

# PARTE IV

# SECURICEMOS NUESTRA

# SERVIDORA WEB

---

## Contenidos

- 12. Seguridad de SSH
  - 13. Seguridad del sistema operativo
  - 14. Seguridad de apache
  - 15. Seguridad del router
  - 16. Pentesting
- 

Como sabemos no hay seguridad perfecta y nosotras tampoco somos unas súper expertas. Vamos a intentar a hacer un repaso de cositas que se pueden hacer para aumentar el nivel de seguridad de nuestra servidora.

Entre otras fuentes tomamos como referencia la propia documentación de Raspberry Pi sobre seguridad: <https://www.raspberrypi.com/documentation/computers/configuration.html>. Las invitamos a buscar sus referentes, preguntar y armar documentación propia.

Les proponemos una lista de medidas de seguridad para tomar para salir a Internet de manera segura. Esta lista debemos repasarla cada tanto para mantener nuestra seguridad actualizada. Repasemos los pasos que tomaremos:

### **Seguridad de SSH**

Cambiar el puerto de las conexiones SSH.

Permitir solo autenticación con llaves SSH.

### **Seguridad del sistema operativo**

Cambiar la contraseña por defecto.

Contraseña para sudo.

Aprender a revisar los logs.

Configurar y activar fail2ban.

Configurar y activar *Firewall* UFW.

Mantener el sistema actualizado.

### **Seguridad de Apache**

Revisar permisos de los directorios y archivos.

Revisar permisos y configuración de .htaccess.

Evitar dar información del sistema operativo.

Evitar que se indexen los directorios.

## **Seguridad del router**

Cambiar contraseña de acceso.

Cambiar contraseña del Wi-Fi.

Cambiar nombre de la red Wi-Fi.

Tipo de seguridad para el Wi-Fi.

Firmware actualizado.

Deshabilitar acceso remoto.

Red DMZ.

## ***Pentesting***

Etapas de un pentesting

Escaneo de puertos (con nmap)

Futuros pasos



Imagen de <https://ciberseguras.org/>

## 12. Seguridad SSH

### 12.1 Cambiar el puerto de las conexiones SSH

Una buena práctica de seguridad es cambiar el puerto por defecto para las conexiones SSH (el puerto 22), ya que si intentan atacarnos probablemente prueben a conectarse por ese puerto. De esta manera, cambiando el puerto se reducirá considerablemente el número de intentos de conexión por parte de posibles atacantes. Pueden elegir el número que quieran entre el 1024 y el 65535. Pero primero hay que revisar que no haya otro servicio utilizando ese mismo puerto. Para ello ejecutamos en la terminal:

```
netstat -punta | grep LISTEN
```

Este comando les mostrará los puertos que están a la escucha. Seguramente les mostrará que el puerto 80 (el de Apache) y el 22 (el de SSH hasta ahora) están a la escucha. En el caso de que ya hayan configurado las conexiones por HTTPS, también les saldrá el puerto 443 a la escucha.

Si el comando **netstat** no lo tuvieran instalado en el sistema por defecto, pueden instalarlo con:

```
sudo apt install net-tools
```

Una vez comprobado que el puerto que queremos está libre (nosotras elegimos el 2251), deben modificar el archivo de configuración de SSH:

```
sudo nano /etc/ssh/sshd_config
```

Cambien la línea **#Port 22** por, por ejemplo, **Port 2251** u otro puerto que prefieran y que no esté ocupado. Es importante que borren

la almohadilla/gato/numeral porque si no el sistema lo interpreta como un comentario.

Guarden el fichero, salgan y reinicien el servicio SSH para que se haga efectivo el cambio en la configuración (leer la advertencia antes de ejecutar):

```
sudo service ssh restart
```

**¡Advertencia!** si están conectadas por SSH a la servidora, después del reinicio del servicio les tirará la conexión y tendrán que volver a conectarse por el nuevo puerto. Si hay un error en el cambio de puerto -a nosotras nos pasó ;-), puede que tengan que ir a conectar a la servidora un teclado y un monitor para revisar la configuración. Así que es mejor si están físicamente cerca de la servidora en este paso.

A partir de ahora para conectarse a la Raspberry Pi via SSH deberán especificar el puerto, porque ya no será el puerto por defecto. Así que el comando que deberán escribir de ahora en más será:

```
ssh -p 2251 pi@<dirección ip>
```

Si se se les olvida el puerto que pusieron o no saben si está abierto, pueden hacer un escaneo de puertos desde otra computadora con nmap así (ver más en el apartado de 16.2):

```
sudo nmap -p 1-65535 -T4 -A -v <ip-servidora>
```

Este cambio de puerto afectará a otros servicios que instalaremos más adelante, como el fail2ban. Recordemos el nuevo número del puerto SSH para futuras configuraciones.

