
Seguridad de la granja web

Andrés Millán Muñoz (*amilmun@correo.ugr.es*)

May 8, 2022

Contents

1	Fe de erratas	2
2	Certificado SSL	4
2.1	Emisión del certificado	4
2.2	Puesta a punto de M2 y M3	7
2.3	Opciones avanzadas	8
2.3.1	Comprobación del certificado	8
2.3.2	Configuración adicional de Apache	9
2.3.3	Configuración adicional de Nginx	10
3	Configuración del firewall	12
3.1	Diseño de las reglas	12
3.2	Ejecución automática del script al arrancar	14
3.2.1	Opciones avanzadas de iptables	15
	Bibliografía	17

En esta práctica, vamos a poner a punto la seguridad de la granja web. Para conseguirlo, instalaremos un certificado SSL para el acceso de HTTPS, y configuraremos un cortafuegos.

Como siempre, las IPs de las máquinas son las siguientes:

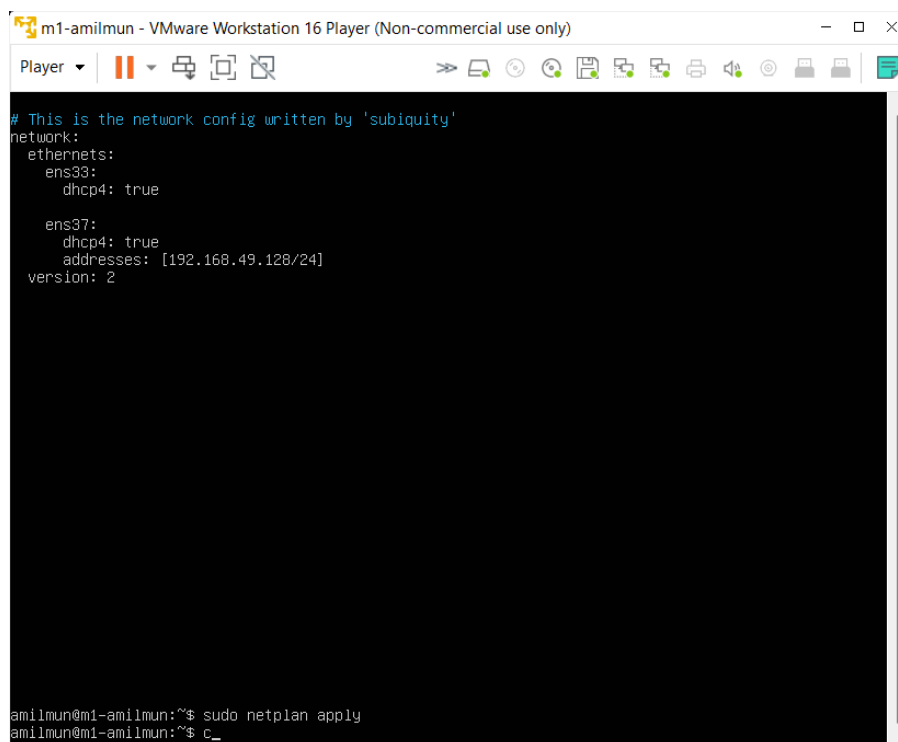
- **M1:** 192.168.49.128.
- **M2:** 192.168.49.129.
- **M3:** 192.168.49.130.

1 Fe de erratas

Antes de comenzar la práctica, voy a explicar por qué todas mis prácticas anteriores eran incorrectas, y por qué me acabo de dar cuenta.

Intentando hacer scp de M1 a M2 para copiar el certificado, me he encontrado con un error extraño. El archivo no se copiaba. O, mejor dicho, se copiaba a la misma máquina aún poniendo la IP correcta. Haciendo más pruebas, nos dimos cuenta de que M1 podía hacer ping a M2, pero M2 no podía a M1. Y desde el localhost, todo funcionaba bien.

¿El error? El netplan estaba mal configurado.



```
m1-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
# This is the network config written by 'subiquity'
network:
  ethernets:
    ens33:
      dhcp4: true

    ens37:
      dhcp4: true
      addresses: [192.168.49.128/24]
  version: 2

amilmun@m1-amilmun:~$ sudo netplan apply
amilmun@m1-amilmun:~$ c_
```

Figure 1.1: Netplan de la práctica 1. Ahora es fácil adivinar el error

Aún limitando la IP a una sola, **dhcp4: true** asigna una adicional. Esta no figura en **ifconfig**, pero sí en **ip a**.

