

SERVIDORES WEB DE ALTAS PRESTACIONES

Práctica 4: Asegurar la granja Web



ugr

Universidad
de **Granada**

Autor: Sergio Aguilera Ramírez

Curso 2019 - 2020

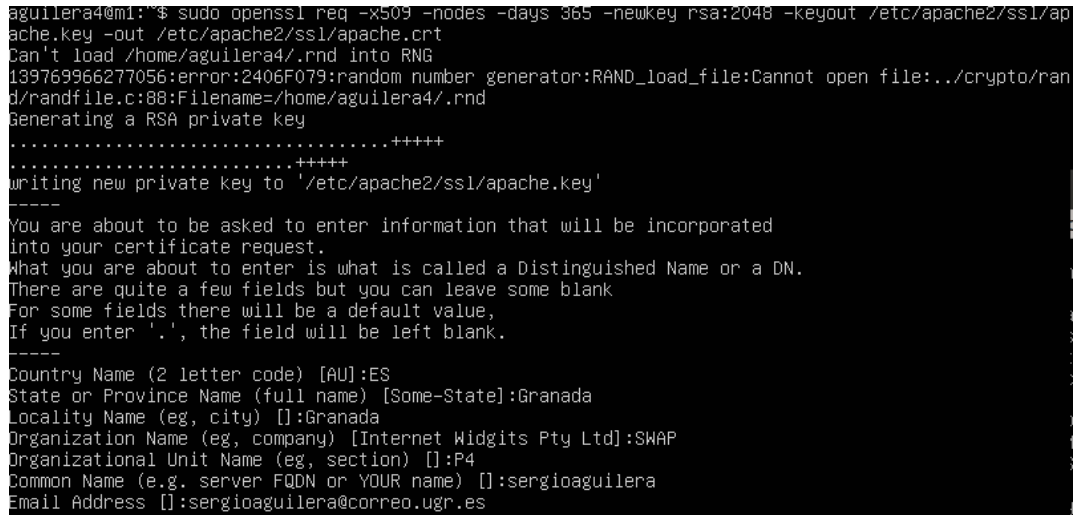
Índice

1. Creación e instalación de certificado SSL autofirmado	2
2. Configuración cortafuegos	4
3. Ejercicio opcional 1: Configurar m3 para peticiones HTTP y HTTPS	5
4. Ejercicio opcional 2: Configurar cortafuegos (peticiones únicamente a m3)	8
5. Bibliografía	11

1. Creación e instalación de certificado SSL autofirmado

Para esta primera parte de la práctica, vamos crear e instalar un certificado SSL autofirmado en la m1. Para ello, en la propia máquina ejecutamos las siguientes órdenes de forma secuencial:

1. `sudo a2enmod ssl`

A terminal window showing the execution of the 'a2enmod ssl' command. It starts with a prompt 'aguilera4@m1:~\$' followed by the command. The output shows the generation of a new RSA private key and a self-signed certificate. It prompts for various fields like Country Name, State or Province Name, Locality Name, Organization Name, Organizational Unit Name, Common Name, and Email Address. The user provides values for Country (ES), State (Granada), Locality (Granada), Organization (SWAP), Organizational Unit (P4), Common Name (sergioaguilera), and Email Address (sergioaguilera@correo.ugr.es).

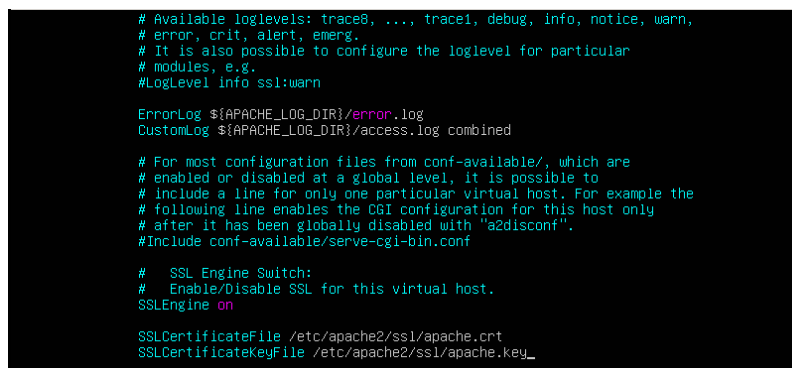
```
aguilera4@m1:~$ sudo a2enmod ssl
Can't load /home/aguilera4/.rnd into RNG
139769966277056:error:2406F079:random number generator:RAND_load_file:Cannot open file:../crypto/rand/randfile.c:88:Filename=/home/aguilera4/.rnd
Generating a RSA private key
.....+++++
.....+++++
Writing new private key to '/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) []:sergioaguilera
Email Address []:sergioaguilera@correo.ugr.es
```