## 12.2 Permitir solo autenticación con llaves SSH

Esto ya lo hicimos apenas instalamos nuestro sistema operativo. Pero en caso de que se lo hayan saltado, lo repetimos. Debemos revisar la línea **PasswordAuthentication** en **/etc/ssh/ssh\_config**  
Abrimos el archivo:

```
sudo nano /etc/ssh/sshd_config
```

Comprueben que esté con "no":

```
PasswordAuthentication no
```

Guarden, salgan y reinicien el servicio:

```
sudo service ssh restart
```



Imagen de <https://repository.anarchaserver.org/picture.php?-/157/category/6>

# HACKEANDO EL PATRIARCADO

## 13. Seguridad del sistema operativo

### 13.1 Cambiar la contraseña por defecto

Como ya vimos en el apartado 7 (parte II), es muy importante cambiar la contraseña que trae la usuaria **pi** por defecto, que se puede cambiar con el siguiente comando:

```
sudo passwd pi
```

### 13.2 Contraseña para sudo

Una forma de aumentar la seguridad es configurar el sistema para que cuando ejecuten un comando con sudo les pida la contraseña. Así si alguien entrara infructuosamente no podría ejecutar comandos que modifiquen el sistema sin tener la contraseña de la usuaria **pi**. Para configurar esto hay que ir al archivo **010\_pi-nopasswd**:

```
sudo nano /etc/sudoers.d/010_pi-nopasswd
```

Y editarlo para que quede así:

```
pi ALL=(ALL) PASSWD: ALL
```

Graben y cierren. Esta vez no hay que reiniciar :).

### 13.3 Aprender a revisar los logs

Los logs son registros de acciones llevadas a cabo en el sistema. Una buena práctica es revisar los logs de la servidora para tener una idea de qué ocurre en ella. Todos los logs se encuentran en **/var/logs**. Es verdad que son archivos interminables y difíciles de entender pero pueden tener sus primeros acercamientos y explorar. Ahora

estamos empezando y no habrá mucho. Pero estaría bien adquirir esta práctica para hacerla de manera recurrente.

Existen muchos tipos de logs que contienen registros de distinta clase. Vamos a centrarnos en revisar los logs de autenticación que están en el archivo **/var/log/auth.log**. En ellos se muestra qué dirección IP ha intentado hacer una conexión y si ha logrado establecerla o no. Las animamos a echar un ojo. Las direcciones IP nos permiten saber el proveedor de servicios de Internet (ISP) que las asignó y la localización aproximada de la máquina que está intentando hacer la conexión (si es que no está usando una VPN, proxy o Tor que cambia la IP de origen). Así que una vez que tenemos identificadas las IP de nuestro log podemos intentar geolocalizarlas. ¿Desde dónde están tratando de conectarse a nuestra servidora? Hay distintos sitios web que nos ayudan a geolocalizar direcciones IP. Utiliza alguno de ellos para averiguarlo.

Otros logs importantes para la administración de la servidora y que pueden ser interesantes de revisar son:

**/var/log/apt/history**: donde pueden encontrar el historial de instalaciones y desinstalaciones de paquetes. También está el comando **dpkg -l** que muestra los paquetes instalados pero sin fecha y hora.

**/var/log/syslog**: donde pueden encontrar los logs del sistema. Por ejemplo, si programaron una tarea con **crontab** (ver apartado 13.6) podrán saber si se ejecutó, cuándo y si dio algún error.



Los logs se van comprimiendo en archivos que tienen la extensión **.gz** que no se pueden abrir con el comando **nano** o **cat** sino con el comando **zcat**. También se puede usar **zmore**, para que nos vaya mostrando las primeras líneas, o **zless**, para ver las últimas líneas y no todo el documento. También es importante saber que como los logs son registros irán creciendo a medida que la servidora esté en uso. Si su almacenamiento es pequeño puede llenarse después de un tiempo. Pueden decidir borrarlos en un momento dado o hacer un respaldo en algún almacenamiento externo. De todos modos, al ser archivos de texto no pesan tanto.

### **13.4 Configurar y activar fail2ban**

Fail2ban es un software que ayudará a prevenir intentos de acceso a la servidora "baneando" (rechazando) a las direcciones IP con intentos infructuosos de conexión. Desde el momento en que nuestra servidora salga al mundo recibiremos intentos masivos de conexión. Nosotras revisamos nuestros logs y la mayoría se hacían desde China. Para prevenir esto, antes de lanzarnos a ese mundo desconocido que es Internet, lo más conveniente es instalar y configurar esta herramienta.

Básicamente lo que hace es leer los logs de los diferentes servicios que estemos corriendo y si detecta un número determinado de intentos de conexión fallidos en un tiempo determinado y desde una misma dirección IP (*maxretry*), bloquea esa dirección IP y no le permite seguir intentando conectarse hasta que no haya pasado un determinado tiempo (*bantime*). Todos estos parámetros se pueden configurar.

También se pueden excluir algunas direcciones IP, como la de nuestra casa, por ejemplo, para que no nos bloquee a nosotras mismas en caso de equivocarnos repetidas veces en un intento de conexión. Para instalar Fail2ban ejecutamos el siguiente comando:

```
sudo apt install fail2ban
```

Fail2ban trae un archivo de configuración de ejemplo: **/etc/fail2ban/jail.conf**. Les recomendamos hacer una copia de ese archivo y nombrarlo **jail.local** y guardarlo en el mismo directorio. Pueden hacerlo ejecutando el siguiente comando:

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Sobre ese archivo que han copiado podrán modificar los parámetros de configuración. Para ello debemos abrirlo y editarlo:

```
sudo nano /etc/fail2ban/jail.local
```

En primer lugar hay tres parámetros a configurar:

- *bantime*: el tiempo en segundos por el que una dirección IP va a estar bloqueada (por ejemplo, 3600 es una hora).
- *findtime*: período de tiempo en segundos en el que se tienen que producir los intentos de conexión para que una dirección IP sea bloqueada.
- *maxretry*: número máximo de intentos que una dirección IP puede intentar conectarse.

