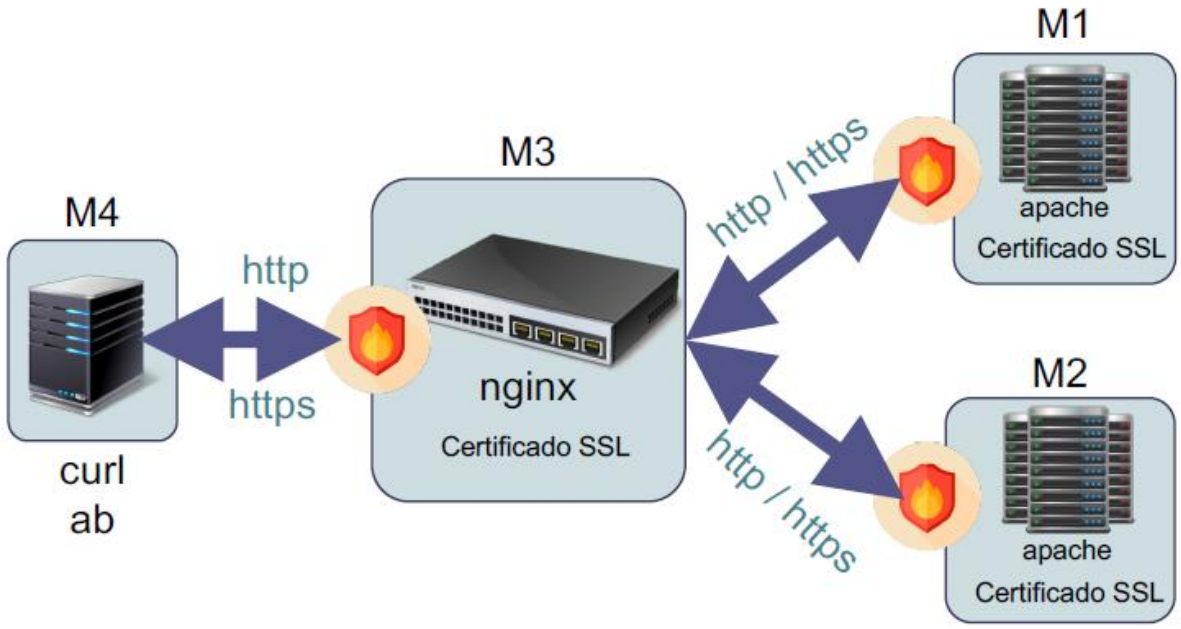
Práctica 4:

# ***Asegurar la granja web***

## *Índice:*

## *1) Introducción*

En esta práctica configuraremos la seguridad de la granja web creada en la práctica anterior, instalando un certificado SSL para configurar el acceso HTTPS a los servidores web y configurando las reglas del cortafuegos para proteger la granja web. El esquema general de la práctica es el siguiente:




El protocolo SSL (Secure Sockets Layer) proporciona servicios de comunicación segura entre cliente y servidor, como, por ejemplo:

- Autenticación (certificados)
- Integridad (firmas digitales)
- Privacidad (encriptación)

Y sirven para brindar seguridad al visitante de su página web, una manera de decirles a sus clientes que el sitio web es auténtico, real y confiable para ingresar datos personales. Estos certificados se pueden generar de diversas formas:

-Mediante una autoridad de certificación (por ejemplo, la Real Casa de Moneda y Timbre).

-Crear nuestros propios certificados SSL auto-firmados usando la herramienta openssl (este será nuestro caso).

-Utilizar certificados del proyecto Certbot (antes Let's Encrypt).

Por otro lado, tenemos el cortafuegos, componente esencial que protege la granja web de accesos indebidos, que actúa como el guardián de la puerta del sistema, permitiendo el tráfico autorizado o denegándolo.

Todos los paquetes TCP/IP deben pasar por el cortafuegos y decidir qué hacer con ellos en base a IPTABLES.

## *2) Instalar un certificado SSL auto-firmado para configurar el acceso por HTTPS:*

La versión actual es la SSLv3, que se considera insegura. El nuevo estándar se llama TLS (Transport Layer Security).

Para generar e instalar un certificado SSL auto-firmado en Ubuntu Server solo debemos activar el módulo SSL de Apache, generar los certificados e indicarle la ruta a los certificados en la configuración. Por tanto, ejecutamos como root en nuestra máquina M1 los siguientes comandos.