Es por estos motivos que SSH dio problemas al pasar de la práctica 1 a la 2; `rsync` no funcionó en la práctica 2; y es posible que no estuviéramos haciendo balanceo de carga en la práctica 3. Podría ser que desde una máquina externa sí funcione correctamente como es el caso de localhost, pero no es seguro.

Desgracias de la configuración de un sever.



Figure 1.2: xkcd.com/1084

2 Certificado SSL

Antes de empezar, generaremos un certificado SSL autofirmado desde la máquina M1, copiándolo a M2 y M3 mediante `scp`.

2.1 Emisión del certificado

Primero, debemos activar el módulo SSL de Apache:

```
1 sudo a2enmod ssl
2 sudo systemctl restart apache2 # Debemos reiniciar el servicio
```

Creamos el directorio de certificados:

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

Generemos un certificado autofirmado, llamado `apache_amilmun.crt`, con clave `apache_amilmun.key` y con una duración de 1 año. Debemos introducir también los datos del certificado apropiados en la creación:

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

```

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
[2]- Done          sudo a2enmod ssl
amilmun@mi-amilmun:~$ systemctl restart apache2
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to restart 'apache2.service'.
Authenticating as: Andrés Millán Muñoz (amilmun)
Password:
==== AUTHENTICATION COMPLETE ====
amilmun@mi-amilmun:~$ sudo mkdir /etc/apache2/ssl
amilmun@mi-amilmun:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache_amilmun.key -out /etc/apache2/ssl/apache_amilmun.crt
Can't load /home/amilmun/.rnd into RNG
140521804284352:error:2406F079:random number generator:RAND_load_file:Cannot open file:../crypto/rand/randfile.c:88:Filename=/home/amilmun/.rnd
Generating a RSA private key
.....+++++
+++++
writing new private key to '/etc/apache2/ssl/apache_amilmun.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) []:amilmun
Email Address []:amilmun@correo.ugr.es
amilmun@mi-amilmun:~$

```

Figure 2.1: Para los datos del certificado, hemos usado la ciudad de Granada, con organización UGR, sección P4, y datos personales propios

Ahora debemos configurar correctamente apache para que use el certificado. Para lograrlo, editamos el archivo `/etc/apache2/sites-available/default-ssl.conf`, y agregamos:

```

1 SSLCertificateFile /etc/apache2/ssl/apache_amilmun.crt
2 SSLCertificateKeyFile /etc/apache2/ssl/apache_amilmun.key

```

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

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

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

# Nuestros certificados!
SSLCertificateFile /etc/apache2/ssl/apache_amlmun.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache_amlmun.key_

# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

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

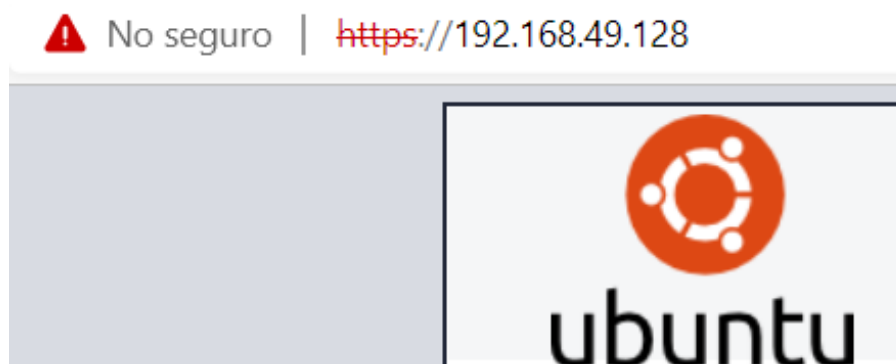
# Certificate Authority (CA):
```

Figure 2.2: Configuración de Apache

Finalmente, debemos activar **default-ssl** y reiniciar apache:

```
1 sudo a2ensite default-ssl
2 sudo systemctl reload apache2
```

Accediendo desde el navegador, podemos ver que se ha cargado correctamente la página

**Figure 2.3:** Acceso por HTTPS

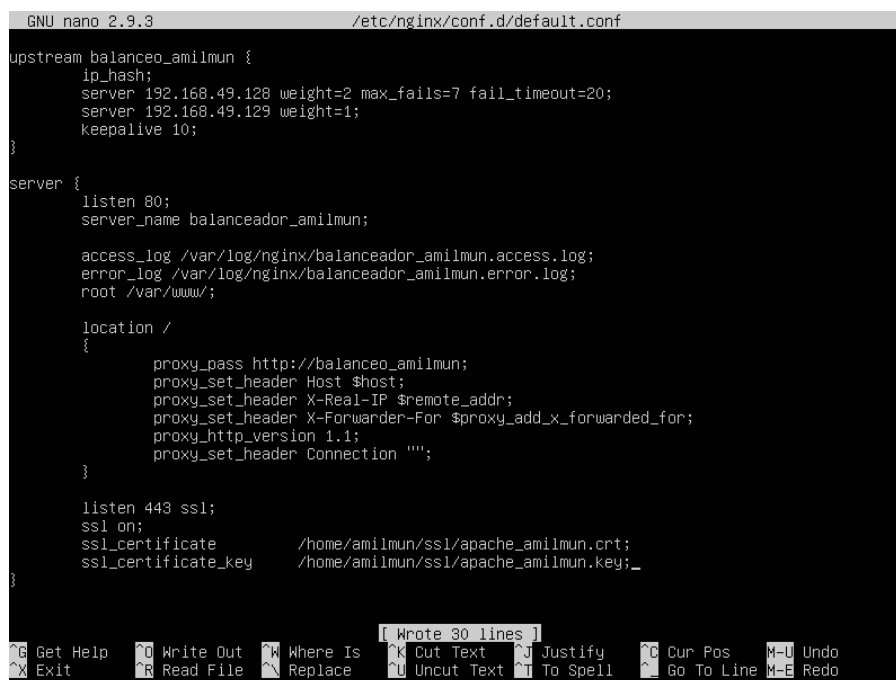
2.2 Puesta a punto de M2 y M3

Ahora, debemos copiar el certificado generado a las máquinas M2 y M3. Usamos `scp` para lograrlo:

```
1 # M1 -> M2
2 sudo scp /etc/apache2/ssl/apache_amilmun.crt amilmun@192.168.49.129:/
  etc/apache2/ssl/apache_amilmun.crt
3 sudo scp /etc/apache2/ssl/apache_amilmun.key amilmun@192.168.49.129:/
  etc/apache2/ssl/apache_amilmun.key
4
5 # M1 -> M3
6 # Desde M3, podemos hacer
7 sudo scp -r amilmun@192.168.49.128:/etc/apache2/ssl /home/amilmun/ssl
```

Cuando lo hagamos, en M2 debemos editar la configuración, activar el módulo y reiniciar Apache, como en M1; mientras que en M3, añadiremos los parámetros pertinentes a la configuración de Nginx:

```
1 listen 443 ssl;
2 ssl on;
3 ssl_certificate /home/amilmun/ssl/apache_amilmun.crt;
4 ssl_certificate_key /home/amilmun/ssl/apache_amilmun.key;
```



```
GNU nano 2.9.3 /etc/nginx/conf.d/default.conf
upstream balanceo_amilmun {
    ip_hash;
    server 192.168.49.128 weight=2 max_fails=7 fail_timeout=20;
    server 192.168.49.129 weight=1;
    keepalive 10;
}

server {
    listen 80;
    server_name balanceador_amilmun;

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

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

    listen 443 ssl;
    ssl on;
    ssl_certificate /home/amilmun/ssl/apache_amilmun.crt;
    ssl_certificate_key /home/amilmun/ssl/apache_amilmun.key;
}

[ Wrote 30 lines ]
G Get Help  O Write Out  W Where Is  K Cut Text  J Justify  C Cur Pos  M-U Undo
X Exit      R Read File  N Replace  U Uncut Text  T To Spell  G Go To Line M-E Redo
```

Figure 2.4: Configuración de Nginx

De esta forma, se puede acceder a <https://192.168.49.130>. Muestra un error de certificado, lo cual es normal; pues no está distribuido por un agente de confianza:

2.3 Opciones avanzadas

2.3.1 Comprobación del certificado

Podemos verificar el certificado emitido escribiendo

```
1 openssl x509 -noout -text -in /etc/apache2/ssl/apache_amilmun.crt
```

Esta opción nos permite conocer a fondo el certificado, y ver cómo se ha generado.