Estos tres parámetros se pueden configurar por defecto para todos los servicios y luego hacer modificaciones específicas para cada uno

por separado. La configuración específica para cada servicio se llama *jail* (jaula) y se nombra con **[nombre del servicio]**. En este caso, nos parece bien activar las siguientes *jails*: **apache-auth**, **apache-badbots**, **apache-noscript**, **sshd**, **sshd-ddos**. Para hacerlo, en en cada apartado o *jail* correspondiente hay que añadir la línea **enabled = true**. Por ejemplo:

```
[sshd]
enabled = true
port     = <puerto ssh>
logpath  = %(sshd_log)s
backend  = %(sshd_backend)s
```

También es importante que para las *jails* **sshd** y **sshd-ddos** en el parámetro **port** pongan el puerto de SSH que hayan configurado (ver apartado 12.1).

Guardamos, salimos y reiniciamos el servicio:

```
sudo service fail2ban restart
```

Los logs de fail2ban se pueden ver en **/var/log/fail2ban.log** así que para ver qué direcciones IP han sido "baneadas" pueden ejecutar el siguiente comando:

```
sudo cat /var/log/fail2ban.log | grep 'Ban'
```

Lo que hace este comando es abrir el archivo de logs de fail2ban y filtrar las líneas que contengan la cadena de caracteres 'Ban'. Ver en la intro a GNU/Linux el comando **grep**.

Otra manera guay/copada/chilera/chida/calidad/tuanis/bacán de ver qué direcciones IP han sido "baneadas" es:

```
sudo iptables -L -n | awk '$1=="REJECT" && $4!  
="0.0.0.0/0"'
```

Este es un comando más complejo pero básicamente lo que hace es revisar las iptables (el *firewall* base del sistema) y ver qué conexiones con origen Internet (0.0.0.0/0) están siendo bloqueadas (REJECT). También hay otras maneras que se comentan acá: <https://server-fault.com/questions/841183/how-to-show-all-banned-ip-with-fail2ban>

Para ver el estado de fail2ban y las jaulas activadas:

```
sudo fail2ban-client status
```

Para ver el estado de una sola jaula:

```
sudo fail2ban-client status <jaula>
```

Para reiniciar la configuración de una jaula:

```
sudo fail2ban-client reload <jaula>
```

Los nombres de las jaulas (*jails*) los pueden ver en el archivo de configuración.

### **13.5 Configurar y activar Firewall UFW**

Un firewall (o cortafuegos en castellano) es un software que funciona como un muro entre el sistema y el exterior. Con él se permitirá denegar o aceptar las conexiones entrantes (*incoming*) o salientes

(*outgoing*) según direcciones IP y puertos. Ayuda a proteger la servidora de posibles ataques. Una forma sencilla de tener un *firewall* es usar UFW (*Uncomplicated Firewall*). Para instalarlo:

```
sudo apt install ufw
```

Por defecto se instala desactivado así que antes de activarlo lo configuraremos. Empecemos con las políticas por defecto. En este caso proponemos que deniegue cualquier conexión de entrada por cualquier puerto y, luego, ya iremos abriendo los puertos que necesitamos. Así que para establecer la denegación de paquetes de entrada por defecto:

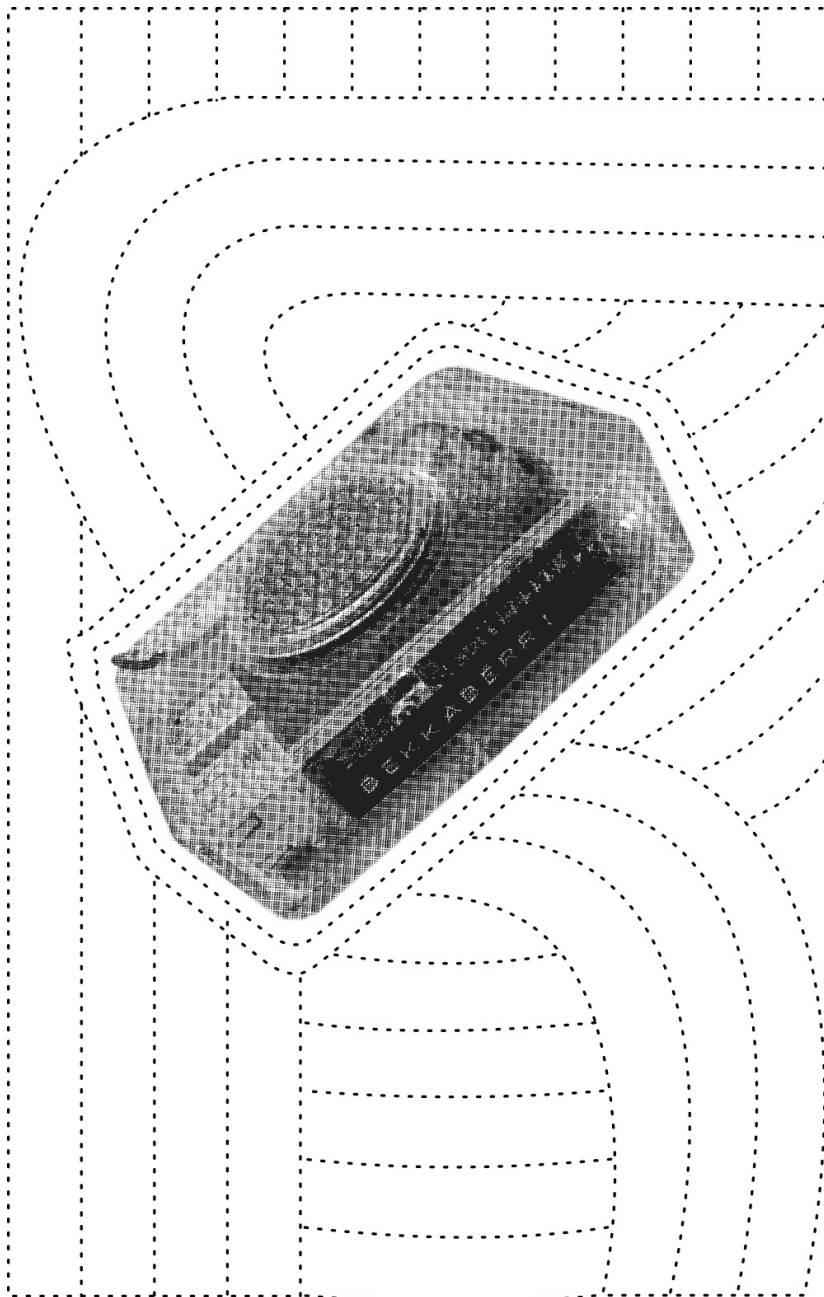
```
sudo ufw default deny incoming
```

Y proponemos que se permita cualquier conexión de salida ya que, como nosotras decidimos adónde salimos, no necesitamos proteger las salidas de paquetes:

```
sudo ufw default allow outgoing
```

Ahora vamos a permitir únicamente las conexiones de entrada que necesitamos. Para el servicio SSH necesitamos abrir el puerto que hayamos configurado (en nuestro caso el 2251); y para el servidor web (Apache), necesitamos abrir el puerto 80 (protocolo HTTP) y el 443 (protocolo HTTPS).

Cualquier puerto tiene dos formas de abrirse, por el protocolo TCP o por el UDP. En este caso solo necesitamos abrir los puertos mencionados por TCP.



Para abrir estos puertos necesitamos ejecutar:

- Para abrir el puerto 80 por TCP: **sudo ufw allow 80/tcp**
- Para abrir el puerto SSH por TCP: **sudo ufw allow <numero del puerto de ssh>/tcp**
- Para abrir el puerto 443 por TCP: **sudo ufw allow 443/tcp**

Si queremos ver las reglas que hemos añadido podemos ejecutar:

**sudo ufw show added**

Deberían tener algo así:

**ufw allow 2251/tcp -- o su puerto de ssh**

**ufw allow 80/tcp**

**ufw allow 443/tcp**

Cuando estén seguras de que tiene las reglas que quieren y necesitan, especialmente la del puerto del SSH, pueden activar el *firewall*. ¡Ojo!, si habilitan el *firewall* y no abren el puerto de la conexión SSH les va a tirar la conexión que tiene abierta. Esto las sacaría de la servidora. Es buena idea seguir estos pasos teniendo acceso físico a la Raspberry Pi. Una vez dicho esto, pueden activarlo con:

**sudo ufw enable**

Les dará una advertencia como la que le explicamos en la nota de arriba: acepten con *yes* (y). Ya tendrán su *firewall* instalado.

Para ver el estado del cortafuegos pueden ejecutar:

**sudo ufw status verbose**

Les debería quedar algo así:

**Status: active**

**Logging: on (low)**

**Default: deny (incoming), allow (outgoing), disabled (routed)**

**New profiles: skip**

To	Action	From
--	-----	----
2251/tcp	ALLOW IN	Any-where
80/tcp	ALLOW IN	Anywhere
443/tcp	ALLOW IN	Anywhere
2251/tcp (v6)	ALLOW IN	Any-where(v6)
80/tcp (v6)	ALLOW IN	Anywhere (v6)
443/tcp (v6)	ALLOW IN	Anywhere (v6)

En el caso de que quieran borrar alguna regla pueden ejecutar:

**sudo ufw delete <la regla que añadieron>**

Para ver las reglas numeradas pueden ejecutar:

**sudo ufw status numbered**

Y borrar por número con:

**sudo ufw delete <numero>**

Estén atentas porque cada vez que añadan o borren reglas los números de las reglas cambiarán.

Si necesitan deshabilitar el *firewall* por unos momentos para hacer alguna comprobación, pueden ejecutar:

**sudo ufw disable**



¡¡No olviden volver a habilitarlo!!

Como ya dijimos:

```
sudo ufw enable
```

### **13.6 Mantener el sistema actualizado**

Una máxima en seguridad digital es mantener nuestros sistemas operativos actualizados. En cada actualización el software se mejora y se cubren los posibles agujeros de seguridad que se pudieran haber encontrado. Por lo tanto, es especialmente importante mantener al día las actualizaciones para que nuestra servidora esté libre de vulnerabilidades. Esta tarea la podemos hacer a mano o automatizarla.

