

SWAP: Asegurar la granja web

David Cabezas Berrido

dxabezas@correo.ugr.es

5 de mayo de 2021

Índice

1. Instalar un certificado SSL autofirmado para configurar el acceso por HTTPS	2
2. Configurar la granja web para que permita HTTPS	3
3. Certificados para Apache y NGINX con Certbot	5
3.1. Certificado para Apache	5
3.2. Certificado para NGINX	6
4. Cortafuegos <i>iptables</i>	6

1. Instalar un certificado SSL autofirmado para configurar el acceso por HTTPS

Toda esta sección la haremos en la máquina 1.

Para habilitar el módulo SSL de Apache2, ejecutamos la siguiente línea.

```
sudo a2enmod ssl
```

Habilita el módulo y sus dependencias. Salida:

```
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
```

Restauramos el servicio con `sudo systemctl restart apache2`. Ahora creamos una carpeta para los certificados de Apache, y creamos un par de clave y certificado. Le ponemos longitud de clave 2048 bits y 365 de validez.

```
dxabezas@n1-dxabezas:~$ sudo mkdir /etc/apache2/ssl
dxabezas@n1-dxabezas:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache_dxabezas.key
-out /etc/apache2/ssl/apache_dxabezas.crt
Can't load /home/dxabezas/.rnd into RNG
140072061030848:error:2406F079:random number generator:RAND_load_file:Cannot open file:../crypto/rand/randfile.c:88:Filename=
/home/dxabezas/.rnd
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/etc/apache2/ssl/apache_dxabezas.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) []:dxabezas
Email Address []:dxabezas@correo.ugr.es
dxabezas@n1-dxabezas:~$ ls /etc/apache2/ssl/
apache_dxabezas.crt  apache_dxabezas.key
```

Figura 1: Rellenamos los datos del certificado como se indica en el guión. Comprobamos que se ha creado el par correctamente.

Como opciones avanzadas comentamos que `-x509` auto-firma el certificado, se obtendría una solicitud de certificado si no usásemos esta opción. Además, la opción `-subj /C=TheCountry/CN=theCommonName/ST=theState/O=theOrganization/...` permite especificar los datos desde la orden, pueden consultarse las abreviaturas en [este post](#) se encuentran los distintos atributos y sus abreviaturas.

Ahora modificamos el fichero de configuración `/etc/apache2/sites-available/default-ssl.conf`, tenemos que tener el siguiente bloque (SSLEngine on ya estaba puesto).

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

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

También tenemos que comentar las líneas que sobrescriben estas directivas más abajo. Guardamos los cambios y ejecutamos

```
sudo a2ensite default-ssl
sudo service apache2 reload
```

Cuando accedemos a la página, nos avisa de que es insegura porque el certificado es auto-firmado. Debemos permitir la excepción en el navegador o añadir `-k` con `curl`. Si le damos al candado junto a la dirección y a **More Information**, podemos visualizar el certificado que hemos creado.

Certificate	
dxabezas	
Subject Name	
Country	ES
State/Province	Granada
Locality	Granada
Organization	SWAP
Organizational Unit	P4
Common Name	dxabezas
Email Address	dxabezas@correo.ugr.es
Issuer Name	
Country	ES
State/Province	Granada
Locality	Granada
Organization	SWAP
Organizational Unit	P4
Common Name	dxabezas
Email Address	dxabezas@correo.ugr.es

Figura 2: Certificado con los datos que hemos creado.

Como opciones avanzadas, mostramos como obtener el certificado sin ayuda del navegador, con `openssl`:

```
openssl s_client -connect 192.168.56.101:443 -showcerts
```

También hay varias opciones adicionales en la configuración de Apache2 SSL. Se activan con

```
SSLOptions +opcion1 +opcion2
```

Por ejemplo, cuando se trabaja con autenticación y se requiere que los clientes también tengan certificados, la opción `FakeBasicAuth` requiere que los clientes pongan el campo Subject the su certificado como usuario, la contraseña siempre es la misma: “xxj31ZMTZzkVA” (que es una encriptación por DES de la palabra “password”), por ello el nombre de Fake.