```
f4:c5:d4:c4:45:d2:b6:20:5b:39:99:6d:93:bb:ef:
35:7f:e1:65:02:a4:e5:36:58:c7:b9:14:22:e7:a9:
37:68:6c:30:78:b9:90:b5:d5:d4:13:67:49:b2:14:
c0:27:db:8a:06:f7:dc:6c:f2:b8:2b:a7:94:ee:68:
68:13:5c:55:b7:a4:f6:26:c4:00:b0:40:cf:68:a9:
dd:ec:f4:ce:29:3b:f0:cb:3c:96:4b:5d:ad:95:07:
ac:ad:a8:9f:fd:38:97:3d:10:fe:22:01:20:53:d5:
91:f9:f0:2c:6b:43:22:5d:ac:8f:f9:ad:2f:de:bc:
37:c4:ce:f5:3d:b0:23:75:4e:a3:c6:4f:4a:7b:54:
23:93:cb:62:a7:3b:3d:28:3f:54:9d:5f:68:cb:10:
59:2f
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
FF:93:14:8C:B3:50:A6:0A:6E:3E:FD:D7:2C:9E:90:E0:86:19:84:E6
X509v3 Authority Key Identifier:
keyid:FF:93:14:8C:B3:50:A6:0A:6E:3E:FD:D7:2C:9E:90:E0:86:19:84:E6

X509v3 Basic Constraints: critical
CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
59:e5:ee:a2:a1:07:cd:85:60:7d:13:67:5e:71:62:ad:6b:84:
0f:dd:d8:8a:fc:c2:0b:fe:ae:37:18:6e:bf:b9:88:19:41:8f:
38:94:80:87:d2:39:ed:db:7b:b8:45:9a:4b:0d:b8:7e:2c:37:
57:ea:ba:65:ff:8e:0f:26:62:4f:43:ba:15:68:8f:cc:4d:15:
a6:4c:cf:11:0e:5a:22:5a:86:94:4e:30:03:16:63:fd:36:ad:
62:62:f3:07:ec:c0:76:08:93:e1:4c:c1:a8:50:74:80:9d:f9:
8c:ac:30:77:d2:b9:55:72:4d:b8:f3:01:40:32:7c:ed:78:69:
d0:25:79:34:8d:ed:17:a1:89:af:8e:5e:83:38:12:59:0a:4b:
a9:c0:a5:4e:68:67:a9:51:68:85:d7:48:ae:6c:4a:6c:85:7c:
8a:d3:ad:27:d7:91:ca:ee:cb:ac:0d:ad:be:b0:b2:5d:d0:05:
2d:16:9d:66:5f:e9:71:a9:72:15:62:30:71:30:4a:c4:de:a6:
a5:b6:99:84:24:0a:e6:7b:1f:e5:fc:e5:dc:41:ea:60:c3:92:
4c:da:c4:bd:7d:0b:c7:c5:b4:81:6f:5b:6e:b8:b4:56:46:15:
e9:5a:84:d3:cf:5c:de:ed:73:97:1b:f7:f8:44:23:fc:37:9d:
3b:00:33:35
amilmun@m1-amilmun:/etc/apache2/ssl$
```

Figure 2.5: Parte final de la salida del certificado

También podemos escribir directamente en la orden de la generación de openssl los datos del certificado, utilizando la opción `-subj`. Por ejemplo, para añadir la organización, haríamos

```
1 sudo openssl req -x509 ... -subj /O=UGR/OU=P4
```

Para más información, ver ([Digicert n.d.](#))

Alternativamente, podemos comprobar el estado del certificado gracias a `openssl`. Desde el localhost, hacemos

```
1 openssl s_client -connect {ip máquina}:443 -showcerts
```

Por ejemplo, para M1 se obtiene

```
c/Users/Andre took 4s
> openssl s_client -connect 192.168.49.128:443 -showcerts

CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = m1-amilmun
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = m1-amilmun
verify return:1
---
Certificate chain
 0 s:CN = m1-amilmun
  i:CN = m1-amilmun
-----BEGIN CERTIFICATE-----
MIIC3DCCAcSgAwIBAgIUU7HgYZQw4FZJN90rWtSQNt67/2kwDQYJKoZIhvcNAQEL
BQAwFTETMBEGA1UEAwKbTETtYW1pbG11bjAeFw0yMjAzMDcxMjI3MzdaFw0zMjAz
MDQxMjI3MzdaMBUxEzARBgNVBAMMCm0xLWFtaWxtdW4wggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQDO/mycdvNwLzOjb5soDRbM3wLSudmTEEXVEVJT1+Uj
+11kJLosgOCPe3ZDjUsPIaWaiNulFecqVNcsvgFm30EzwxFXOQLC8eH2EgYFBaPQ
1HywYfQe5QvdssAmNJonMfOrUtD0hGu2yZM6T1/mtK0gZV2uL9lLwe7vVEZZV41g
ngTZmUgCb4id4ADxvIgrpG0yPmCkMoojS8wcdJsAhYXWXTmhzrp5jaf9WKQA4YYM
K1bz4zLKPPp3hlOT+myU38nUIUt74gAGZo3b7+fYlZE5Lq6wa+apVwDdde6Yarmi
```

Figure 2.6: Openssl permite comprobar el certificado. La salida está cortada.

2.3.2 Configuración adicional de Apache

Aunque estas opciones no las acabaremos usando, existen algunos parámetros interesantes que podemos editar.

Uno de ellos es la redirección a HTTPS desde HTTP. Para lograrlo, podemos editar la configuración `/etc/apache2/sites-available/000-default.conf` del puerto 80, escribiendo ([DigitalOcean n.d.](#))

```
1 <VirtualHost *:80>
2     Redirect "/" "https://{IP}/"
3 </VirtualHost>
```

```
GNU nano 2.9.3 /etc/apache2/sites-available/000-default.conf Modified
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

# 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

# Redirección a HTTPS
Redirect "/" "https:///"
</VirtualHost>

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

Figure 2.7: Dado que no tenemos un nombre para el servidor, no podemos completar correctamente la configuración.

2.3.3 Configuración adicional de Nginx

Las operaciones de SSL consumen recursos adicionales. Una de las más costosas es el handshake. Para minimizarlas, podemos habilitar que las conexiones *keepalive* puedan mandar varias peticiones a la misma conexión. Otra opción es reutilizar los parámetros de la sesión SSL para evitar los handshakes en operaciones paralelas y subsecuentes ([Nginx n.d.](#)).

Estos parámetros se pueden editar en el archivo `/etc/nginx/conf.d/default.conf`:

```
        proxy_pass http://balanceo_amilmun;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarder-For $proxy_add_x_forwarded_for;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }

    listen 443 ssl;
    ssl on;
    ssl_certificate      /home/amilmun/ssl/apache_amilmun.crt;
    ssl_certificate_key  /home/amilmun/ssl/apache_amilmun.key;

    ssl_session_cache    shared:SSL:10m;
    ssl_session_timeout  10m;
    keepalive_timeout    70;
}

amilmun@m3-amilmun:~$ sudo service restart nginx
restart: unrecognized service
amilmun@m3-amilmun:~$ sudo systemctl restart nginx
amilmun@m3-amilmun:~$
```

Figure 2.8: Los parámetros que utilizamos son `ssl_session_cache`, `ssl_session_timeout` y `keepalive_timeout`

3 Configuración del firewall

Todo servidor que se precie debe tener un cortafuegos configurado. En esta sección, pondremos en marcha el nuestro utilizando `iptables`.

3.1 Diseño de las reglas

El plan será denegar todo tipo de conexión por defecto, y entonces, habilitaremos las que a nosotros nos interesen. En este caso, serán las conexiones (puesto que estamos en un servidor) y el tráfico proveniente de SSH, HTTP y HTTPS.

Para ello, crearemos un script en alguna de las máquinas o el localhost con el siguiente contenido:

```
1 #!/bin/bash
2
3 # Eliminar cualquier configuración anterior
4 iptables -F
5 iptables -X
6
7 # Denegar por defecto el tráfico
8 iptables -P INPUT DROP
9 iptables -P FORWARD DROP
10 iptables -P OUTPUT DROP
11
12 # Habilitar conexión con el localhost (interfaz lo)
13 iptables -A INPUT -i lo -j ACCEPT
14 iptables -A OUTPUT -o lo -j ACCEPT
15
16 # Permitir conexiones
17 iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
18 iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
19
20 # Permitir SSH
21 iptables -A INPUT -p tcp --dport 22 -j ACCEPT
22 iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
23
24 # Permitir HTTP
25 iptables -A INPUT -p tcp --dport 80 -j ACCEPT
26 iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
27
```

```

28 # Permitir HTTPS
29 iptables -A INPUT -p tcp --dport 443 -j ACCEPT
30 iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT

```

Este script debemos traspassarlo a cada máquina con `scp`, como hicimos con los certificados. La ruta será `/home/amilmun/.iptables/iptables_script.sh`.

```

SWAP/practicas on master [$] took 2s
) scp iptables_script.sh amilmun@192.168.49.128:/home/amilmun/.
iptables_script.sh
amilmun@192.168.49.128's password:
iptables_script.sh      100% 801 275.0KB/s 00:00

```

Podemos ejecutarlo con `sudo ./iptables_script.sh` en cada una.