Para generar un certificado auto-firmado, activamos el módulo SSL de Apache y reiniciamos el servicio:

*sudo a2enmod ssl*

*sudo service apache2 restart*

Seguidamente debemos crear el directorio que contendrá los certificados dentro de Apache:

*sudo mkdir /etc/apache2/ssl*

```
joselepedraza@m1:~$ sudo a2enmod ssl
[sudo] password for joselepedraza:
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 certifica
tes.
To activate the new configuration, you need to run:
  systemctl restart apache2
joselepedraza@m1:~$ sudo service apache2 restart
joselepedraza@m1:~$ sudo mkdir /etc/apache2/ssl
joselepedraza@m1:~$ _
```

Por último, debemos generar el certificado a través de la herramienta *openssl* con el siguiente comando, esto nos generará un formulario a rellenar con nuestros datos para configurar su dominio:

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

```
joselepedraza@m1:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ss
l/apache.key -out /etc/apache2/ssl/apache.crt
Can't load /home/joselepedraza/.rnd into RNG
140383633981888:error:2406F079:random number generator:RAND_load_file:Cannot open file:../crypto/ran
d/randfile.c:88:Filename=/home/joselepedraza/.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) []:joseperoman
Email Address []:joseperoman@correo.ugr.es
joselepedraza@m1:~$ ls /etc/apache2/ssl/
apache.crt  apache.key
joselepedraza@m1:~$ _
```

Para instalar el certificado después de generar los archivos *.ctr* y *.key* debemos activar en la configuración del servidor la configuración ssl editando el archivo de configuración *default-ssl.conf* de apache:

*sudo vim /etc/apache2/sites-available/default-ssl*

Y añadimos la ruta de los certificados debajo del parámetro *SSLEngine on*:

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

```
#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
```

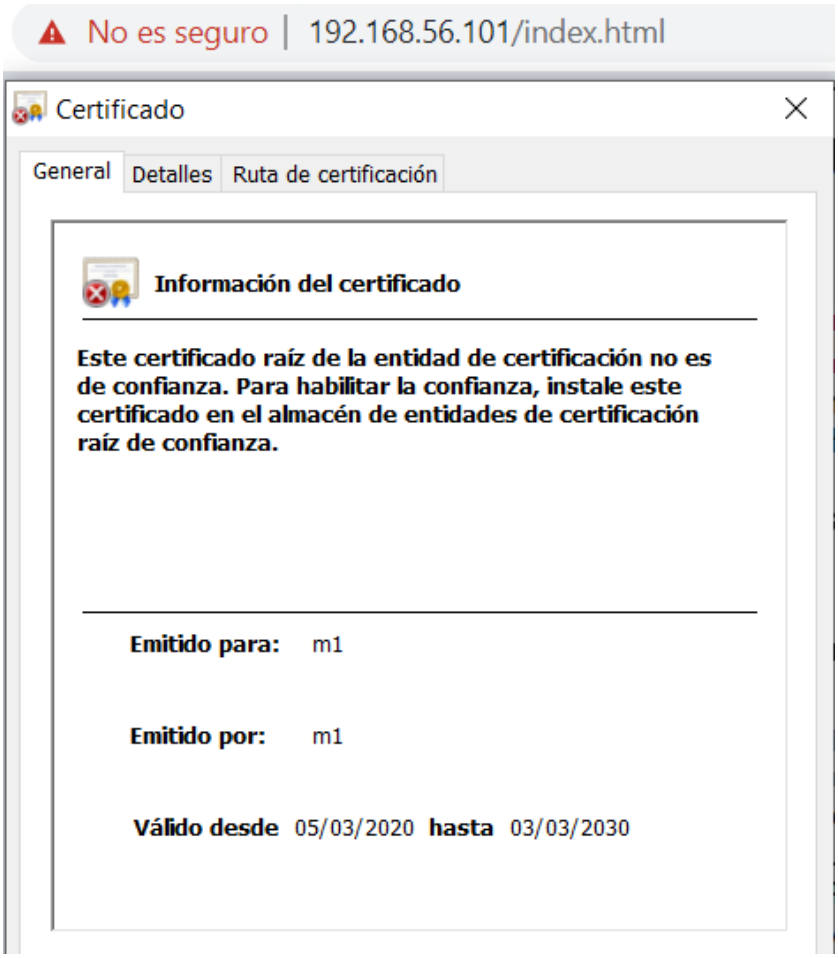
Activamos el sitio *default-ssl* y reiniciamos Apache:

*a2ensite default-ssl*
*service apache2 reload*

```
joselepedraza@m1:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
joselepedraza@m1:~$ sudo service apache2 reload
joselepedraza@m1:~$ _
```

Una vez reiniciado Apache, accedemos al servidor web mediante el protocolo HTTPS y veremos, si estamos accediendo con un navegador web, que en la barra de dirección sale en rojo el https, ya que se trata de un certificado auto-firmado:

```
C:\Users\Josele>curl -k https://192.168.56.101/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m1)
        </body>
</html>
```



Finalmente, y como queremos que la granja nos permita usar el HTTPS, debemos configurar el balanceador para que también acepte este tráfico (puerto 443). Para hacer esto, copiaremos la pareja de archivos generados en la máquina m1 (el *.crt* y *.key)* a todas las máquinas de la granja web. Es muy importante no generar más certificados para trabajar en un principio únicamente con este.

Para copiar estos archivos podemos usar la herramienta *scp* o *rsync*, escribiendo los siguientes comandos (tanto a la máquina m2 como a la máquina m3):

*sudo scp apache.crt joselepedraza@192.168.56.102:/home/joselepedraza/apache.crt*

*sudo scp apache.key joselepedraza@192.168.56.102:/home/joselepedraza/apache.key*

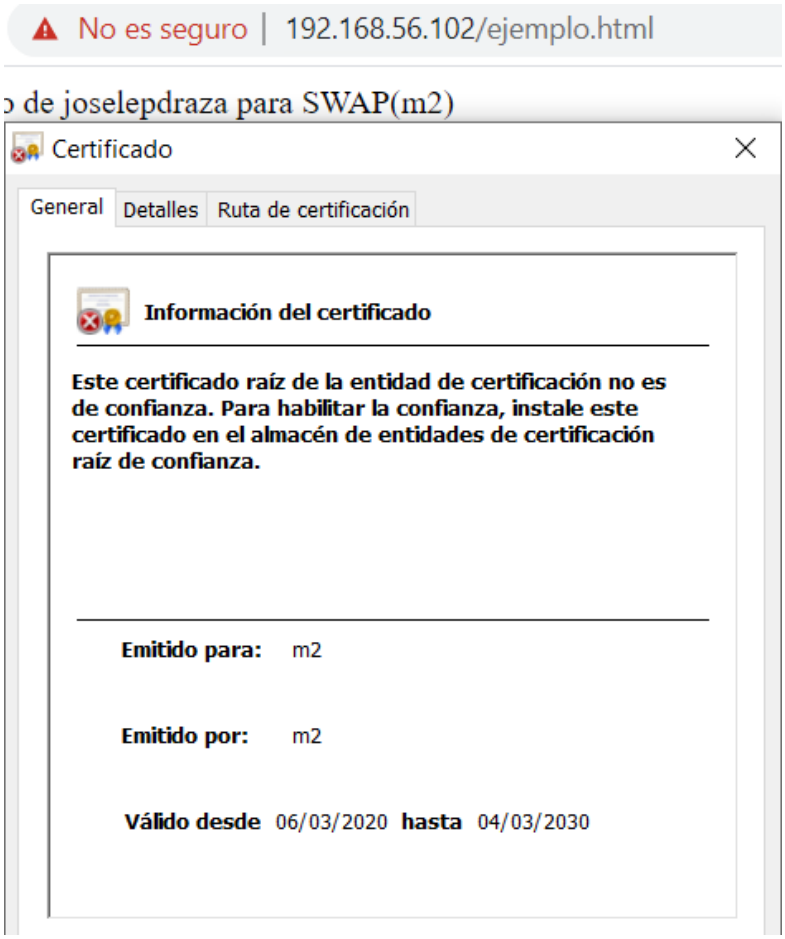
```
joselepedraza@m1:/etc/apache2/ssl$ sudo scp apache.crt joselepedraza@192.168.56.102:/home/joselepedr
aza/apache.crt
[sudo] password for joselepedraza:
joselepedraza@192.168.56.102's password:
apache.crt                                    100% 1448     2.4MB/s   00:00
joselepedraza@m1:/etc/apache2/ssl$ sudo scp apache.key joselepedraza@192.168.56.102:/home/joselepedr
aza/apache.key
joselepedraza@192.168.56.102's password:
apache.key                                    100% 1704     2.1MB/s   00:00
joselepedraza@m1:/etc/apache2/ssl$ _
```

```
joselepedraza@m2:~$ ls
apache.crt  apache.key
joselepedraza@m2:~$ _
```

