3º Grado en Ingeniería Informática (Tecnologías de la Información) - Servidores Web de Altas Prestaciones

PRÁCTICA 4

Asegurar la granja web

Autor: Adrián Acosa Sánchez

Índice

Instalar un certificado SSL autofirmado para configurar el acceso por HTTPS

Generación del certificado

Para poder configurar el acceso a nuestra granja web mediante el protocolo HTTPS, debemos obtener un certificado SSL. Éste se puede conseguir de diversas maneras. La que vamos a ver ahora es generándonos uno de manera autofirmada. Para ello solo tenemos que activar el módulo SSL que proporciona Apache2. Luego, tendremos que ejecutar los siguientes comandos en nuestra máquina:

```
adrianacosa@m1-adrianacosa:~$ sudo a2enmod ssl
[sudo] password for adrianacosa:
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
adrianacosa@m1-adrianacosa:~$ sudo service apache2 restart
adrianacosa@m1-adrianacosa:~$ sudo mkdir /etc/apache2/ssl
```

Cuando ejecutemos el siguiente comando de openssl, nos pedirá la información que contendrá el certificado tales como nombre de país, provincia, localidad, etc. :

```
adrianacosa@m1-adrianacosa:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
 /etc/apache2/ssl/apache_adrianacosa.key -out /etc/apache2/ssl/apache_adrianacosa.crt
Generating a RSA private key
.........................................+++++
...................+++++
writing new private key to '/etc/apache2/ssl/apache_adrianacosa.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) []:adrianacosa
Email Address []:adrianacosa@correo.ugr.es
adrianacosa@m1-adrianacosa:~$
```

Configuración de Apache para habilitar el módulo SSL

Una vez realizado este paso, ya tendremos nuestro certificado autofirmado. Para poder habilitarlo de manera correcta en nuestra granja, debemos cambiar la configuración por defecto del SSL que proporciona apache para usar nuestro certificado recién creado (editando el archivo /etc/apache2/sites-available/default-ssl.conf añadiendo las siguientes líneas):

```
#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache_adrianacosa.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache_adrianacosa.key
```

Cuando hagamos esto, solo queda activar el sitio default-ssl y reiniciar apache:

```
adrianacosa@m1-adrianacosa:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
adrianacosa@m1-adrianacosa:~$ sudo service apache2 relodd
Usage: apache2 {start|stop|graceful-stop|restart|reload|force-reload}
adrianacosa@m1-adrianacosa:~$ sudo service apache2 reload
```

Y comprobando el certificado podemos ver que es el que acabamos de generar nosotros:


adrianacosa
Subject Name
Country ES
State/Province Granada
Locality Granada
Organization SWAP
Organizational Unit P4
Common Name adrianacosa
Email Address adrianacosa@correo.ugr.es
Issuer Name
Country ES
State/Province Granada
Locality Granada
Organization SWAP
Organizational Unit P4
Common Name adrianacosa
Email Address adrianacosa@correo.ugr.es
Validity
Not Before Sun, 08 May 2022 18:42:19 GMT
Not After Mon, 08 May 2023 18:42:19 GMT
Public Key Info
Algorithm RSA
Key Size 2048
Exponent 65537
Modulus DC:95:E1:F5:D0:13:C9:6F:CB:F4:8D:95:6F:0F:67:2E:86:AF:BE:28:49:65:08:EC:...

Configuración del resto de máquinas de la granja

Un aspecto importante es que lo que acabamos de hacer es habilitar el módulo HTTPS de apache en nuestra máquina 1, pero queremos que en nuestra granja web el balanceador de carga pueda aceptar también éste tipo de tráfico a través de HTTPS por lo que tendremos que pasarle éste certificado tanto a la máquina 2 como a nuestro balanceador. Para ello vamos a usar la herramienta scp de la siguiente manera:

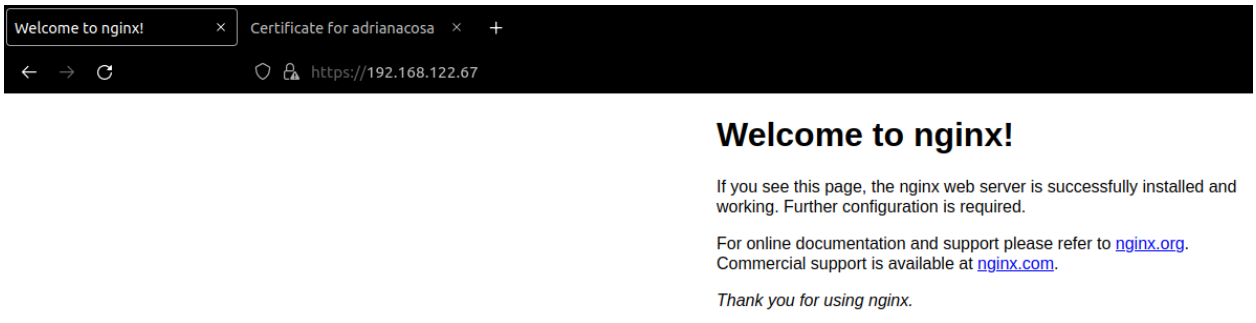
```
adrianacosa@m1-adrianacosa:~$ sudo scp /etc/apache2/ssl/apache_adrianacosa.crt adrianacosa@192
.168.122.36:/home/adrianacosa/apache_adrianacosa.crt
The authenticity of host '192.168.122.36 (192.168.122.36)' can't be established.
ECDSA key fingerprint is SHA256:DgxtgGhPFgAfSJ8DY2KOB9mLsRZgIl60pAOF180oq5o.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.122.36' (ECDSA) to the list of known hosts.
adrianacosa@192.168.122.36's password:
apache_adrianacosa.crt                                    100% 1444     1.7MB/s   00:00
adrianacosa@m1-adrianacosa:~$ sudo scp /etc/apache2/ssl/apache_adrianacosa.crt adrianacosa@192
.168.122.67:/home/adrianacosa/apache_adrianacosa.crt
The authenticity of host '192.168.122.67 (192.168.122.67)' can't be established.
ECDSA key fingerprint is SHA256:JmJrmLLIth5Hgoe4qff4+MQgC6gRDxN2J2YwdllUbAo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.122.67' (ECDSA) to the list of known hosts.
adrianacosa@192.168.122.67's password:
apache_adrianacosa.crt                                    100% 1444     1.6MB/s   00:00
adrianacosa@m1-adrianacosa:~$ sudo scp /etc/apache2/ssl/apache_adrianacosa.key adrianacosa@192
.168.122.36:/home/adrianacosa/apache_adrianacosa.key
adrianacosa@192.168.122.36's password:
apache_adrianacosa.key                                    100% 1704     1.1MB/s   00:00
adrianacosa@m1-adrianacosa:~$ sudo scp /etc/apache2/ssl/apache_adrianacosa.key adrianacosa@192
.168.122.67:/home/adrianacosa/apache_adrianacosa.key
adrianacosa@192.168.122.67's password:
apache_adrianacosa.key                                    100% 1704     1.6MB/s   00:00
adrianacosa@m1-adrianacosa:~$
```

Y una vez tenemos los archivos en ambas máquinas, lo primero es configurar la máquina 2 exactamente igual que lo hemos hecho con la máquina 1, es decir, crear la carpeta ssl, mover los archivos que hemos copiado mediante scp, activar el módulo SSL de apache, configurar el archivo default-ssl.conf añadiendo las mismas líneas que en la máquina 1 y volver a reiniciar el servicio de apache para poder hacer uso de HTTPS.