Primero debemos actualizar los repositorios:

```
sudo apt update
```

Y, a continuación, actualizar el sistema:

```
sudo apt upgrade
```

Si lo hacemos manualmente tendremos que ejecutar estos comandos cada cierto tiempo. Pero como es posible que nos olvidemos podemos programar las actualizaciones. Automatizaremos esta tarea utilizando la herramienta **crontab** que ya viene instalada en nuestro Raspbian. Para programar las tareas hay que abrir el archivo de configuración con el siguiente comando:

```
sudo crontab -e
```

Si leemos con cuidado veremos que hay un ejemplo ya escrito, está con un '#' al inicio lo cual significa que es un comentario -también se dice que está "parcheado"- así que no se ejecutará. Pero nos sirve de guía.

Para automatizar la acción tenemos que especificar cada cuánto queremos que se repita la tarea y qué comandos queremos que ejecute. Por ejemplo, para que ejecute un comando todos los días a las 7:00 AM deberíamos escribir:

```
0 7 * * * sudo apt update && sudo apt install  
ssh apache2 -y >> /var/log/apache_ssh_update.txt  
2>&1
```

Este comando actualizaría SSH y apache2 si encuentra actualizaciones disponibles. **-y** significa que instale sin preguntar yes o no, ya que no vamos a estar a las 7 de la mañana para decirle yes. La parte **>> /var/log/apache\_ssh\_update.txt 2>&1** es un poco más compleja: hace que el resultado del comando se escriba en el archivo de logs **apache\_ssh\_update.txt**. Podremos ir a consultarlo para ver si se ha ejecutado correctamente.

Para nosotras SSH y Apache son los dos servicios más importantes a mantener actualizados frente a vulnerabilidades de seguridad. Pero no son los únicos. También tienen que estar actualizados todos los paquetes que están en el repositorio de seguridad de Raspberry Pi OS. La herramienta más conocida para mantener sistemas GNU/Linux (basados en Debian) actualizados es **unattended-upgrades**. Y queremos indicar a continuación unas nociones básicas.

Para instalarlo:

```
sudo apt install unattended-upgrades
```

Para activar la herramienta:

```
sudo dpkg-reconfigure -plow unattended-upgrades
```

Cuando pregunta: Automatically download and install stable updates? Responder 'yes'.

Los archivos de configuración son:

```
/etc/apt/apt.conf.d/50unattended-upgrades
```

```
/etc/apt/apt.conf.d/20auto-upgrades
```

La configuración que viene por defecto nos parece optima: cada día comprueba si hay actualizaciones de seguridad y en tal caso se hacen. Si quieren implementar otra configuración, pueden consular este tutorial con más detalles: <https://diocesanos.es/blogs/equipotic/2021/06/28/unattended-upgrades-como-automatizar-las-actualizaciones-de-nuestro-linux/>.

**Nota:** les recomendamos entrar en la máquina una vez al mes y hacer una actualización manual con **sudo apt update && sudo apt upgrade**. Algunas actualizaciones requerirán un reinicio de la máquina, háganlo cuando lo pida con **sudo reboot**.

Llegará un momento en el que toque actualizar la versión de Raspberry Pi OS, normalmente una vez al año toca. Aquí pueden aprender cómo actualizar el sistema operativo: <https://www.raspberrypi.com/documentation/computers/os.html#updating-and-upgrading-raspberry-pi-os>.

Para pasar de una versión a otra hay que editar el archivo `sources.list`:

**`sudo nano /etc/apt/sources.list`**

Y sustituir el nombre de la versión actual por la siguiente. Por ejemplo, "buster" por "bullseye". Después de guardar y salir del archivo se procede a la actualización con:

**`sudo apt update && sudo apt full-upgrade`**

Y también será necesario reiniciar: **`sudo reboot`**

Para comprobar qué versión tenemos, está el comando:

**`lsb_release -a`**

**Nota:** después una actualización de distribución es posible que muchos paquetes ya no sean necesarios, para ganar espacio podemos hacer: **`sudo apt autoremove`**.

## 14. Seguridad para Apache

### 14.1 Revisar permisos de los directorios y archivos

Los permisos que se dan en GNU/Linux a los directorios y archivos son de lectura, de escritura y de ejecución; y, se asignan de manera independiente a la dueña del archivo, a las usuarias de un grupo o a todas las usuarias. Esos permisos se expresan en un número de tres cifras.

En el apartado 18 explicaremos cómo se leen y gestionan los permisos, pero pueden aprender su lógica en: <https://blog.desdelinux.net/permisos-basicos-en-gnulinux-con-chmod/>.

Para nuestra servidora web los archivos que están en **/var/www/html/** deberán tener permisos **644** y los directorios **755**. Para todos los directorios y archivos en esta ubicación la propietaria debería ser **pi**, el grupo propietario deberá ser **www-data** y **pi** tiene que pertenecer al grupo **www-data**.

Si aún sólo tenemos un **index.html** en el directorio **/var/www/html**, pueden hacer estos cambios más adelante cuando lleguen al apartado 18.

Si están revisando la seguridad y ya tienen contenido en **/var/www/html** les recordamos como poner los permisos adecuados:

```
sudo usermod -a -G www-data pi
sudo chown -R pi:www-data /var/www/html/
cd /var/www/html
```

```
sudo find . -type d -exec chmod 755 {} \;  
sudo find . -type f -exec chmod 644 {} \;
```

Para comprobar los permisos del archivo pueden ejecutar:

```
ls -la /var/www/html
```

## 14.2 Revisar permisos y configuración de `.htaccess`

Si están siguiendo la guía por orden el archivo `.htaccess` aún no existirá. Este es un archivo que sirve para configurar muchas cosas de Apache, por eso hay que protegerlo muy bien.