Una vez copiados los archivos a la máquina m2 debemos crear al igual que en m1, el directorio */etc/apache2/ssl*, copiar los archivos *.crt* y *.key*, configurar *default-ssl.conf* (añadiendo las dos líneas como hicimos en m1), activar el sitio *default-ssl* y por último reiniciar el servicio de Apache. Para ellos ejecutamos en m2 los mismos comandos que los ejecutados anteriormente en m1.

Y como vemos a continuación ya tenemos instalado el certificado en la máquina m2 y podemos acceder a través de HTTPS:

```
C:\Users\Josele>curl -k https://192.168.56.102/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m2)
        </body>
</html>
```




Ahora solo nos queda configurar el balanceador de la máquina m3, donde pondremos la ruta a la carpeta donde hayamos copiado el *apache.crt* y el *apache.key*. Después, en el balanceador Nginx debemos añadir nuevo server al archivo */etc/nginx/conf.d/default.conf* igual que el configurado en la práctica anterior pero ahora añadiendo los siguientes datos:

*listen 443 ssl;*
*ssl on;*
*ssl_certificate /home/usuario/ssl/apache.crt*
*ssl_certificate_key /home/usuario/apache.key*

Copio los archivos *.crt* y *.key* desde la máquina m1 hasta la máquina m3:

```
joselepedraza@192.168.56.103's password:
apache.crt                                                    100% 1448     2.8MB/s   00:00
joselepedraza@m1:/etc/apache2/ssl$ sudo scp apache.key joselepedraza@192.168.56.103:/home/joselepedr
aza/apache.key
joselepedraza@192.168.56.103's password:
apache.key                                                    100% 1704     1.9MB/s   00:00
joselepedraza@m1:/etc/apache2/ssl$ _
```

```
joselepedraza@m3:~$ ls
apache.crt  apache.key
joselepedraza@m3:~$
```

Una vez hecho esto y comprobado que las demás herramientas de balanceo de carga previamente instaladas están desactivadas, procedemos a reconfigurar el archivo de configuración de Nginx añadiendo un nuevo *server* como sigue:

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

        location /
        {
                proxy_pass http://backendApache;
                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/joselepedraza/apache.crt;
        ssl_certificate_key /home/joselepedraza/apache.key;

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

        location /
        {
                proxy_pass http://backendApache;
                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 "";
        }
}_
```

Después de reiniciar el servicio de Nginx, comprobamos que funciona correctamente haciéndole peticiones HTTPS a la dirección IP del balanceador:

```
C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m1)
        </body>
</html>