Para el caso del balanceador tendremos que poner la ruta donde hayamos copiado tanto el certificado como la clave del certificado. Después hay que añadir un nuevo server al archivo de configuración de nginx:

```
adrianacosa@m3-adrianacosa:~$ mkdir ssl
adrianacosa@m3-adrianacosa:~$ mv apache_adrianacosa* ssl
```

```
adrianacosa@m3-adrianacosa:~$ cat /etc/nginx/conf.d/default.conf
upstream balanceo_adrianacosa{
        server 192.168.122.43 weight=2;
        server 192.168.122.36 weight=1;
}

server{
        listen 80;
        server_name balanceador_adrianacosa;

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

        location /
        {
                proxy_pass http://balanceo_adrianacosa;
                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/adrianacosa/ssl/apache_adrianacosa.crt;
        ssl_certificate_key     /home/adrianacosa/ssl/apache_adrianacosa.key;
}
adrianacosa@m3-adrianacosa:~$
```



Y como podemos comprobar, funciona todo correctamente devolviendo el mismo certificado que antes.

Configuración del cortafuegos

En este apartado nos vamos a dedicar a configurar el cortafuegos de nuestra granja web. Este se encarga de filtrar los paquetes TCP/IP que entran y salen en nuestra granja web, permitiendo o denegando estos según ciertos criterios. La forma que tenemos de configurar el cortafuegos en linux es usando la herramienta *iptables*. El superusuario puede definir reglas con ésta herramienta de filtrado de paquetes o NAT entre otras funcionalidades, manteniendo los datos captados en el log del sistema.

Para configurar de manera correcta *iptables* conviene que por defecto establezcamos reglas para denegar cualquier tipo de servicio, y luego aceptar el tráfico que deseemos que sí pueda entrar o salir de nuestra granja web. Por último estableceremos los rangos de direcciones IP a los que aplicar diversas reglas y mantendremos los logs del tráfico que ha sido denegado e intentos de acceso para poder comprobar una vez puesto en marcha el cortafuegos la posibilidad de haber sufrido algún ataque a nuestra granja.

Denegación de todo el tráfico entrante a la granja excepto HTTP y HTTPS

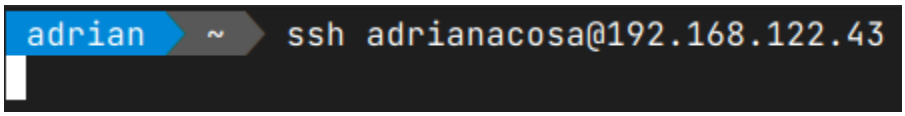
Para poder llevar a cabo ésta configuración, vamos a hacer uso de un script de linux donde realizaremos todos los pasos indicados en el párrafo anterior. El script es el siguiente:

```
#!/bin/bash
# (1) Eliminamos todas las reglas que hubiera para
# realizar una configuracion limpia
iptables -F
iptables -X

# (2) establecemos las politicas por defecto, es decir, la denegacion
# de cualquier tipo de trafico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

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

# (4) Permitimos la salida del equipo con conexiones nuevas que
# solicitemos, conexiones establecidas y relacionadas. Permitir la entrada
# solo de 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
```

Hasta aquí tendríamos una configuración de un script básica denegando cualquier tipo de tráfico entre la granja y el exterior. Ahora para poder terminar con el script que hemos comentado, lo que haremos será permitir el tráfico en los puertos 80 y 443 con las siguientes órdenes:

```
# (5) Permitimos tanto el trafico HTTP/HTTPS abriendo los
# los puertos 80 y 443
iptables -A INPUT -i enp1s0 -p tcp -m multiport --dports 80,443 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o enp1s0 -p tcp -m multiport --sports 80,443 -m state --state ESTABLISHED -j ACCEPT
```

De ésta manera podemos comprobar que a partir la ejecución del script completo deja de funcionar las nuevas conexiones que se realicen utilizando SSH:



Como se puede ver ya no me pide la contraseña ni nada, se queda ahí esperando.