2. Configurar la granja web para que permita HTTPS

Primero replicamos la configuración en la máquina 2. En lugar de crear un nuevo certificado, copiamos el de M1. Desde M1:

```
sudo scp /etc/apache2/ssl/apache_dxabezas.crt dxabezas@192.168.56.102:/home/dxabezas/apache_dxabezas.crt
sudo scp /etc/apache2/ssl/apache_dxabezas.key dxabezas@192.168.56.102:/home/dxabezas/apache_dxabezas.key
```

Ahora desde M2:

```
sudo mkdir /etc/apache2/ssl
sudo mv /home/dxabezas/apache_dxabezas.* /etc/apache2/ssl/
sudo a2enmod ssl & sudo service apache2 restart
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Escribimos las líneas

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

y comentamos las equivalentes que ya había. Terminamos con

```
sudo a2ensite default-ssl
sudo service apache2 reload
```

Cuando nos conectamos a M2 desde el navegador por HTTPS obtenemos el mismo resultado, y también podemos visualizar el mismo certificado.

Falta configurar M3. Otra vez desde M1 copiamos el certificado:

```
sudo scp /etc/apache2/ssl/apache_dxabezas.crt dxabezas@192.168.56.103:/home/dxabezas/apache_dxabezas.crt
sudo scp /etc/apache2/ssl/apache_dxabezas.key dxabezas@192.168.56.103:/home/dxabezas/apache_dxabezas.key
```

Y desde M3:

```
mv apache_dxabezas.* ssl/
sudo nano /etc/nginx/conf.d/default.conf
```

Y creamos un nuevo server como se indica en el guión:

```
server {
    listen 443 ssl;
    ssl on;
    ssl_certificate      /home/dxabezas/ssl/apache_dxabezas.crt;
    ssl_certificate_key  /home/dxabezas/ssl/apache_dxabezas.key;
    server_name balanceador_dxabezas;
    access_log /var/log/nginx/balanceador_dxabezas.access.log;
    error_log /var/log/nginx/balanceador_dxabezas.error.log;
    root /var/www/;

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

Tras restaurar NGINX, podemos acceder desde el navegador por HTTPS a la IP de M3 de la misma forma que hacíamos con M1 y M2 (se marca como insegura y hay que añadir una excepción). Una vez entramos, M1 y M2 se turnan para servirnos su `index.html`. También podemos ver el mismo certificado tal y como hacíamos antes.

Al igual que antes, todas las máquinas aceptan también tráfico HTTP.

Como opciones avanzadas, comentaremos [protocolos](#) y [cifrados](#) para SSL.

Dentro del archivo de configuración del server (NGINX), la directiva `ssl_protocols` limita las conexiones a las compatibles con las versiones de SSL y TLS que indiquemos de la siguiente lista: SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2, TLSv1.3. Por defecto, las versiones que se aceptan son las de este ejemplo:

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
```

Igualmente, la directiva `ssl_ciphers` limita las conexiones a las compatibles con los sistemas de cifrado que se especifiquen, deben escribirse en un formato entendible por OpenSSL, por ejemplo:

```
ssl_ciphers ALL:!aNULL:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

3. Certificados para Apache y NGINX con Certbot

Instalamos Certbot en M1 y M3 con

```
sudo apt install certbot
```

3.1. Certificado para Apache

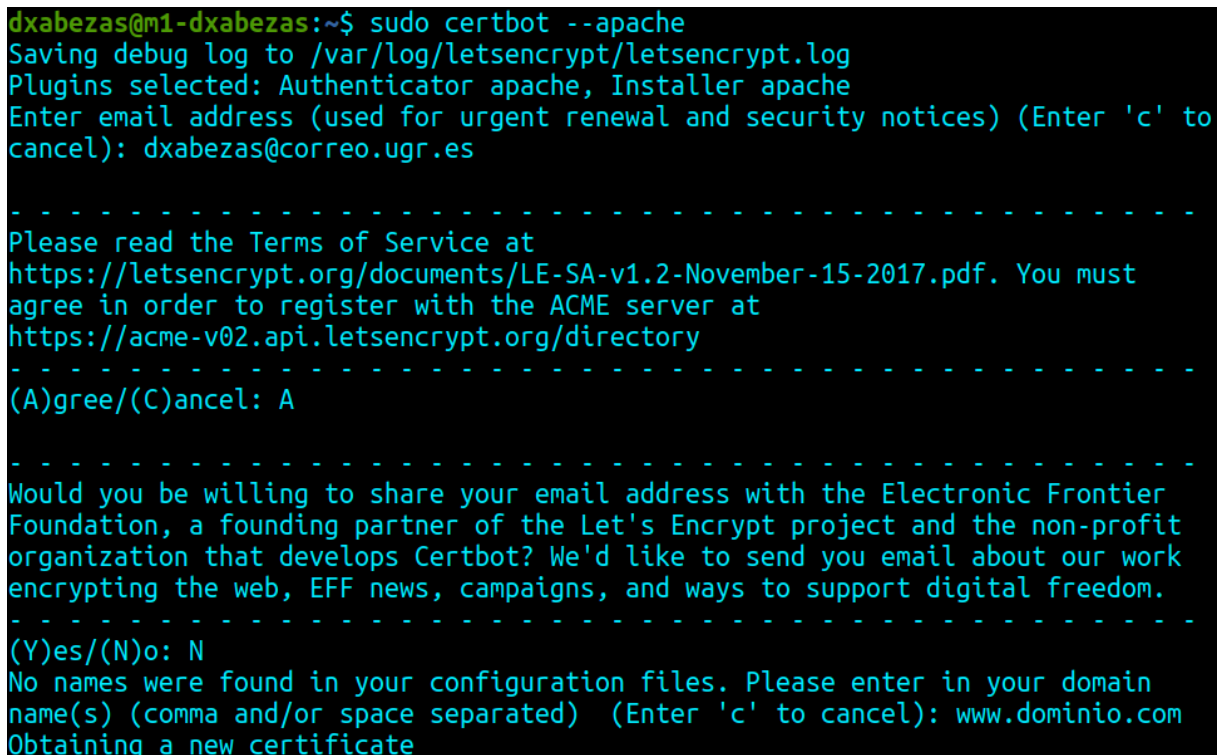
Es necesario instalar el plugin de Apache en M1. Ejecutamos:

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

Ahora, siguiendo las instrucciones de [esta página](#), podemos obtener e instalar un certificado utilizando la siguiente orden.

```
sudo certbot --apache
```

Nos pide algunos datos que debemos rellenar.



```
dxabezas@m1-dxabezas:~$ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Enter email address (used for urgent renewal and security notices) (Enter 'c' to
cancel): dxabezas@correo.ugr.es

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

-----
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
-----
(Y)es/(N)o: N
No names were found in your configuration files. Please enter in your domain
name(s) (comma and/or space separated) (Enter 'c' to cancel): www.dominio.com
Obtaining a new certificate
```

Figura 3: Rellenamos los datos para generar e instalar un certificado para que un sitio web acepte tráfico HTTPS. Este comando genera el certificado y realiza la configuración necesaria.

Sin embargo, Certbot sólo genera certificados para nombres de dominio, lo que dificulta en gran medida que llevemos esto a la práctica.

Como opciones avanzadas, en la misma página explica qué debemos hacer si sólo queremos generar el certificado y queremos llevar a cabo la configuración a mano.

```
sudo certbot certonly --apache
```

También es interesante comentar que por defecto Certbot instala un comando Crontab que renueva los certificados automáticamente. En nuestro caso, nos ha creado el siguiente archivo.

```
dxabezas@m1-dxabezas:~$ cat /etc/cron.d/certbot
# /etc/cron.d/certbot: crontab entries for the certbot package
#
# Upstream recommends attempting renewal twice a day
#
# Eventually, this will be an opportunity to validate certificates
# haven't been revoked, etc. Renewal will only occur if expiration
# is within 30 days.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
0 */12 * * * root test -x /usr/bin/certbot -a \! -d /run/systemd/system
&& perl -e 'sleep int(rand(43200))' && certbot -q renew
```

3.2. Certificado para NGINX

Primero instalamos (en M3) el plugin de NGINX.

```
sudo apt install python-certbot-nginx
```

Ahora, siguiendo las instrucciones de [esta página](#), ejecutamos:

```
sudo certbot --nginx
```

Y nos pide los mismos datos que para el de Apache.

También podemos usar

```
sudo certbot certonly --nginx
```

si sólo queremos generar el certificado para configurarlo manualmente.

4. Cortafuegos *iptables*

Nota: Las IP de las máquinas 1 y 2 son 192.168.56.102 y 192.168.56.101 en lo que sigue.

El cortafuegos *iptables* ya viene instalado en todas las máquinas.

Para configurarlo debemos escribir primero las reglas más generales y después las más específicas, ya que tiene prioridad la última regla que se ejecuta (como es lógico). Comenzamos con un script que es equivalente a desactivar el cortafuegos, borra todas las reglas y permite todo el tráfico.

```
#!/bin/sh

# off.sh
# Eliminar todas las reglas, permitir todo el tráfico

iptables -F
iptables -X
iptables -Z
iptables -t nat -F
iptables -t nat -X
iptables -t nat -Z
```

```
iptables -t mangle -F
iptables -t mangle -X
iptables -t mangle -Z
```

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

Una cadena es una lista de reglas que se aplican a una serie de paquetes.