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m2)
        </body>
</html>

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m1)
        </body>
</html>
```

## 3) *Configurar las reglas del cortafuegos para proteger la granja web:*

En general, todos los paquetes TCP/IP que entren o salgan de la granja web deben pasar por el cortafuegos, que debe examinar y bloquear aquellos que no cumplan los criterios de seguridad establecidos. Estos criterios se configuran mediante un conjunto de reglas, usadas para bloquear puertos específicos, rangos de puertos, direcciones IP, rangos de IP, tráfico TCP o tráfico UDP.

*iptables* es una herramienta de cortafuegos, de espacio de usuario, con la que el superusuario define reglas de filtrado de paquetes, de traducción de direcciones de red y mantiene registros de log. Herramienta construida sobre Netfilter, una parte del núcleo Linux que permite interceptar y manipular paquetes de red.

Para configurar adecuadamente *iptables* en una máquina Linux, conviene establecer como reglas por defecto la denegación de todo el tráfico, salvo el que permitamos después explícitamente. Una vez hecho esto, a continuación, definiremos nuevas reglas para permitir el tráfico solamente en ciertos sentidos necesarios, ya sea de entrada o de salida. Por último, definiremos rangos de direcciones IP a los cuales aplicar diversas reglas, y mantendremos registros (logs) del tráfico no permitido y de intentos de acceso para estudiar más tarde posibles ataques.

A continuación mostraremos cómo utilizar la herramienta para establecer ciertas reglas y filtrar algunos tipos de tráfico, o bien controlar el acceso a ciertas páginas y seguidamente de esta pequeña introducción a los comandos básicos más relevantes, implementaremos ciertos scripts para aplicarlos en nuestras máquinas m1,m2 y m3 para asegurar nuestra granja web.

Para consultar información de la herramienta podemos ver su página del manual:

*man iptables*
*iptables -h*

Para comprobar el estado del cortafuegos, debemos ejecutar:

iptables *-L -n -v*

También se puede parar el cortafuegos y eliminar al mismo tiempo todas sus reglas:

*iptables -F*
*iptables -X*
*iptables -t nat -F*
*iptables -t nat -X*
*iptables -t mangle -F*
*iptables -t mangle -X*
*iptables -P INPUT ACCEPT*
*iptables -P OUTPUT ACCEPT*

Para denegar cualquier tráfico de información, podemos hacer:

*iptables -P INPUT DROP*
*iptables -P OUTPUT DROP*
*iptables -P FORWARD DROP*
*iptables -L -n -v*

Para bloquear el tráfico de entrada, podemos hacer:

*iptables -P INPUT DROP*
*iptables -P FORWARD DROP*
*iptables -P OUTPUT ACCEPT*
*iptables -A INPUT -m state --state NEW, ESTABLISHED -j ACCEPT*
*iptables -L -n -v*

Abrir el puerto 22 para permitir el acceso por SSH:

*iptables -A INPUT -p tcp --dport 22 -j ACCEPT*
*iptables -A OUTPUT -p udp --sport 22 -j ACCEPT*

Abrir los puertos HTTP/HTTPS (80 y 443 respectivamente) para configurar un servidor web:

*iptables -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT*
*iptables -A INPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT*

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*

Bloquear todo el tráfico de entrada/salida para una IP específica:

*iptables -I INPUT -s 150.214.13.13 -j DROP*
*iptables -I OUTPUT -s 31.13.83.8 -j DROP*

Evitar el acceso a www.facebook.com especificando el nombre de dominio:

*iptables -A OUTPUT -p tcp -d www.facebook.com -j DROP*

En algunas ocasiones, en lugar de repetir conjuntos de reglas para diferentes puertos, conviene usar reglas que integren la opción multipuerto (reglas de una única línea):

*iptables -A INPUT -i eth0 -p tcp -m multiport --dports 22,80,443 -m state --state NEW, ESTABLISHED -j ACCEPT*
*iptables -A OUTPUT -o eth0 -p tcp -m multiport --sports 22,80,443 -m state --state ESTABLISHED -j ACCEPT*

Por último, conviene comprobar el funcionamiento del cortafuegos recién configurado. Para ello, pediremos al sistema que nos muestre qué puertos hay abiertos y qué demonios o aplicaciones los tienes en uso con la orden *netstat:*

*netstat -tulpn*
*netstat -tulpn | grep :80* (para mostrar únicamente el estado del puerto 80)

Lo habitual es crear un script que se ejecute en el arranque del sistema. Como veremos a continuación crearé varios scripts para la configuración de seguridad de nuestra granja web.

En primer lugar, configuraremos el cortafuegos en nuestras máquinas servidoras (m1 y m2) y nos crearemos dos scripts por cada una, el primero encargado de crear las reglas del cortafuegos y el segundo encargado de restaurar la configuración original permitiendo todas las conexiones.

En primer lugar, creamos el script *firewallresetconf.sh* en la máquina m1 como vemos a continuación con el comando:

*sudo vim firewallresetconf.sh*

```
#!/bin/bash

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

# Aceptar todas las conexiones
        iptables -P INPUT ACCEPT
        iptables -P OUTPUT ACCEPT
        iptables -P FORWARD ACCEPT

        iptables -L -n -v
~
```

En segundo lugar, creamos el script *iptablesfirewallconf.sh* donde permitiremos únicamente las conexiones a través de los puertos 80 (para http), 443 (para https) y 22 (para ssh) como vemos a continuación:

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

# Politica por defecto: denegar todo el trafico
        iptables -P INPUT DROP
        iptables -P OUTPUT DROP
        iptables -P FORWARD DROP

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

# Abrir el puerto 22 para permitir el acceso por SSH:
        iptables -A INPUT -p tcp --dport 22 -j ACCEPT --source 192.168.56.103
        iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# Permitir el trafico por el puerto 80 (HTTP):
        iptables -A INPUT -p tcp --dport 80 -j ACCEPT --source 192.168.56.103
        iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

# Permitir el tráfico por el puerto 443 (HTTPS):
        iptables -A INPUT -p tcp --dport 443 -j ACCEPT --source 192.168.56.103
        iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT

iptables -L -n -v
```