Figura 1: Comando a2enmod

2. `sudo service apache2 restart`
3. `sudo mkdir /etc/apache2/ssl`
4. `sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache2.crt`

Tras ejecutar estas órdenes, nos pedirá información "personal" para rellenar el certificado. Una vez rellenado estos datos, se crearán dos archivos en la carpeta ssl, referentes al certificado. Ahora pasamos a configurar el fichero default-ssl.conf añadiendo las líneas que especifican la ruta hasta los archivos del certificado (*Figura 2*):

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

A terminal window showing the contents of the default-ssl.conf file. It includes comments about log levels and modules, and configuration lines for ErrorLog, CustomLog, and SSL engine switch. The SSLCertificateFile and SSLCertificateKeyFile lines are highlighted in green.

```
# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

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

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

#
# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
#
SSLEngine on

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

Figura 2: Configuración default-ssl.conf m1.

Posteriormente activamos el sitio default-ssl mediante el comando `sudo a2ensite default-ssl` y reiniciamos el servicio apache. Accedemos al servidor web a través del navegador (Chrome) mediante el protocolo HTTPS. Como vemos en la *Figura 3* el https sale en rojo ya que hemos creado un certificado autofirmado (*Figura 4*).

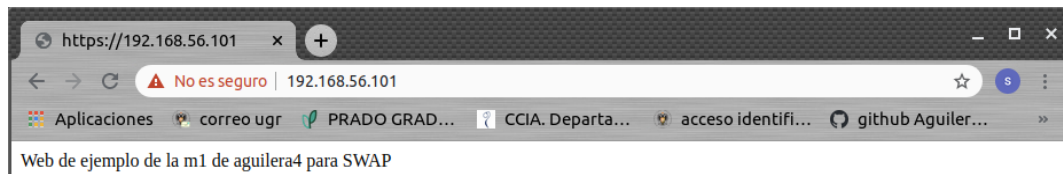
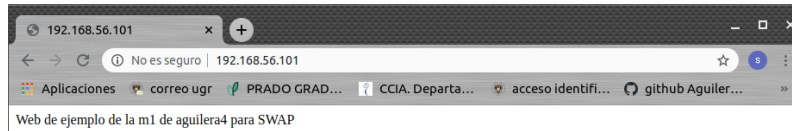


Figura 3: Barra navegador



Figura 4: Certificado autofirmado

Además, comprobamos que el servicio HTTP funciona correctamente:



2. Configuración cortafuegos

En esta sección, vamos a asegurar nuestro servidor web final m1 evitando accesos no deseados, mediante la configuración del cortafuegos. Para ello, se utiliza el comando *iptables*, que permite bloquear puertos, direcciones IP, etc. mediante el establecimiento de un conjunto de reglas.

Para llevar a cabo esto, se a creado un script llamado *script_cortafuegos1.sh* (Figura 5), que permite asegurar nuestro servidor web, dejando realizar peticiones por los puertos 80 (HTTP) y 443 (HTTPS).

```
#!/bin/bash
# Eliminamos todos las reglas anteriores
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# Denegamos 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

# Permitimos que el localhost tenga acceso
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Permitimos el tráfico por el puerto 22 conexiones ssh
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# Permitimos el tráfico por el puerto 80 para peticiones HTTP
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

# Permitimos el tráfico por el puerto 443 para peticiones HTTPS
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT
```

Figura 5: Configuración iptables M1

Una vez creado el script para la configuración del cortafuegos, crearemos un demonio (servicio) que ejecute dicho script al inicio del arranque de la máquina. En la ruta */etc/systemd/system/* creamos un archivo llamado *activar_cortafuegos.service*. Este archivo será el servicio que se lanzará al iniciar la máquina, donde su configuración se puede ver en la Figura 6. En la variables 'ExecStart' introducimos la ruta donde se encuentra el script bash que queremos ejecutar, en nuestro caso se encuentra en */home/aguillera4/*. Ambos ficheros (*activar_cortafuegos.service* y *script_cortafuegos1.sh*) deben tener permisos de ejecución, si no los tienen se los damos a través del comando *sudo chmod +x fichero*.