Tras ejecutarlo, podemos ver que no podemos hacer ping a M1:

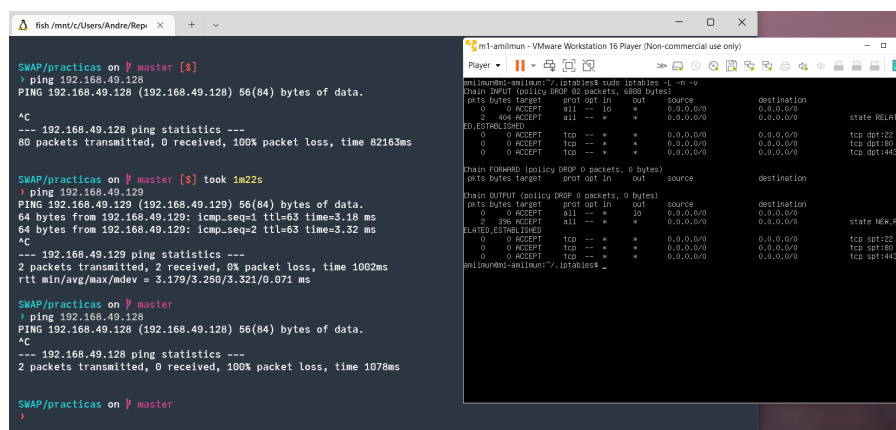


Figure 3.1: Pings denegados por iptables

Mientras que `curl` funciona perfectamente:

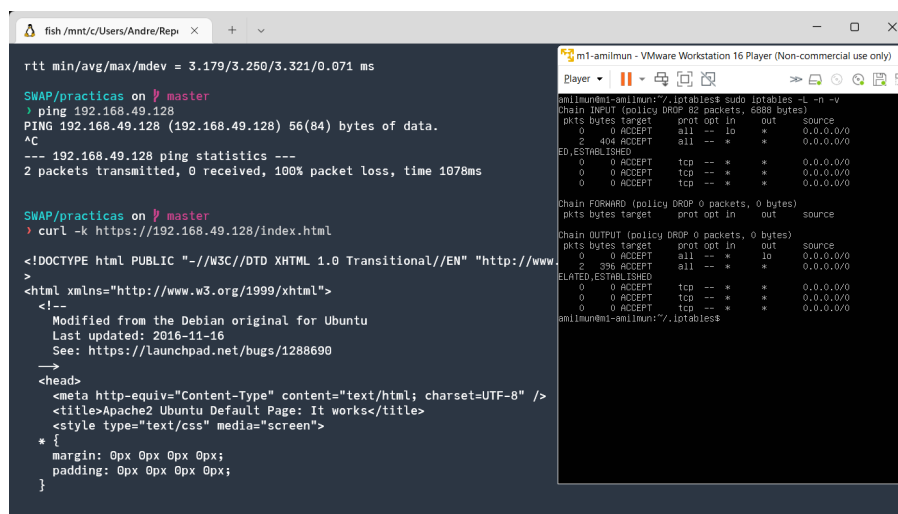


Figure 3.2: Sin embargo, el tráfico HTTPS funciona

3.2 Ejecución automática del script al arrancar

Ejecutar un script al inicio del sistema es sencillo. Para conseguirlo, creamos el archivo `/etc/rc.local` con el siguiente contenido:

```
1 #!/bin/sh -e
2 /home/amilmun/.iptables/iptables_script.sh
3 exit 0
```

Hacemos `sudo chmod +x /etc/rc.local` y listo.