Este script consta de varias partes:

- ➔ Primeramente, eliminamos las posibles reglas definidas anteriormente.
- ➔ Asumimos una política por defecto de denegación de todo el tráfico.
- ➔ Permitimos el acceso desde localhost.
- ➔ Abrimos los puertos 80, 443 y 22: En este caso, indicamos la fuente desde donde recibe la conexión mediante *--source <ipm3>* en la orden de entrada, asignándole la IP de nuestro balanceador de carga (m3).

Una vez tenemos esta configuración, si alguien intenta conectar directamente con el backend (máquinas servidoras m1 y m2), se le mostrará un error de la siguiente forma: “time out”, permitiendo únicamente la conexión a través de nuestra máquina m3 balanceadora de carga.

Le damos permisos de ejecución y ejecutamos el script *iptablesfirewallconf.sh* y vemos las reglas de *iptables* creadas, donde podemos comprobar que tanto para las conexiones de entrada como las de salida, están abiertos los puertos 80, 443 y 22 con tcp y estamos aceptando las conexiones de entrada y salida con el localhost:

```
joselepedraza@m1:~$ sudo ./iptablesfirewallconf.sh
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     tcp  --  *      *       192.168.56.103       0.0.0.0/0           tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       192.168.56.103       0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *       192.168.56.103       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 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     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
joselepedraza@m1:~$ _
```

Añadimos estos scripts a la máquina m2 con *scp*, lo ejecutamos en m2 y comprobamos su correcto funcionamiento haciendo *curl* desde localhost:

*sudo scp iptablesfirewallconf.sh*
*joselepedraza@192.168.56.102:/home/joselepedraza/iptablesfirewallconf.sh*

*sudo scp firewallresetconf.sh*
*joselepedraza@192.168.56.102:/home/joselepedraza/firewallresetconf.sh*

```
joselepedraza@m1:~$ sudo ./firewallresetconf.sh
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
joselepedraza@m1:~$ sudo scp iptablesfirewallconf.sh joselepedraza@192.168.56.102:/home/joselepedraz
a/iptablesfirewallconf.sh
joselepedraza@192.168.56.102's password:
iptablesfirewallconf.sh                                     100%  897     1.4MB/s   00:00
joselepedraza@m1:~$ sudo scp firewallresetconf.sh joselepedraza@192.168.56.102:/home/joselepedraza/f
irewallresetconf.sh
joselepedraza@192.168.56.102's password:
firewallresetconf.sh                                        100%  257   369.0KB/s   00:00
joselepedraza@m1:~$ _
```

Debemos resetear la configuración previamente aplicada para copiar los scripts a m2 con *scp* (una vez copiados, ejecutamos en m1 y m2 el script *iptablesfirewallconf.sh).*

Comprobamos que las conexiones directas con el backend (m1 y m2) son rechazadas mientras que las conexiones a través del balanceador las acepta mostrando el contenido de las páginas alojadas en m1 y m2, sea por HTTP o HTTPS:

```
C:\Users\Josele>curl 192.168.56.101/ejemplo.html
curl: (7) Failed to connect to 192.168.56.101 port 80: Timed out

C:\Users\Josele>curl -k https://192.168.56.101/ejemplo.html
curl: (7) Failed to connect to 192.168.56.101 port 443: Timed out

C:\Users\Josele>curl 192.168.56.102/ejemplo.html
curl: (7) Failed to connect to 192.168.56.102 port 80: Timed out

C:\Users\Josele>curl -k https://192.168.56.102/ejemplo.html
curl: (7) Failed to connect to 192.168.56.102 port 443: Timed out

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m2)
        </body>
</html>

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m1)
        </body>
</html>
```

Una vez comprobado el correcto funcionamiento del cortafuegos, debemos hacer que el script *iptablesfirewallconf.sh* se ejecute cada vez que la máquina se encienda (tanto en m1 como en m2) ya que por el contrario esta configuración se perdería cada vez que apagamos. Para esto tenemos dos opciones:

Para la primera opción, debemos crear en */etc* el archivo *rc.local* ya que en las distribuciones Ubuntu 18.04 en adelante no lo tenemos disponible. Por tanto, lo creamos como sigue y nos aseguramos que de que tiene permiso de ejecución y de que tiene la estructura correcta como se explica en el siguiente artículo (https://www.claudiokuenzler.com/blog/783/ubuntu-18.04-rc.local-file-not-exist-launch-with-systemd-rc-local).