La opción `-F` elimina todas las reglas de todas las cadenas (ya que no se ha especificado ninguna cadena en concreto), `-X` elimina las cadenas y `-Z` pone a cero los contadores de bytes transmitidos.

La opción `-t` indica la tabla a usar. Las distintas tablas (presentes en la configuración por defecto) pueden encontrarse en [el manual](#). Si no se indica nada, se trabaja con la tabla `filter`, que cuenta (si no se añaden más) con tres cadenas (no se borran con `-X`): `INPUT`, `FORWARD` y `OUTPUT`. Cada tipo de paquete cae en una tabla y una cadena.

La opción `-P` sirve para establecer una política para una cadena.

Ahora presentamos un script que deniega todo el tráfico.

```
#!/bin/sh

# denegacion.sh
# Denegación de todo el tráfico
```

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

El siguiente script contiene la configuración básica de un sitio web. Denegando todo el tráfico excepto en los puertos correspondientes de los servicios que queremos mantener activos: SSH, HTTP y HTTPS.

```
#!/bin/sh

# basic.sh
# Configuración básica de un sitio web
# Denegar todo excepto SSH, HTTP y HTTPS

# 1: Eliminar todas las reglas y permitir todo el tráfico, configuración limpia
./off.sh

# 2: Denegación implícita
./denegacion.sh

# 3: Permitir conexiones, puesto que somos un servidor web
iptables -A INPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

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

# 5: Abrir 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

# 6: Permitir tráfico por el puerto 80 (HTTP)
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
```

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

La opción `-A` añade nuevas reglas a una cadena. Por ejemplo, los paquetes entrantes (INPUT), transmitidos por protocolo (opción `-p`) TCP con puerto de destino (`--dport`) 22 caerían en la regla debajo de #5. La opción `-m state` indica que el estado debe ser uno de los especificados, y la opción `-j` indica la acción a aplicar a los paquetes que correspondan con la regla (ACCEPT, DROP, REJECT,...) Las opciones `-i` y `-o` indican las interfaces de entrada y salida respectivamente.

```
dcabezas@Lenovo:~$ curl -k https://192.168.56.102
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    Web de dxabezas para SWAP (máquina 1) </br>
  </body>
</html>
dcabezas@Lenovo:~$ curl http://192.168.56.102
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    Web de dxabezas para SWAP (máquina 1) </br>
  </body>
</html>
dcabezas@Lenovo:~$ ping 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
^C
--- 192.168.56.102 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7166ms
```

Figura 4: Con esta configuración, la máquina 1 acepta tráfico SSH (no mostrado), HTTP y HTTPS, pero observamos que responde a un PING.

Sólo tenemos que copiar este script en las distintas máquinas, y ya tenemos nuestra configuración básica.

```
scp * dxabezas@192.168.56.101:/home/dxabezas/.iptables/
```

Ya lo tenemos también en M2, y 192.168.56.103 para M3. Hay que crear el directorio primero y desactivar el cortafuegos, ya que la configuración que hemos hecho no permite que la máquina establezca conexiones por SSH, sólo que las reciba. También se podría añadir la línea

```
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

en el bloque #3, para que las máquinas puedan abrir conexiones.

Como opciones avanzadas, para aceptar también tráfico DNS basta con abrir el puerto 53. La segunda propuesta es más interesante, vamos a configurar M1 y M2 para que sólo acepten peticiones de M3. Para ello, hay que modificar los bloques #6 y #7 y añadir las opciones `-s` (source) en los entrantes y `-d` (destination) en los salientes.

```
# 6: Permitir tráfico por el puerto 80 (HTTP) sólo de M3
iptables -A INPUT -p tcp --dport 80 -s 192.168.56.103 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -d 192.168.56.103 -j ACCEPT
```



