Seguridad de servicios y de red

1. Comprobar los puertos abiertos

a. Los sockets

Las conexiones de red entre dos máquinas se efectúan mediante **sockets.** Un socket es una conexión entre dos puntos empleando una red. Una máquina dispone de una dirección IP y de puertos (virtuales) de conexión numerados, a los cuales se vinculan servicios (ver el capítulo La red). Un cliente establece una conexión desde un puerto de su máquina (número de puerto > 1024, en general elegido de manera aleatoria entre los puertos libres) hacia un puerto determinado de otra máquina, por ejemplo un servidor web en el puerto 80. La comunicación establecida entre los dos pasa por un socket. Se puede configurar el sistema para aceptar o rechazar conexiones desde o hacia determinados puertos locales o distantes; igualmente para las direcciones IP. Es el papel del firewall (cortafuergos) como, por ejemplo, **Netfilter**.

En una instalación Linux típica no se activan los cortafuegos, excepto si dispone de esa opción durante la instalación. No se filtran los puertos y cada máquina exterior puede intentar establecer una conexión a la suya a través de un puerto: eso se denomina «abrir un socket».

No significa necesariamente que haya un agujero de seguridad: si no hay servicio alguno a la escucha, la conexión es imposible. En situación normal, los equipos inician muchos servicios por defecto. Si algunos padecen vulnerabilidades, o se han configurado de manera demasiado laxa, se produce un riesgo real de intrusión.

b. Información desde netstat

El capítulo La red le ha presentado la herramienta **netstat**, que permite obtener información y estadísticas de red sobre una máquina local. En particular se puede comprobar cuáles son los puertos a la escucha en su máquina, quién ha establecido una conexión y qué procesos locales (servicios) están a la escucha:

```
# netstat -apnt -A inet
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad  Dirección local   Dirección remota      Estado      PID/Program name
tcp        0      0  127.0.0.53:53     0.0.0.0:*           ESCUCHAR     499/systemd-resolve
tcp        0      0  0.0.0.0:22        0.0.0.0:*           ESCUCHAR     663/sshd: /usr/sbin
tcp        0      0  127.0.0.1:631     0.0.0.0:*           ESCUCHAR     692/cupsd
tcp        0     64  192.168.1.72:22   192.168.1.42:51485  ESTABLECIDO  15431/sshd: alejand
```

La línea en negrita muestra una conexión SSH establecida entre dos máquinas.

c. La herramienta nmap

Existe un batallón de herramientas de seguridad, de comprobación, de pruebas de fiabilidad, etc. La herramienta **nmap** forma parte de él. Se define como una herramienta de exploración de red y de control de seguridad. Permite probar las conexiones de red de una máquina determinada y obtener mucha información. En particular, analizando tramas, logra determinar el tipo y la versión del sistema operativo remoto.

El manual de instrucciones de nmap tiene más de 2000 líneas, por lo que es imposible estudiarlo todo. A continuación mostramos algunas de sus posibilidades:

Examine los puertos a la escucha en la máquina de prueba que se ha usado durante la redacción de este libro. Varios puertos están abiertos (unos servicios están a la escucha). Algunos de ellos pueden presentar riesgos: telnet, netbios-ssn y microsoft-ds (recursos compartidos de Windows), FTP, vnc, etc.

```
# nmap localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-23 18:23 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Not shown: 994 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
```

```
631/tcp  open  ipp
2049/tcp open  nfs

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

En algunos casos, es posible desactivar los servicios innecesarios:

```
# systemctl stop cups smbd nmbd nfs-server rpcbind rpcbind.socket

# nmap localhost
...
PORT    STATE SERVICE
22/tcp  open  ssh
```

En cuanto se detiene un servicio de red, ya no se puede acceder al puerto.

El parámetro `-A` permite detectar además el sistema operativo remoto y su versión. Para esto, uno o varios puertos deben estar abiertos. Más aún; nmap interroga a cada servicio asociado a los puertos encontrados para recuperar información. Abajo y en negrita resaltamos los valores más relevantes:

```
# nmap -A machine

Not shown: 1676 closed ports
PORT    STATE SERVICE    VERSION
9/tcp   open  discard
13/tcp  open  daytime
21/tcp  open  ftp        vsftpd 2.0.5
22/tcp  open  ssh        OpenSSH 4.3p2 Debian 9 (protocol 2.0)
25/tcp  open  smtp       Postfixsmtpd
37/tcp  open  time       (32 bits)
53/tcp  open  domain
80/tcp  open  http       Apache httpd 2.2.3 ((Debian) PHP/5.2.0-8+etch10)
111/tcp  open  rpcbind     2 (rpc #100000)
113/tcp  open  ident       OpenBSD identd
139/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: SLYNET)
199/tcp  open  smux        Linux SNMP multiplexer
```

```
445/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: SLYNET)
606/tcp  open  mountd      1-2 (rpc #100005)
631/tcp  open  ipp         CUPS 1.2
901/tcp  open  http        Samba SWAT administration server
1389/tcp open  ldap        OpenLDAP 2.2.X
2049/tcp open  nfs         2 (rpc #100003)
3128/tcp open  squid-http
7937/tcp open  nsrexec     1 (rpc #390113)
7938/tcp open  rpcbind     2 (rpc #100000)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.14 - 2.6.17
Uptime: 112.678 days (since Wed Jan 30 21:39:53 2008)
Network Distance: 2 hops
Service Info: Host:  machine.midominio.org; OSs: Unix, Linux, OpenBSD

OS and Service detection performed. Please report any incorrect results
at http://insecure.org/nmap/submit/ .
Nmap finished: 1 IP address (1 host up) scanned in 115.360 seconds
```

Esto quiere decir que una máquina dispone de muy pocos secretos si no está bien protegida. El servidor encontrado tiene como sistema operativo Linux Debian Etch, que dispone de un núcleo comprendido entre 2.6.14 y 2.6.17 (un error, porque el núcleo realmente es un 2.6.18). Los servicios con los que cuenta son:

- OpenSSH 4.3p2
- Apache 2.2.3 y PHP 5.2.0
- Postfix
- Samba 3.x
- Cups 1.2
- OpenLDAP 2.2.X
- Vsftpd 2.0.5

¡Versiones bien antiguas! Basta con consultar los boletines de seguridad para ver si una de estas versiones es conocida por tener problemas de seguridad; si es así, su servidor presenta riesgos. Este es el caso aquí, y se impone una actualización global.

2. Suprimir los servicios inútiles

a. Cuestiones generales

Si alguien "hackea" su sistema, es que hay alguien malintencionado que ha encontrado la manera de entrar. En contra de la idea generalizada, la instalación de un firewall no resuelve todos los problemas: más aún cuando, en una estación de trabajo, la tendencia es abrir varios puertos de redes hacia Internet (o mejor dicho, desde Internet): FTP, HTTP, p2p, ssh y así sucesivamente. Ahora bien, con que un único servicio iniciado tenga un agujero de seguridad es suficiente para que su máquina sea atacada y aumenten los problemas.

Aunque el servicio no sea muy conocido, es posible que la configuración que haya aplicado sea demasiado sencilla o laxa. No sería muy inteligente dejar que su servidor ssh aceptara las conexiones desde el exterior si su contraseña o la del root es joselito, password o algo similar. Durante un ataque, el autor de este libro tuvo la oportunidad de constatar que el atacante intentaba conectarse en bucle, mediante ssh, utilizando una serie de logins/contraseñas predefinidos entre varias combinaciones clásicas preconfiguradas por defecto en determinados programas. Piense, por ejemplo, en herramientas del tipo **fail2ban**. Un caso sencillo es el de una instalación de MySQL por defecto donde la cuenta de administración no tiene contraseña.

A finales de 2016, un ataque masivo a Internet se produjo mediante los agujeros de seguridad de los objetos conectados (IoT - *Internet of Things*): los monitores de bebés, frigoríficos o cámaras web conectadas paralizaron un servicio DNS vital. ¿La causa? Los accesos telnet (texto claro) con las contraseñas por defecto no modifiadas y conocidas por todos.

A principios de 2017, alguien se percató de que más de 33.000 bases de datos noSQL mongoDB estaban expuestas directamente en Internet, sin contraseña de administrador. Bastaba con conectarse directamente para recuperarlo todo, luego destruirlo, y luego pedir un rescate.

Evalúe desactivar todos los servicios que no necesita. Si resulta que necesita algunos en ciertos momentos, y en otros no, no dude en iniciarlos sólo cuando los necesite para pararlos después. Piense también en cambiar sus contraseñas por defecto. Si el servicio permite el uso de un protocolo seguro, actívelo. Asimismo, en su firewall (netfilter u otro),

deje abiertos sólo los puertos estrictamente necesarios.

b. Servicios standalone

Los servicios standalone, es decir, iniciados de manera independiente, están controlados mediante el comando **service**, `/etc/init.d/service` o **systemctl**. Para controlar las paradas y reinicio de estos servicios de manera permanente, utilice los comandos **chkconfig** (distribuciones RPM), **update-rc.d** (Debian), **systemctl enable/disable/mask** (systemd).

c. Servicios xinetd

Se pueden activar y desactivar los servicios controlados por xinetd con la opción **disable** de su archivo de configuración.