```
[Unit]
Description=cortafuegos m1
After=syslog.target

[Service]
Type=forking
ExecStart=/home/aguilera4/script_cortafuegos1.sh

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

Figura 6: Servicio m1

Tras completar la configuración de nuestro servicio, debemos recargar 'systemd' para que tenga en cuenta este nuevo servicio. Para ello, ejecutamos las siguientes órdenes secuencialmente:

1. **`sudo systemctl daemon-reload`**: refrescamos systemd.
2. **`sudo systemctl start activar_cortafuegos.service`**: arrancamos el servicio.
3. **`sudo systemctl enable activar_cortafuegos.service`**: habilitamos el servicio para que se autoarranque.

Para comprobar que el servicio ha sido configurado correctamente, reiniciamos la máquina y comprobamos el estado del cortafuegos con el comando `iptables -L -n -v`. Observando la *Figura 7* vemos como los puertos 80 y 443 están listos para escuchar.

```
aguilera4@m1:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 1 packets, 40 bytes)
 pkts bytes target    prot opt in     out     source                 destination             state NEW,ESTABLISHED
 29 5632 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 25 packets, 2448 bytes)
 pkts bytes target    prot opt in     out     source                 destination
 8    760 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
```

Figura 7: Estado cortafuegos m1

3. Ejercicio opcional 1: Configurar m3 para peticiones HTTP y HTTPS

Para este primer ejercicio opcional, se va a transferir el certificado creado en m1 al resto de las máquinas (m2 y balanceador) y configuraremos el balanceador para que este acepte y balancee de forma correcta el tráfico HTTP y HTTPS.

Para ello, utilizaremos el comando `scp` para copiar los archivos del certificado m1 a las máquinas m2 y balanceador (*Figura 8 y 9*).

```

aguilera4@m1:~$ sudo scp /etc/apache2/ssl/apache.crt aguilera4@192.168.56.110:/home/aguile
.crt
The authenticity of host '192.168.56.110 (192.168.56.110)' can't be established.
ECDSA key fingerprint is SHA256:u58eSA971r8xtt00hKtRnFXeT3oBUp8iEIuZtR1z2Ls.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.110' (ECDSA) to the list of known hosts.
aguilera4@192.168.56.110's password:
apache.crt                                100% 1460      2.6MB/s
aguilera4@m1:~$ sudo scp /etc/apache2/ssl/apache.key aguilera4@192.168.56.110:/home/aguile
.key
aguilera4@192.168.56.110's password:
apache.key                                100% 1708      3.0MB/s

```

Figura 8: Estado cortafuegos m1

```

aguilera4@m1:~$ sudo scp /etc/apache2/ssl/apache.crt aguilera4@192.168.56.102:/home/aguile
.crt
aguilera4@192.168.56.102's password:
apache.crt                                100% 1460      2.0MB/s
aguilera4@m1:~$ sudo scp /etc/apache2/ssl/apache.key aguilera4@192.168.56.102:/home/aguile
.key
aguilera4@192.168.56.102's password:
apache.key                                100% 1708      2.7MB/s

```

Figura 9: Estado cortafuegos m1

Respecto a la máquina 'm2', los pasos para la configuración del certificado son similares a los de m1. En primer lugar creamos una carpeta llamada 'ssl' en la ruta `/etc/apache2/`. Seguidamente, movemos los archivos del certificado (`apache.crt` y `apache.key`) a esta carpeta. Una vez realizado los pasos anteriores, ejecutamos el comando `'sudo a2enmod ssl & sudo service apache2 restart'`. De igual forma que para 'm1', añadimos las líneas `SSLCertificateFile /etc/apache2/ssl/apache.crt` y `SSLCertificateKeyFile /etc/apache2/ssl/apache.key` en el archivo `default-ssl.conf` (Figura 10). Una vez que tenemos estos dos archivos en la carpeta ssl, activamos la carpeta ssl para apache y reiniciamos el servicio.

```

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

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

```

Figura 10: Configuración default-ssl.conf

Una vez realizado los pasos anteriores, activamos default.ssl mediante el comando 'a2ensite' y recargamos el servicio apache2. Por consiguiente, nuestro certificado ya estaría configurado para m2 (Figura 11).