```
# 7: Permitir tráfico por el puerto 443 (HTTPS) sólo de M3
iptables -A INPUT -p tcp --dport 443 -s 192.168.56.103 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -d 192.168.56.103 -j ACCEPT
```

Después de copiar el script a M2 y aplicar las reglas, la granja sigue funcionando sin problema. Sin embargo, no se obtiene respuesta al conectarse desde el anfitrión.

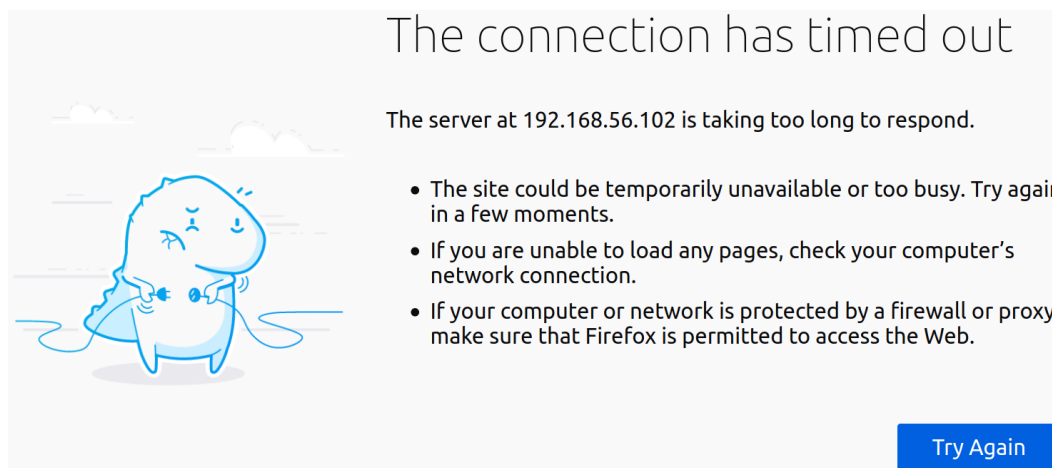


Figura 5: Ya no podemos hacer peticiones directamente a los servidores finales, hay que usar al balanceador como intermediario.

También permitiremos que nuestras máquinas puedan enviar y recibir PINGS, ahora mismo no responden (como se ve en la Figura 4) y tampoco envían (como se ve en la siguiente figura).

```
dxabezas@m1-dxabezas:~/.iptables$ ping 192.168.56.103
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 192.168.56.103 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3059ms
```

Figura 6: No se pueden enviar pings.

Las siguientes reglas habilitan los PINGS.

```
dxabezas@m1-dxabezas:~/.iptables$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
dxabezas@m1-dxabezas:~/.iptables$ sudo iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
dxabezas@m1-dxabezas:~/.iptables$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=0.760 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=0.966 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=1.04 ms
^C
--- 192.168.56.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.760/0.922/1.042/0.124 ms
```

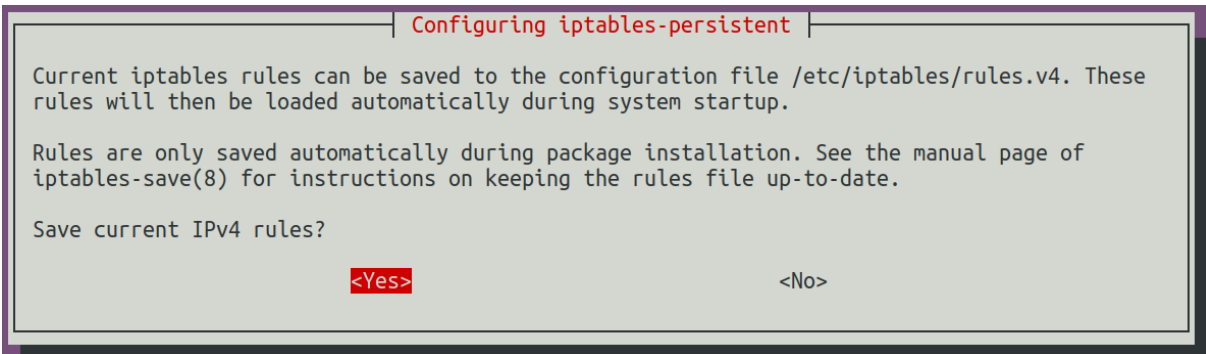
Figura 7: Ya se pueden enviar y recibir PINGS.

Finalmente, mostramos como hacer que la configuración del cortafuegos se ejecute al arranque del sistema. Hay dos

formas de conseguir esto. La primera es añadir la ejecución del script con las reglas al fichero `.bashrc`, de tal forma que esta configuración se aplica al arranque del sistema.

Otra forma de conseguir esto es la herramienta `iptables-persistent`. La instalamos con `sudo apt install iptables-persistent` (no podemos acceder a internet con el cortafuegos activado).

Nos pregunta si queremos salvar la configuración actual como permanente.



Podemos actualizar las reglas mediante los siguientes comandos, que salvan la configuración actual de *iptables* como permanente (tras el arranque). Hay un fichero para IPv4 y otro para IPv6.

```
# iptables-save > /etc/iptables/rules.v4
# ip6tables-save > /etc/iptables/rules.v6
```

(He necesitado hacerlo como root, no podía con sudo).