```
$ pwd
/etc/xinetd.d
chargen-dgram:    disable      = yes
chargen-stream:   disable      = yes
daytime-dgram:    disable      = yes
daytime-stream:   disable      = yes
discard-dgram:    disable      = yes
discard-stream:   disable      = yes
echo-dgram:       disable      = yes
echo-stream:      disable      = yes
tcpmux-server:    disable      = yes
time-dgram:       disable      = yes
time-stream:      disable      = yes
```

Para actualizar los cambios, fuerce **xinetd** para que vuelva a cargar su configuración.

```
# /etc/init.d/xinetd reload
```

o

```
# systemctl reload xinetd
```

3. Los tcp_wrappers

Las **envolturas TCP** o simplemente **tcp_wrappers** permiten la comprobación de los accesos a un servicio de red determinado (service, xinetd, portmapper). Cada programa que utiliza los **tcp_wrappers** se compila con la librería **libwrap** de manera estática (el comando ldd no permite ver la librería).

Para saber si un servicio de red está compilado con libwrap, se introduce el comando siguiente:

```
strings -f <binario> | grep hosts_access
```

Aquí tenemos un ejemplo con **xinetd**, que utiliza las **tcp_wrappers**:

```
# strings -f /usr/sbin/* |grep hosts_access
/usr/sbin/auditd: hosts_access
/usr/sbin/rpcbind: hosts_access
/usr/sbin/rpc.mountd: hosts_access
/usr/sbin/rpc.rquotad: hosts_access
/usr/sbin/rpc.statd: hosts_access
/usr/sbin/sshd: hosts_access
/usr/sbin/xinetd: hosts_access
```

Si no se devuelve ninguna línea, el programa no utiliza las **tcp_wrappers**.

Entre los servicios que utilizan las **tcp_wrappers**, encontramos:

- **sendmail** (incluyendo postfix);
- **sshd** (ssh);
- **xinetd** (y por lo tanto de manera indirecta todos los servicios asociados);
- **vsftpd** (FTP);

- **portmap** (y por lo tanto nis, NFS);
- **in.telnetd** (telnet), así como la mayoría de los servicios soportados por xinetd.

La comprobación de acceso a un servicio embebido TCP se hace en tres etapas:

- ¿se autoriza el acceso de manera explícita?
- si no es el caso, ¿se prohíbe el acceso de manera explícita?
- si no es el caso, por defecto, se autoriza el acceso.

Los archivos de configuración son `/etc/hosts.allow` y `/etc/hosts.deny` . La sintaxis es común:

```
daemon_list: client_list [:options]
```

- **daemon_list**: lista de los **ejecutables (NO DE LOS SERVICIOS)** separados por comas. Puede poner **ALL** para especificar todos los servicios. Si se dispone de varias interfaces red, se puede usar la sintaxis con @: servicio@ip.

```
in.telnetd: ...
sshd, portmap: ...
sshd@192.168.1.7: ...
```

- **client_list**: clientes autorizados o prohibidos para este servicio. Se puede especificar la dirección IP, el nombre, la máscara de red, el nombre de la red, etc.