Para poder probar más cosas, he creado un script que limpia todas las reglas del cortafuegos, que consiste en lo siguiente:

```
#!/bin/bash
# (1) Eliminar todas las reglas (configuracion limpia)
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# politica por defecto: aceptar todo
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -L -n -v
```

Configuración del resto de máquinas

Teniendo el script configurado completamente, voy a realizar una copia con scp a cada una de las máquinas desde la máquina 1 y voy a ejecutar el script en cada uno:

```
adrianacosa@m1-adrianacosa:~$ sudo scp accept-http-https-ssh.sh adrianacosa@192.168.122.36:/home/adrianacosa
adrianacosa@192.168.122.36's password:
accept-http-https-ssh.sh                      100% 1072   539.8KB/s   00:00
adrianacosa@m1-adrianacosa:~$ sudo scp accept-http-https-ssh.sh adrianacosa@192.168.122.67:/home/adrianacosa
adrianacosa@192.168.122.67's password:
accept-http-https-ssh.sh                      100% 1072     1.2MB/s   00:00
adrianacosa@m1-adrianacosa:~$
```

Y simplemente en cada una de las máquinas restantes ejecutaremos el script para tener la misma configuración que en el resto. Por ejemplo, en el balanceador ejecutamos:

```
adrianacosa@m3-adrianacosa:~$ sudo chmod +x accept-http-https-ssh.sh
adrianacosa@m3-adrianacosa:~$ sudo ./accept-http-https-ssh.sh
adrianacosa@m3-adrianacosa:~$
```

Script que permite SSH, PING y DNS exclusivamente

Esto se puede hacer muy fácilmente editando el script que acabo de realizar habilitando exclusivamente los puertos pertenecientes a SSH y DNS. Para aceptar también los ping tenemos que añadir dos reglas nuevas para aceptar los ping entrantes y salientes:

```
# (5) Permitimos tanto el trafico DNS y SSH especificando los puertos de
# dichos servicios
iptables -A INPUT -i enp1s0 -p tcp -m multiport --dports 22,53 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o enp1s0 -p tcp -m multiport --sports 22,53 -m state --state ESTABLISHED -j ACCEPT

# (6) Permitimos el trafico entrante y saliente del protocolo icmp
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Donde especificamos que el protocolo en vez de ser *tcp* será *icmp* y que si es de entrada a la granja será *echo-reply* y si es de salida será de tipo *echo-request*.

De la misma manera que hicimos con el script anterior, lo pasamos a cada una de las máquinas de la granja mediante *scp*:

```
adrianacosa@m1-adrianacosa:~$ sudo scp accept-ssh-ping-dns.sh adrianacosa@192.168.122.36:/home/adrianacosa
adrianacosa@192.168.122.36's password:
accept-ssh-ping-dns.sh                                                                      100% 1252   761.8KB/s   00:00
adrianacosa@m1-adrianacosa:~$ sudo scp accept-ssh-ping-dns.sh adrianacosa@192.168.122.67:/home/adrianacosa
adrianacosa@192.168.122.67's password:
accept-ssh-ping-dns.sh                                                                      100% 1252   278.8KB/s   00:00
adrianacosa@m1-adrianacosa:~$
```

Y lo ejecutamos en cada una de las máquinas, por ejemplo en la máquina 2:

```
adrianacosa@m2-adrianacosa:~$ chmod +x accept-ssh-ping-dns.sh
adrianacosa@m2-adrianacosa:~$ sudo ./accept-ssh-ping-dns.sh
adrianacosa@m2-adrianacosa:~$
```

Y podemos comprobar que no podemos realizar, por ejemplo, peticiones HTTP hacia cualquiera de las máquinas de la granja:

```
adrian ~ curl http://192.168.122.36
```

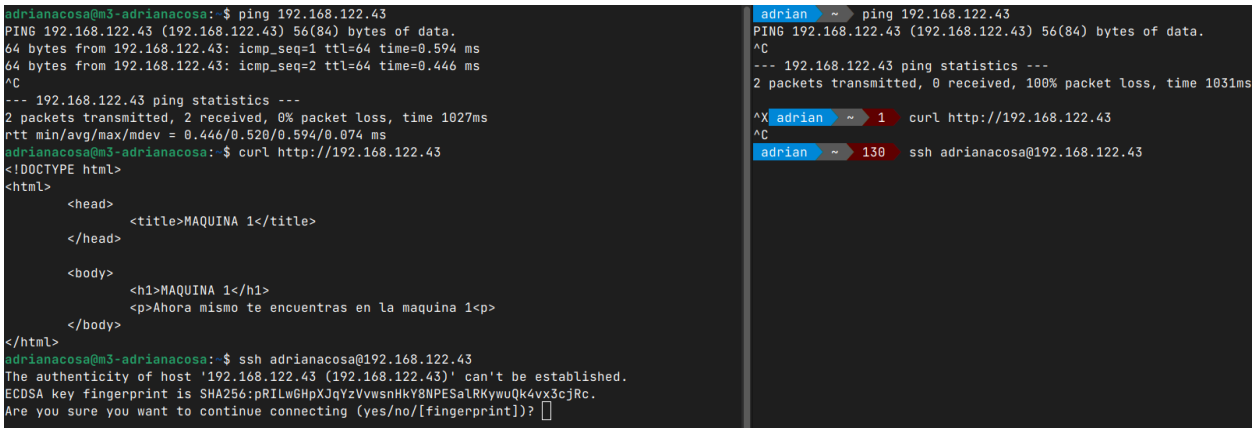
Como vemos, no obtenemos respuesta por parte del servidor debido a que el paquete no ha llegado a entrar.

Sólo la máquina 3 acepta peticiones HTTP/HTTPS

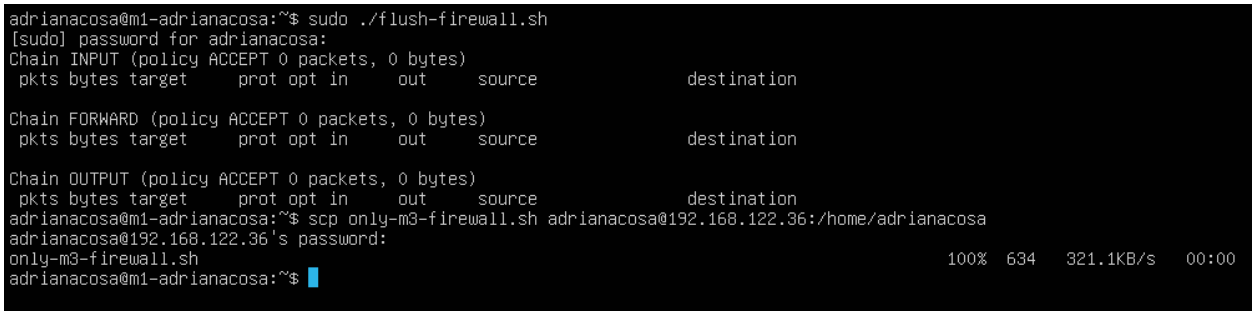
En éste apartado vamos a configurar que sólo la máquina 3 sea capaz de aceptar peticiones HTTP/HTTPS mientras que las máquinas 1 y 2 sólo aceptarán peticiones de la máquina 3 de cualquier tipo. Para ello lo primero que vamos a configurar son las máquinas 1 y 2, donde haremos un script en el que ambas máquinas denegarán todos los paquetes que no sean provenientes de la máquina 3. El script empieza como todos los demás, denegando cualquier tipo de paquete INPUT o OUTPUT.

```
#!/bin/bash
# (1) Eliminamos todas las reglas que hubiera para
# realizar una configuracion limpia
iptables -F
iptables -X

# (2) establecemos las politicas por defecto, es decir, la denegacion
# de cualquier tipo de trafico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