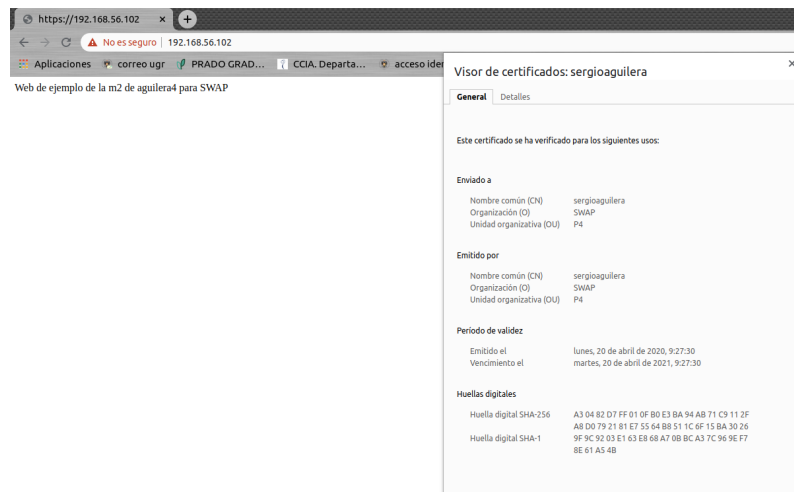


Figura 11: Comprobación certificado m2 (Navegador)

Por otro lado, configuramos m3 para peticiones HTTP y HTTPS. Como hemos comentado antes, los archivos del certificado se han transferido de m1 a m3, y guardado en la carpeta ssl en la ruta `/home/aguilera4/ssl/`. Solo quedaría crear un nuevo servidor en el archivo de configuración de nginx, para peticiones HTTPS. El nuevo servidor se llamará `balanceador1`, cuya configuración se puede ver en la *Figura 12*.

```
server{
    listen 443 ssl;
    ssl on;
    ssl_certificate      /home/aguilera4/ssl/apache.crt;
    ssl_certificate_key  /home/aguilera4/ssl/apache.key;

    server_name balanceador1;

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

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

Figura 12: Configuración nuevo servidor Nginx

Una vez realizado los pasos anteriores, reiniciamos el servicio nginx y comprobamos su correcto funcionamiento enviando peticiones HTTP y HTTPS (*Figura 13*).


```

sergio@sergio-GS63VR-7RF:~$ curl http://192.168.56.110/
<HTML>
  <BODY>
    Web de ejemplo de la m2 de aguilara4 para SWAP
  </BODY>
</HTML>

sergio@sergio-GS63VR-7RF:~$ curl http://192.168.56.110/
<HTML>
  <BODY>
    Web de ejemplo de la m1 de aguilara4 para SWAP
  </BODY>
</HTML>

sergio@sergio-GS63VR-7RF:~$ curl -k https://192.168.56.110/
<HTML>
  <BODY>
    Web de ejemplo de la m2 de aguilara4 para SWAP
  </BODY>
</HTML>

sergio@sergio-GS63VR-7RF:~$ curl -k https://192.168.56.110/
<HTML>
  <BODY>
    Web de ejemplo de la m1 de aguilara4 para SWAP
  </BODY>
</HTML>

```

Figura 13: Comprobación balanceador Nginx (HTTP y HTTPS)

4. Ejercicio opcional 2: Configurar cortafuegos (peticiones únicamente a m3)

Por último, como tarea opcional 2, vamos configurar nuestra granja web para que los servidores web finales (m1 y m2) únicamente puedan recibir peticiones tanto HTTP como HTTPS del balanceador (m3), es decir, si realizamos una petición HTTP o HTTPS desde m4 (en mi caso la máquina anfitriona) a m1 o m2 estos la rechazarán, en cambio si enviamos esta petición a m3 será aceptada.

Para ello, configuramos el cortafuegos en la m3 para que únicamente acepte peticiones HTTP y HTTPS, al igual que se a explicado anteriormente, creamos un demonio que se ejecute al arrancar dicha máquina *Figura 14*, donde el script a ejecutar es el mostrado en la *Figura 15*.