```
... : 192.168.1.7, 192.168.1.8
... : 192.168.1.
... : puesto1, puesto2
... : 192.168.1.0/255.255.255.0
... : .midominio.org
```

La lista de clientes admite una sintaxis avanzada:

- **ALL**: correspondencia sistemática.

- **LOCAL**: todos los anfitriones cuyo nombre no contiene punto (puesto1, puesto2, etc.).
- **UNKNOWN**: anfitrión cuyo nombre no se puede resolver.
- **KNOWN**: anfitrión cuyo nombre se puede resolver.
- **PARANOID**: anfitrión cuyo nombre no se puede resolver o cuyo IP no tiene resolución inversa.
- **EXCEPT**: permite excluir ciertos anfitriones.

```
ALL EXCEPT puesto10
```

Para verificar una regla, el sistema lee primero **`/etc/hosts.allow`** , luego **`/etc/hosts.deny`** . La búsqueda se detiene en la primera correspondencia encontrada. Una línea en hosts.allow autoriza la conexión. Una línea en hosts.deny prohíbe la conexión. Si no se deniega de manera explícita el acceso, se autoriza: la petición no corresponde a ningún criterio.

En el ejemplo siguiente:

- Sólo los miembros de la subred 192.168.1.0 tienen permiso para conectarse al servidor FTP (prohibido para todos los demás).
- Los anfitriones puesto 1 y puesto 2 tienen acceso a telnet y portmap.
- Los anfitriones de baddominio.org, excepto trusted, no tienen conexión alguna posible.
- Se prohíbe el servicio pop/imap a todos los de la red 192.168.0.0, salvo 192.168.1.5.

```
# /etc/hosts.allow
vsftpd:         192.168.1.
in.telnetd, portmap:    puesto1, puesto2