En el apartado 18 de la parte V explicaremos cómo crearlo, configurar lo básico y protegerlo. Quizás es mejor que sigan adelante y vuelvan aquí al final para revisar.

Para proteger el `.htaccess` lo que hay que recordar es que debe tener permisos **640**. Para cambiarlos ejecutamos:

```
sudo chmod 640 /var/www/html/.htaccess
```

Y nos aseguramos que en el archivo estén las líneas que protejan el acceso. Para hacerlo utilizaremos `cat` que es el comando más sencillo para mostrar los datos de un archivo:

```
sudo cat /var/www/html/.htaccess
```

Deberías ver algo así:

```
# STRONG HTACCESS PROTECTION`
```

```
<Files ~ "^.*\.([Hh][Tt][Aa])">
    Order allow,deny
    Deny from all
    Satisfy all
</Files>
```

El **.htaccess** debería tener los permisos:

```
-rw-r----- 1 pi www-data 235 abr 25 14:04 .htaccess
```

### 14.3 Evitar dar información del sistema operativo

Apache ofrece una serie de datos sobre el sistema operativo de nuestra servidora a través de los **ServerTokens** y la **ServerSignature**. Para restringir esa información debemos abrir el archivo de configuración:

```
sudo nano /etc/apache2/conf-enabled/security.conf
```

Y editar la línea **ServerTokens OS** para que diga **Prod** y que **ServerSignature** cambie de **On** a **Off**.

Guarden, salgan y acuérdense de reiniciar el servicio para que se apliquen los cambios:

```
sudo service apache2 restart
```

### 14.4 Evitar que se indexen los directorios

Para que los directorios alojados en **/var/www/html** no se indexen, es decir, que no se listen en el navegador. Por ejemplo, que si se va a <https://labekka.red/media/> el navegador busqué algún archivo

HTML para mostrar pero si no lo hay listará otros archivos que haya (que es posible que no queramos mostrar). Para evitar esto tenemos que editar el archivo **/etc/apache2/apache2.conf** y borrar la palabra **Indexes**:

```
<Directory /var/www/>  
    Options Indexes FollowSymLinks  
    AllowOverride None  
    Require all granted  
</Directory>
```

Guarden, salgan y acuérdense de reiniciar el servicio para que se apliquen los cambios:

```
sudo service apache2 restart
```

## 15. Seguridad del router

Para acceder a la configuración de un router casero (el típico que instalan las compañías a las que les contratamos el servicio de Internet) normalmente hay que ir a una computadora que esté conectada al router por cable o por Wi-Fi. Escribimos en la barra del navegador la dirección IP local del router que normalmente es **192.168.1.1** o **192.168.0.1**. Se cargará una página donde les pedirá usuario y contraseña de acceso. En ocasiones estos datos están apuntados en el propio router. Si no los encuentran pueden averiguar buscando en Internet la usuario/contraseña por defecto según la marca y modelo del router. Muchas veces es algo como:



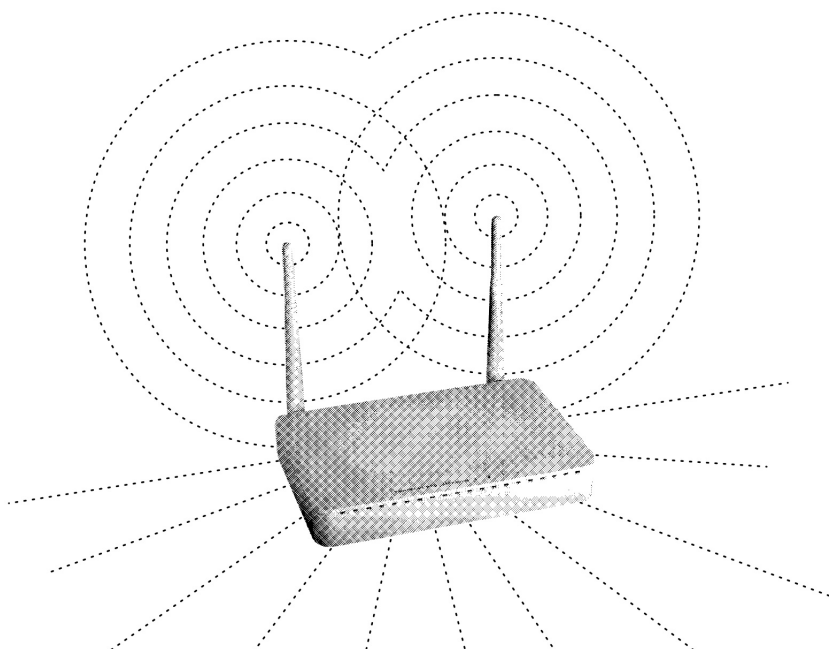
admin/admin

admin/1234

1234/1234 (este funcionó para nuestro ZTE)

user/1234

Si no lo consiguen pueden llamar a su proveedor de servicios de Internet para que se las diga. La persona que está contratando Internet debería tener todo el derecho a administrar su *router* así que no deberían negarles un acceso con permisos de administración. En el apartado 10.1 volvemos a hablar de esto más detalladamente.