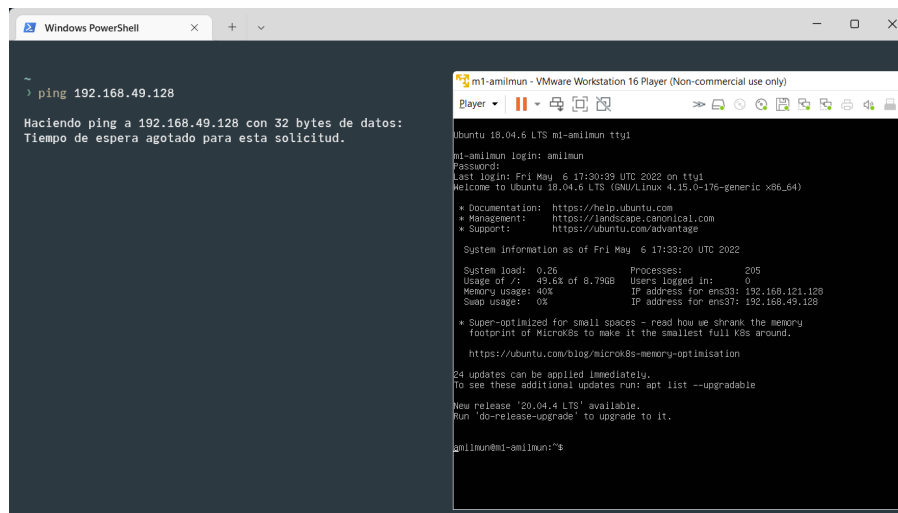


Figure 3.3: Las reglas se aplican automáticamente. Vemos que no podemos hacer ping

3.2.1 Opciones avanzadas de iptables

Entre las opciones propuestas, vamos a documentar un par: que M1 y M2 solo acepten peticiones desde M3 y habilitar `ping`.

Para permitir el tráfico HTTP(S) de M1 y M2 exclusivamente desde M3, debemos modificar las reglas

```
1 iptables -A INPUT -p tcp --dport 80 -j ACCEPT
2 iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
3 iptables -A INPUT -p tcp --dport 443 -j ACCEPT
4 iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT
```

para que incluyan la IP de M3 (192.168.49.130):

```
1 iptables -A INPUT -p tcp --dport 80 -s 192.168.49.130 -j ACCEPT
2 iptables -A OUTPUT -p tcp --sport 80 -d 192.168.49.130 -j ACCEPT
3 iptables -A INPUT -p tcp --dport 443 -s 192.168.49.130 -j ACCEPT
4 iptables -A OUTPUT -p tcp --sport 443 -d 192.168.49.130 -j ACCEPT
```

Para permitir los pings, debemos usar las siguientes reglas (Vivek Gite 2022):

```
1 iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
2 iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

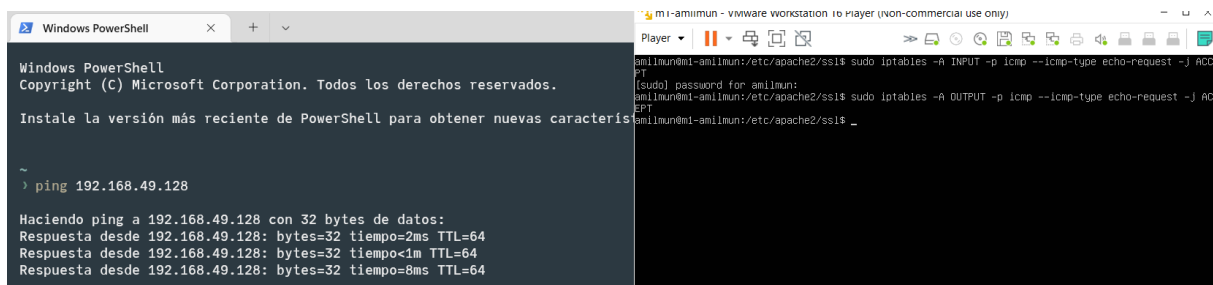


Figure 3.4: Ahora M1 acepta pings

Bibliografía

- Digicert. n.d. "OpenSSL Quick Reference Guide." Accessed May 8, 2022. <https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm#Usingthe-subjSwitch>.
- DigitalOcean. n.d. "Cómo Crear Un Certificado SSL Autofirmado Para Apache En Ubuntu 18.04." Accessed May 8, 2022. <https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-18-04-es>.
- Nginx. n.d. "Configuring HTTPS Servers." Accessed May 8, 2022. https://nginx.org/en/docs/http/configuring_https_servers.html.
- Vivek Gite. 2022. "Linux Iptables Allow or Block ICMP Ping Request." April 14, 2022. <https://www.cyberciti.biz/tips/linux-iptables-9-allow-icmp-ping.html>.