# /etc/hosts.deny
ALL: .baddominio.org except trusted.baddominio.org
vsftpd,in.telnetd,portmap: ALL
dovecot : 192.168.0. EXCEPT 192.168.1.5
```

4. Netfilter

a. Presentación

Netfilter es una arquitectura de filtro de los paquetes para los núcleos de Linux a partir de la versión 2.4. El filtrado se hace en el mismo núcleo en las capas 2, 3 y 4 del modelo OSI, es decir, los vínculos dados, red y transporte. Por ejemplo, es capaz de actuar a bajo nivel en las interfaces ethernet (2), en la pila IP (3) y en los protocolos de transporte como TCP (4). El filtrado no tiene estado: como Netfilter sólo inspecciona los encabezamientos de los paquetes, es muy veloz y no conlleva tiempo de espera, o muy poco.

Se pueden inspeccionar los contenidos de los paquetes (protocolos aplicativos) usando extensiones, pero este trabajo se delega a herramientas de usuario.

Dicho de otro modo, netfilter es un firewall que actúa a nivel del protocolo.

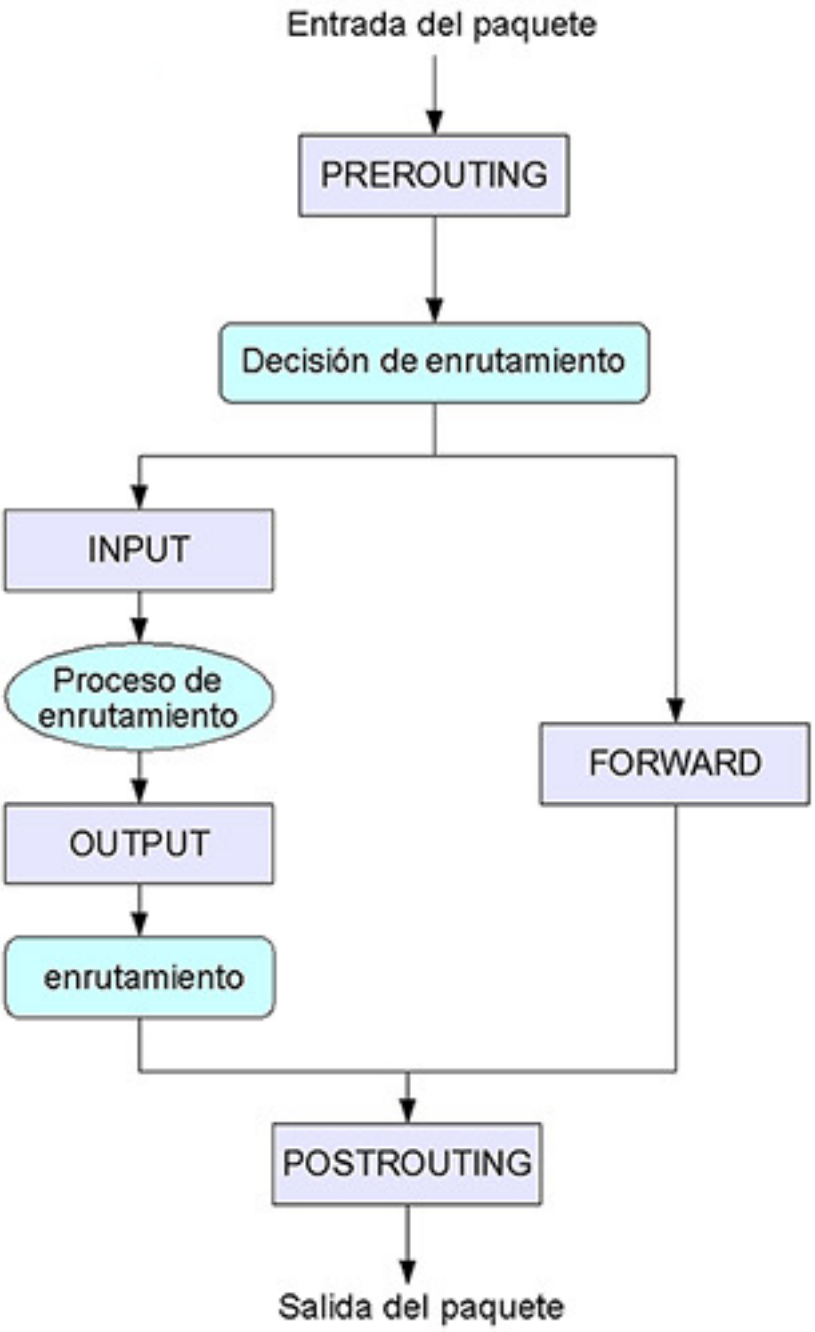
El programa usuario que permite actuar sobre las reglas de filtrado es **iptables**.

Se realiza la implementación a nivel del núcleo con módulos.

El sustituto de netfilter apareció con el núcleo 3.13 y se llama **nftables**. Es reciente pero ha empezado a usarse en distribuciones como RHEL9 o CentOS 8. Diríjase a la documentación de su distribución para obtener los comandos.

b. Vida de un paquete

Mejor un esquema que un largo discurso. El paquete llega por arriba y sale por abajo. Entre ambos puntos, pasa por diferentes niveles de netfilter.



Las etapas de la vida de un paquete de red con netfilter

Cada estado (rectángulo) corresponde a un punto de filtrado posible mediante el comando **iptables**.

- **PREROUTING**: trata los paquetes a su llegada. Si un paquete tiene como destino el sistema local, se le tratará con un proceso local (INPUT, OUTPUT). En caso contrario, y si está activado el forwarding, se aplicarán las reglas FORWARD y

POST_ROUTING.

- **FORWARD**: los paquetes sólo cruzan el sistema local. Trata los paquetes enrutados a través del sistema local.
- **INPUT**: trata los paquetes destinados al sistema local, en entrada (después del enrutamiento).
- **OUTPUT**: trata los paquetes que dejan el sistema local, antes POSTROUTING.
- **POSTROUTING**: trata los paquetes justo antes de su salida del sistema.

c. Principio de las reglas

Cuando netfilter trata un paquete, lo relaciona con un determinado número de reglas que determinan lo que se debe hacer con él.

- Se ordenan las reglas: la posición de una regla en una lista indica si se utilizará la regla y cuándo se hará.
- Se prueban los paquetes con cada una de las reglas, una tras otra.
- Netfilter funciona según el mecanismo de la primera correspondencia. Si una regla corresponde, se desatienden las otras y se detiene la comprobación.
- Una regla puede especificar varios criterios.
- Para que una regla corresponda a un paquete, todos los criterios deben corresponder.
- Si a pesar de todas las reglas el paquete pasa, se puede aplicar una regla por defecto.

d. Destinos de reglas

Un destino de regla determina qué acción se debe emprender cuando un paquete corresponde a los criterios de una regla. Se utiliza la opción `-j` de **iptables** para especificar el destino.

Los dos destinos básicos son DROP y ACCEPT. netfilter cuenta con extensiones que añaden otros destinos, como LOG o REJECT.

- **DROP**: se rechaza el paquete. No se manda ninguna notificación a la fuente.

- **REJECT**: se rechaza el paquete. Notifica un error a la fuente.
- **ACCEPT**: se acepta el paquete.
- **LOG**: se manda una información de seguimiento al syslog.

Puede crear reglas sin destino. En este caso, la regla incrementará el recuento de un contador de paquetes y un contador de bytes asociados a la regla con objetivos estadísticos.

e. Primer ejemplo

A continuación se presenta una regla sencilla que prohíbe los paquetes procedentes de 192.168.1.11.