## **15.1 Cambiar contraseña de acceso**

Después de conseguir el acceso al *router* con la contraseña que tenga es importante cambiarla y ponerle una segura. Está claro que las contraseñas por defecto no son seguras. Recuerden que una contraseña segura tiene que ser larga (más de 16 caracteres), tener letras, números y símbolos, y no contener información personal.

Cada *router* tiene un panel de administración distinto, depende del fabricante y el *firmware* que haya decidido instalarle. En ocasiones este software no es muy amigable y casi siempre está en inglés. Tengan paciencia y si no se aclaran intenten buscar algún manual del fabricante por Internet. Les invitamos a guiarse por su intuición hasta llegar a la sección en la que se cambia la contraseña de acceso.

## **15.2 Cambiar contraseña del Wi-Fi**

Otra contraseña importante a cambiar es la del Wi-Fi (*wireless* o conexión inalámbrica). Por seguridad no deberíamos tener la contraseña escrita en un papelito a la vista de todas las personas que nos visiten en casa o la colectiva. Algunos *routers* nos permiten configurar redes de Wi-Fi para invitadas con contraseñas específicas.

## **15.3 Cambiar nombre de la red Wi-Fi**

También es importante cambiar el nombre de la red Wi-Fi porque normalmente los nombres que vienen por defecto revelan datos de la compañía proveedora de servicios de Internet. Por ejemplo: Movistar\_2G\_W567. Cuanta menos información demos, mejor.

## **15.4 Tipo de seguridad para el Wi-Fi**

Existen diferentes tipos de seguridad para las redes Wi-Fi. La más insegura es el tipo WEP. Fue de las primeras y utilizarla hoy en día es casi lo mismo que no usar nada. Después está la seguridad WPA (Wi-Fi Protected Access) y sus diferentes versiones WPA1, WPA2 y WPA3. Lo que les recomendamos es que seleccionen WPA3 si el *router* que usan lo permite o, de entre las opciones que tengan, opten por la más segura posible. Pero siempre eviten WEP. Si quieren saber más sobre WPA pueden leer: [https://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access)

## **15.5 Firmware actualizado**

Mantenerlo actualizado ayuda a la seguridad de nuestro dispositivo y nuestra red ya que las vulnerabilidades de seguridad que puedan haberse encontrado se van corrigiendo en las actualizaciones. Es probable que el fabricante no ofrezca actualizaciones para su *firmware* pero, si lo hace, es bueno instalarlas. Utilizar un *router* muy viejito con *firmware* muy viejito puede exponernos a vulnerabilidades. Pero, como siempre, depende de lo que tengamos a mano. Valoren las opciones.

También es útil saber que existe firmware desarrollado por comunidades de software libre que se puede instalar en un *router*. Esto nos evita utilizar el que viene instalado por defecto por el fabricante. Un ejemplo es OpenWRT <https://openwrt.org/>.

## **15.6 Deshabilitar acceso remoto**

Algo súper importante es deshabilitar la configuración remota del *router*. Por lo general esta opción está deshabilitada por defecto. Pero mejor asegurarse de que es así.

Es decir, hay que impedir que desde fuera de la red local (que no esté conectada al *router* por Wi-Fi o por cable) pueda acceder a la página de acceso o *login* ya que si supiera la contraseña podría acceder al panel de administración del *router*.

## **15.7 Red DMZ**

En el apartado 10, en donde explicamos cómo hacer la servidora accesible desde Internet, damos detalles sobre cómo configurar una red DMZ. Si están siguiendo por orden esta guía aun no habrán llegado a ese apartado y pueden seguir avanzando sin preocuparse. Si están dando un repaso a los puntos de seguridad este es el momento en el que revisar que han configurado la red DMZ correctamente.

## **16. Pentesting**

Muchas veces es útil "cambiarse el sombrero" y ponernos en el lugar de la atacante para conocer las fallas o debilidades que puede tener nuestra servidora, para de esta manera poder resolverlas y prevenir futuros ataques. Para ello podemos llevar a cabo un *pentesting* que será básicamente conducir un ataque hacia nuestro sistema que nos permita conocer sus vulnerabilidades y fallas de seguridad.

Es importante notar que parte de las acciones llevadas a cabo en un *pentesting* –como por ejemplo el escaneo de puertos– son ilegales en varios países (no tenemos ahora la info exacta de en cuales) y no debe realizarse sin previo consentimiento de las involucradas. A su vez, debemos tener en cuenta que serán pruebas que impactarán sobre el sistema pudiendo llegar a provocar su caída temporal.

Nota: si están siguiendo la guía en orden y aún la servidora no es accesible desde Internet y tendrán que hacer el *pentesting* desde una computadora que esté conectada al mismo router. Cuando ya sea accesible desde Internet, podrán hacer el escaneo desde cualquier conexión con la dirección IP de la servidora o el dominio (si ya lo configuraron).

### **16.1 Etapas de un *pentesting***

La etapa inicial de un *pentesting* tiene por objetivo recabar información sobre el sistema. Información en sentido amplio: desde quién es la *sysadmin* de la servidora, dónde se encuentran los equipos, qué sistema operativo corre, qué servicios tiene instalados y cuáles son sus versiones, etc. Estos datos se podrán obtener de manera pasiva y/o activa. Pasiva sería rastreando la información disponible y pública en Internet. Por ejemplo, los subdominios de un sitio que se encuentran indexados en la web (subdominio.nuestrodominio.net) o anotando teléfonos de contacto y nombres de personas que integran la colectiva.