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

# (4) Permitir solo los paquetes de entrada y salida solamente
# si provienen de y salen hacia la maquina 3
iptables -A INPUT -s 192.168.122.67 -p tcp -m multiport --dports 80,443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -d 192.168.122.67 -p tcp -m multiport --sports 80,443 -m state --state ESTABLISHED_-j ACCEPT
```

Los 3 primeros puntos son los de la configuración básica anteriormente comentada. El cuarto punto es el importante, donde sólo se aceptan las peticiones provenientes y salientes a M3 (la IP que sale es la de M3). He usado la opción *--sports* para indicar desde qué puertos puede devolver la información. Y comprobamos que funciona intentando acceder de cualquier manera desde mi máquina *host* y desde M3 para comparar funcionamientos:



Y como podemos ver, sólo podemos acceder a la máquina 1 desde M3. Ahora sólo quedaría pasar por *scp* el script y de iptables a la máquina 2 y ejecutarlo para que sólo se pueda acceder desde la máquina 3 al igual que la máquina 1:



Y lo ejecutamos igual que el resto de scripts. Una vez hecho esto, ya tenemos el acceso a M1 y M2 únicamente por M3. Ahora vamos a hacer que a M3 sólo se pueda acceder mediante HTTP o HTTPS con el script que hicimos al inicio. De esa manera tendríamos nuestra granja con sólo acceso mediante HTTP y HTTPS a través de la máquina 3 y acceso desde M1 y M2 a M3 mediante HTTP o HTTPS exclusivamente

Iniciar cortafuegos al inicio del sistema

Para poder iniciar el cortafuegos cada vez que arranquemos nuestro servidor, existen varias opciones. Una de ellas, por ejemplo, es usar "*crontab -e*" pasándole el script que hemos configurado con permisos de ejecución. Pero yo he decidido configurar un demonio en *systemd* de manera que cuando esté habilitado e iniciado, en cada inicio se ejecute en cada máquina los scripts correspondientes hechos en el apartado anterior.

Para comenzar, crearemos un archivo descriptor del servicio que vamos a llevar a cabo. En este caso le voy a llamar *cortafuegos.service* dentro del directorio */etc/systemd/system*. Dentro de éste tendremos que escribir lo siguiente:

```
[Unit]
Description=Servicio de inicializacion del cortafuegos

[Service]
Type=simple
ExecStart=/bin/bash /home/adrianacosa/accept-http-https.sh

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

Y para poder iniciarlo tenemos que darle los permisos necesarios y activar el servicio como si fuese cualquier servicio de linux:

```
adrianacosa@m3-adrianacosa:~$ sudo chmod 644 /etc/systemd/system/cortafuegos.service
adrianacosa@m3-adrianacosa:~$ sudo systemctl enable cortafuegos.service
Created symlink /etc/systemd/system/multi-user.target.wants/cortafuegos.service → /etc/systemd/system/cortafuegos.service.
adrianacosa@m3-adrianacosa:~$
```

Y después de hacer reboot vemos cómo al mirar el estado sale que se ha ejecutado con código de status 0, o sea, que se ha ejecutado correctamente:

```
adrianacosa@m3-adrianacosa:~$ sudo systemctl status cortafuegos.service
• cortafuegos.service - Servicio de inicializacion del cortafuegos
     Loaded: loaded (/etc/systemd/system/cortafuegos.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Tue 2022-05-10 13:51:40 UTC; 2s ago
    Process: 1121 ExecStart=/bin/bash /home/adrianacosa/accept-http-https.sh (code=exited, status=0/SUCCESS)
   Main PID: 1121 (code=exited, status=0/SUCCESS)

May 10 13:51:40 m3-adrianacosa systemd[1]: Started Servicio de inicializacion del cortafuegos.
May 10 13:51:40 m3-adrianacosa systemd[1]: cortafuegos.service: Succeeded.
adrianacosa@m3-adrianacosa:~$ 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     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  enp1s0 *       0.0.0.0/0            0.0.0.0/0           multiport dports 80,443 state NEW,ESTAB
LISHED

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  --  *      lo      0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      enp1s0  0.0.0.0/0            0.0.0.0/0           multiport sports 80,443 state ESTABLISH
ED
adrianacosa@m3-adrianacosa:~$
```

Hasta aquí tendríamos la configuración de la máquina 3. Para las otras dos máquinas sería exactamente igual pero con el script que hemos hecho en el apartado anterior:

```
[Unit]
Description=Servicio de inicializacion del cortafuegos

[Service]
Type=Simple
ExecStart=/bin/bash /home/adrianacosa/only-m3-firewall.sh

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

Bibliografía

1.- https://www.redeszone.net/gnu-linux/iptables-configuracion-del-firewall-en-linux-con-iptables/

2.- https://www.baeldung.com/linux/run-script-on-startup