```

[Unit]
Description=cortafuegos balanceador
After=syslog.target

[Service]
Type=forking
ExecStart=/home/aguilara4/script_cortafuegos.sh

[Install]
WantedBy=multi-user.target

```

Figura 14: Servicio cortafuegos del balanceador

```
#!/bin/bash
# Eliminamos las reglas anteriores
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# Denegamos 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

# Permitimos el tráfico por el puerto 80 para peticiones HTTP
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

# Permitimos el tráfico por el puerto 443 para peticiones HTTPS
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT
```

Figura 15: Script cortafuegos balanceador

Comprobamos si el estado del cortafuegos del balanceador es correcto.

```
aguilera4@balanceador:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination    state NEW,E
STABLISHED
    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 49 packets, 3802 bytes)
 pkts bytes target    prot opt in     out     source    destination    tcp spt:80
    12  1711 ACCEPT    tcp  --  *      *       0.0.0.0/0  0.0.0.0/0      tcp spt:443
    20   6805 ACCEPT    tcp  --  *      *       0.0.0.0/0  0.0.0.0/0      tcp spt:443
```

Figura 16: Estado cortafuegos balanceador

Ahora solo quedaría establecer las reglas en los servidores web finales donde se rechacen todas las peticiones que no provengan de la máquina balanceadora. En el caso de m1 se ha creado un nuevo script para la configuración del cortafuegos (*Figura 17*) y se ha modificado el script especificado en el servicio *activar_cortafuegos.service* para que ahora ejecute el script final (*Figura 18*).

```
[Unit]
Description=cortafuegos m1
After=syslog.target

[Service]
Type=forking
ExecStart=/home/aguilera4/script_cortafuegos.sh

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

Figura 17: Servicio cortafuegos m1

```
#!/bin/bash
# Eliminamos todas las reglas existentes
iptables -F
iptables -X

# Denegamos todo el tráfico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Permitimos acceso a la máquina local
#iptables -A INPUT -i lo -j ACCEPT
#iptables -A OUTPUT -o lo -j ACCEPT

# Permitimos salida a conexiones nuevas, establecidas y relacionadas, además permitimos la entrada solo a conexiones establecidas y relacionadas
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Permitimos el acceso a la M3
iptables -I INPUT -s 192.168.56.110 -j ACCEPT
iptables -I OUTPUT -s 192.168.56.110 -j ACCEPT
```

Figura 18: Script cortafuegos m1

De igual forma se ha llevado a cabo la configuración de m2, se ha creado un demonio igual que para m1 (*Figura 19*) y el script de configuración del cortafuegos con las mismas reglas que para m1 (*Figura 18*).

```
[Unit]
Description=cortafuegos m2
After=syslog.target

[Service]
Type=forking
ExecStart=/home/aguilera4/script_cortafuegos.sh

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

Figura 19: Servicio cortafuegos m2

Comprobamos el estado del cortafuegos para ambos servidores.

```
aguilera4@m1:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 71 packets, 5178 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       192.168.56.110       0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0             0.0.0.0/0             state RELATED,ESTABLISHED

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     all  --  *      *       192.168.56.110       0.0.0.0/0
   66 4878 ACCEPT     all  --  *      *       0.0.0.0/0             0.0.0.0/0             state NEW,RELATED,ESTABLISHED
```

Figura 20: Reglas M1 cortafuegos Final

```

aguilera4@m2:~$ sudo iptables -L -n -v
sudo: unable to resolve host m2: Resource temporarily unavailable
Chain INPUT (policy DROP 100 packets, 6796 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT    all  --  *      *       192.168.56.110       0.0.0.0/0
    0     0 ACCEPT    all  --  *      *       0.0.0.0/0             0.0.0.0/0             state RELATED,ESTABLISHED

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    all  --  *      *       192.168.56.110       0.0.0.0/0
  100 6796 ACCEPT    all  --  *      *       0.0.0.0/0             0.0.0.0/0             state NEW,RELATED,ESTABLISHED

```

Figura 21: Reglas M2 cortafuegos Final

5. Bibliografía

<http://ipset.netfilter.org/iptables.man.html>

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