Por otro lado, es posible recabar información de un sistema de manera activa, es decir interactuando con el sistema en cuestión, a través de un escaneo de puertos para ver –entre otras cosas– qué servicios expone la servidora a Internet.

## 16.2 Escaneo de puertos (con nmap)

Nmap es una herramienta de código abierto que sirve para hacer escaneos de puertos. Está bueno aprender como usar lo básico y escanear nuestros puertos por si alguno se quedó abierto por error. Nmap tiene una versión de entorno gráfico que se llama Zenmap por si la quieren probar.

Nmap es una herramienta que van a tener que usar desde otra computadora (que no sea la Raspberry), para escanear la servidora desde "fuera". Si usan GNU/Linux puede ser que ya la tengan instalada, si no pueden instalar nmap y Zenmap con:

```
sudo apt install nmap zenmap
```

Si usan otro sistema pueden descargar aquí para windows, MacOS y otros: <https://nmap.org/download.html>

Por defecto nmap escanea los puertos entre 1 y 1024 (más algunos puertos especificados en el archivo de configuración nmap-services). No se escanean todos los puertos posibles (1-65535) dado que cada escaneo a un puerto implica una interacción con la servidora que impacta sobre el sistema pudiendo sobresaturarlo. Teniendo en cuenta esto usamos el tipo de escaneo por defecto de nmap, salvo que el objetivo sea lograr absoluta exhaustividad chequeando los 65535 puertos.

Entonces, de la siguiente manera escaneamos los puertos por defecto de la servidora:

```
sudo nmap <ip-de-la-servidora>
```

Después de hacer el escaneo, que se tardará unos minutos se mostrará algo como lo siguiente:

```
Starting nmap 7.40 ( https://nmap.org ) at 2019-06-08 12:24 CEST
nmap scan report for <ip-de-la-servidora>
Host is up (0.064s latency).
rDNS record for <ip-de-la-servidora>: nues-tra_isp
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
nmap done: 1 IP address (1 host up) scanned in 9.82 seconds
```

Deberían comprobar que solo están abiertos los puertos que ustedes abrieron: 80, 443 y el que hayan elegido para SSH. Si aún no han configurado el certificado SSL/TLS para las conexiones HTTPS (por el puerto 443) aún no les debería salir este puerto abierto.

Para que les salga el puerto de SSH que hayan configurado (si está por encima del 1024), pueden ejecutar lo siguiente con la franja de puertos que quieran escanear (ejemplo 1-3000):

```
sudo nmap -p 1-3000 <ip-de-la-servidora>
```

Es posible complejizar el escaneo agregando dos parámetros a nmap: **-SV** y **-SC**.

El primero le indica a nmap que haga pruebas para comprobar la versión del servicio que está corriendo en cada puerto. Esta informa-

ción es muy útil si se buscan vulnerabilidades (si sabemos por ejemplo que la versión de la servidora web está obsoleta será más fácil encontrar vulnerabilidades documentadas y especificaciones de cómo aprovecharse de sus fallas).



*Aquí en nuestro proceso. Como saben, la seguridad es un proceso y no un producto.*

Por último **-sc** (o **--script default** que es lo mismo) le indica a nmap que extienda el escaneo utilizando sus propios *scripts* rotulados bajo la categoría "por defecto".

¿Qué es esto? nmap es un programa muy poderoso y entre otro montón de cosas al instalarlo incluye una serie de *scripts* o códigos que permiten extender sus funcionalidades. Estos *scripts* se categorizan en:



- **default**, aquellos que amplían la información obtenida sobre una servidora);
- **auth**, aquellos que intentan romper autenticaciones débiles); y,
- **vuln**, útiles para chequear vulnerabilidades de seguridad), entre otros.

El listado de los *scripts* de cada una de las categorías está disponible en: <https://nmap.org/nsedoc/categories/default.html>.

Por último los escaneos de nmap con estas parametrizaciones comenzarán a demorar más tiempo, es por ello que será de utilidad guardar el resultado en un archivo para consultarlo más adelante (con el parámetro **-oA archivo-destino**).

Finalmente corremos un escaneo más avanzado con las opciones indicadas:

```
sudo nmap -sC -sV -oA archivo <ip-de-la-servidora>
```

Para probar otros scripts (no solo los por defecto) puedes seguir este post en español: <https://www.welivesecurity.com/la-es/2015/02/12/auditando-nmap-scripts-escanear-vulnerabilidades/>

### **16.3 Futuros pasos**

Por último las siguientes etapas del pentesting involucrarán ganar y mantener acceso a la servidora, es decir aquellas vulnerabilidades críticas que permitan aprovecharse de ellas para atacar al sistema. Y -en muchos casos- borrar los rastros de que se estuvo allí. Como ya

indicamos es posible investigar si existen vulnerabilidades para las versiones de los servicios que corremos con nmap o también buscando directamente el servicio y su versión en <https://www.exploit-db.com/> o en <https://duckduckgo.com> (o cualquier buscador).

En este punto se vuelve evidente la importancia de mantener todo el software actualizado, dado que versiones obsoletas del mismo abren la puerta a estos tipos de ataques.

Otras herramientas para hacer este tipo de pruebas se listan en la web de nmap: <https://sectools.org/tag/web-scanners/> y <https://sectools.org/tag/vuln-scanners/> ¡Ojo!, no todas son software libre ni gratuitas.

¡Seguimos!