Todo esto deberemos realizarlo tanto en m1 como en m2 de la misma manera.

Una segunda opción más sencilla es hacerlo a través de la herramienta *crontab* para programar tareas incluyendo en el archivo */etc/crontab* la línea siguiente para que se ejecute nuestro script *iptablesfirewallconf.sh* cada vez que se inicie el sistema (tanto en la máquina m1 como en la máquina m2):

*@reboot root /home/joselepedraza/iptablesfirewallconf.sh*

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# 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 /home/joselepedraza/iptablesfirewallconf.sh
#
```

Después de agregar la misma línea en el archivo *crontab* de la máquina m2, reiniciamos las máquinas y comprobamos que funciona correctamente como vemos a continuación:

```
C:\Users\Josele>curl 192.168.56.101/ejemplo.html
curl: (7) Failed to connect to 192.168.56.101 port 80: Timed out

C:\Users\Josele>curl -k https://192.168.56.101/ejemplo.html
curl: (7) Failed to connect to 192.168.56.101 port 443: Timed out

C:\Users\Josele>curl -k https://192.168.56.102/ejemplo.html
curl: (7) Failed to connect to 192.168.56.102 port 443: Timed out

C:\Users\Josele>curl 192.168.56.102/ejemplo.html
curl: (7) Failed to connect to 192.168.56.102 port 80: Timed out

C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m2)
        </body>
</html>

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m1)
        </body>
</html>
```

Por último, vamos a configurar nuestra máquina m3 balanceadora con *iptables* para que solo acepte peticiones HTTPS y HTTP. Como hemos hecho en las máquinas m1 y m2, crearemos dos scripts de la misma forma, uno que resetea la configuración del cortafuegos y otro donde únicamente se permitan peticiones a través de estos dos tipos de conexiones.

*firewallresetconf.sh:*

```
#!/bin/bash

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

# Aceptar todas las conexiones
        iptables -P INPUT ACCEPT
        iptables -P OUTPUT ACCEPT
        iptables -P FORWARD ACCEPT

iptables -L -n -v
```

*iptablesfirewallconf.sh:*

```
#!/bin/sh

# Eliminar todas las reglas (configuracion limpia)
        ./firewallresetconf.sh

# Politica por defecto: denegar todo el trafico
        iptables -P INPUT DROP
        iptables -P OUTPUT DROP
        iptables -P FORWARD DROP

# Permitir la salida del equipo (output) con conexiones nuevas que solicitemos, conexiones estableci
das y relacionadas. Permitir la entrada (input) 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

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

# Abrir el puerto 22 para permitir el acceso por SSH:
        iptables -A INPUT -p tcp --dport 22 -j ACCEPT
        iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# 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

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


iptables -L -n -v
```

De igual modo que hicimos en las máquinas m1 y m2, programamos la ejecución del script para que se ejecute cada vez que se encienda la máquina m3, agregando al archivo */etc/crontab* la siguiente línea:

@reboot root /home/joselepedraza/iptablesfirewallconf.sh

Reiniciamos la máquina m3 y comprobamos que acepta y sirve las peticiones http y https correctamente:

```
C:\Users\Josele>curl 192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m1)
        </body>
</html>

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html
<html>
        <body>
                Web de ejemplo de joselepdraza para SWAP(m2)
        </body>
</html>

C:\Users\Josele>curl 192.168.56.101/ejemplo.html
curl: (7) Failed to connect to 192.168.56.101 port 80: Timed out

C:\Users\Josele>curl 192.168.56.102/ejemplo.html
curl: (7) Failed to connect to 192.168.56.102 port 80: Timed out

C:\Users\Josele>curl -k https://192.168.56.101/ejemplo.html
curl: (7) Failed to connect to 192.168.56.101 port 443: Timed out

C:\Users\Josele>curl -k https://192.168.56.102/ejemplo.html
curl: (7) Failed to connect to 192.168.56.102 port 443: Timed out
```

Como podemos comprobar desde localhost, las peticiones web a la granja web se sirven únicamente a través del balanceador (máquina m3) y no a través de los servidores finales (máquina m1 y m2).