Chain INPUT (policy ACCEPT 6208 packets, 444K 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 3512 packets, 351K bytes)
 pkts bytes target     prot opt in     out     source                   destination
```

Ya con las reglas correctas del cortafuegos vamos a actualizar el sistema con el comando:

*sudo apt update*

```
imm98@m1-imm98:~$ sudo apt update
[sudo] password for imm98:
0% [Connecting to es.archive.ubuntu.com]
```

Y una vez que esté actualizado el sistema vamos a proceder a instalar certbot con el comando:

*sudo apt install -y certbot*

```
imm98@m1-imm98:~$ sudo apt install -y certbot
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-minimal python-pyicu
  python2.7 python2.7-minimal python3-acme python3-certbot python3-configargparse python3-future
  python3-josepy python3-lib2to3 python3-mock python3-parsedatetime python3-pbr
  python3-requests-toolbelt python3-rfc3339 python3-tz python3-zope.component python3-zope.event
  python3-zope.hookable
Suggested packages:
  python3-certbot-apache python3-certbot-nginx python-certbot-doc python-doc python-tk
  python2.7-doc binutils binfmt-support python-acme-doc python-future-doc python-mock-doc
The following NEW packages will be installed:
  certbot libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-minimal
  python-pyicu python2.7 python2.7-minimal python3-acme python3-certbot python3-configargparse
  python3-future python3-josepy python3-lib2to3 python3-mock python3-parsedatetime python3-pbr
  python3-requests-toolbelt python3-rfc3339 python3-tz python3-zope.component python3-zope.event
  python3-zope.hookable
0 upgraded, 24 newly installed, 0 to remove and 62 not upgraded.
Need to get 5126 kB of archives.
After this operation, 23.6 MB of additional disk space will be used.
0% [Connecting to es.archive.ubuntu.com]
```

Una vez que tenemos instalado Certbot, podemos mostrar una lista de los certificados administrados por Certbot con la orden:

*sudo certbot certificates*

```
imm98@m1-imm98:~$ sudo certbot certificates
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-----
No certs found.
-----
```

Como todavía no tenemos ningún certificado administrado por Certbot nos aparece el mensaje "No certs found"

Vamos a proceder a obtener un certificado otorgando acceso certbot al directorio raíz de un servidor web. Para ello utilizaremos el comando:

*sudo certbot certonly --webroot*

```
imm98@m1-imm98:~$ sudo certbot certonly --webroot
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator webroot, Installer None
Enter email address (used for urgent renewal and security notices) (Enter 'c' to
cancel): imm98@correo.ugr.es

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
-----
(A)gree/(C)ancel: A

-----
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
-----
(Y)es/(N)o: N
Please enter in your domain name(s) (comma and/or space separated) (Enter 'c'
to cancel): www.example_imm98.com_
```

Para crear el certificado nos pedirá que aceptemos los términos de servicio y que introduzcamos nuestro nombre de dominio. Como no tengo nombre de dominio, he escrito un supuesto dominio llamado [www.example\\_imm98.com](http://www.example_imm98.com)

```
An unexpected error occurred:
Error creating new order :: Cannot issue for "www.example_imm98.com": Domain name contains an invalid character
```

Como el nombre de dominio no existe, nos aparece este mensaje de error.



### a. APACHE CON CERTBOT

En esta sección vamos a ver cómo proteger Apache con Certbot. Vamos a utilizar un certificado de SSL para Apache en Ubuntu.

Lo primero que vamos a hacer es instalar el paquete Apache que tiene Certbot. Para ello ejecutaremos el comando:

```
sudo apt install -y python-certbot-apache
```

```
imm98@m1-imm98:~$ sudo apt install -y python-certbot-apache_
```

Y ahora procedemos a ejecutarlo con el comando:

```
sudo certbot run --apache
```

```
imm98@m1-imm98:~$ sudo certbot run --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
No names were found in your configuration files. Please enter in your domain
name(s) (comma and/or space separated) (Enter 'c' to cancel):
```

Tras ejecutar el comando de arriba vemos que nos solicita de nuevo el nombre de dominio que en nuestro caso no tenemos.

Certbot debe encontrar el host virtual adecuado en su configuración de Apache, concretamente, buscando una directiva llamada *ServerName* que coincida con el dominio para el que necesitamos el certificado. Deberíamos tener un archivo llamado *example\_imm98.com.conf* pero al no tener dominio no lo tenemos. Este archivo debe estar ubicado en el directorio */etc/apache2/sites-available/* y debería de contener lo que aparece a continuación:

```
imm98@m1-imm98:~$ sudo nano /etc/apache2/sites-available/example_imm98.com.conf_
```

```
ServerName example_imm98
```

```
ServerName example_imm98
```

Una vez que esté todo correcto, debemos reiniciar el servicio de apache con el comando:

```
sudo systemctl reload apache2
```

```
imm98@m1-imm98:~$ sudo systemctl reload apache2
```

Para obtener un certificado SSL para el dominio `example_imm98` debemos de escribir el siguiente comando:

```
sudo certbot --apache -d example_imm98 -d www.example\_imm98
```

```
imm98@m1-imm98:~$ sudo certbot --apache -d example_imm98 -d www.example_imm98_
```

siendo [www.example\\_imm98](http://www.example_imm98) y `example_imm98` los nombres para los cuales el certificado tiene validez.

## **b. NGINX CON CERTBOT**

Ahora vamos a utilizar Certbot para conseguir un certificado SSL gratuito para Nginx. Primero vamos a instalar el software de Certbot y su complemento de Nginx el comando:

```
sudo apt install certbot python3-certbot-nginx
```

```
imm98@m1-imm98:~$ sudo apt install certbot python3-certbot-nginx
```

Certbot debe encontrar el host virtual adecuado en su configuración de Nginx, concretamente, buscando una directiva llamada *ServerName* que coincida con el dominio para el que necesitamos el certificado. Deberíamos de tener un archivo llamado `example_imm98.com` en el directorio **/etc/nginx/sites-available/** . Dentro de este archivo deberíamos de encontrar la línea:

```
server_name example_imm98.com www.example_imm98.com
```

Para ver el contenido de este archivo escribimos:

```
sudo nano /etc/nginx/sites-available/example_imm98.com
```

```
imm98@m1-imm98:~$ sudo nano /etc/nginx/sites-available/example_imm98.com
```

y deberíamos de tener la siguiente línea:

```
server_name example_imm98.com www.example_imm98.com;
```

Una vez que hemos modificado la configuración de nginx tenemos que reiniciar el servicio. Esto se hace con la orden:

```
sudo systemctl reload nginx.service
```

```
imm98@m1-imm98:~$ sudo systemctl reload nginx.service
```

Por último, para obtener un certificado SSL para el dominio `example_imm98` debemos de escribir el siguiente comando:

```
sudo certbot --nginx -d example_imm98.com -d www.example_imm98.com
```

```
imm98@m1-imm98:~$ sudo certbot --nginx -d example_imm98.com -d www.example_imm98.com
```

donde se ejecuta `certbot` con el complemento `--nginx` y se usa `-d` para indicar los dominios para los que este certificado va a ser válido, concretamente para `example.com` y `www.example.com` .