```
# iptables -A INPUT -s 192.168.1.11 -j DROP
```

- **-A**: punto de filtrado (INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING), llamado también **cadena**.
- **-s**: fuente, puede ser una dirección IP, un nombre de anfitrión, una red, etc.
- **-j**: jump, objetivo o destino de la regla (ACCEPT, DROP, REJECT...)

Resultado: prohibirá la entrada de los paquetes cuya fuente es 192.168.1.11.

f. Operaciones básicas

Se numeran las reglas a partir de 1.

➔ Añada una regla al final de la cadena con **-A**.

```
# iptables -A INPUT -s 192.168.1.2 -j DROP
```

➔ Inserte una regla con **-I** en la posición deseada.

```
# iptables -I OUTPUT 3 -d 192.168.1.25 -j DROP # insertar en la 3ª posición
```

Suprima una regla con **-D**.

```
# iptables -D INPUT 1 # suprima la regla 1
```

Liste las reglas con **-L**.

```
# iptables -L OUTPUT
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Utilice **-F** (flush) para suprimir el conjunto de las reglas.

```
# iptables -F
```

Utilice **-P** (policy) para modificar las reglas por defecto de una cadena.

```
# iptables -P INPUT DROP
# iptables -L INPUT
Chain INPUT (policy DROP)
target     prot opt source               destination
```

Tenga cuidado al agregar cadenas a cadenas existentes. Cuando la política predeterminada en `INPUT` es del tipo `ACCEPT`, la última regla es a menudo `-J REJECT` para prohibir todo excepto las reglas anteriores. Con `-A INPUT`, la nueva regla se ubicará después de ella, y por lo tanto no tendrá efecto.

g. Criterios de correspondencia

General

Los criterios de correspondencia determinan la validez de una regla. Se deben comprobar todos los criterios para bloquear un paquete. Los criterios básicos son:

- **-i**: interfaz entrante (filtrado de capa 2);
- **-o**: interfaz saliente (filtrado de capa 2);
- **-p**: protocolo de capa 4. Para los nombres, ver el archivo **`/etc/protocols`** ;
- **-s**: dirección IP de la fuente (o red);
- **-d**: dirección IP de destino (o red).

→ Prohibir las entradas por eth0.

```
iptables -A INPUT -i eth0 -j DROP
```

→ Prohibir el forward entre eth1 y eth2.

```
iptables -A FORWARD -i eth1 -o eth0 -j DROP
```

→ Prohibir el protocolo ICMP en entrada (¡el ping!).

```
iptables -A INPUT -p icmp -j DROP
```

TCP, UDP e ICMP

Siguiendo el protocolo (capa 4), algunas opciones son posibles. Es el caso de tcp, udp o icmp (en particular utilizado por ping). Se realiza en general el filtrado a nivel de protocolo con extensiones a netfilter.

- **-p**: protocolo (tcp, udp, icmp, etc.)
- **--sport**: puerto fuente
- **--dport**: puerto destino

Si desea prohibir, por ejemplo, las conexiones entrantes con destino al puerto 80 (servidor httpd), proceda como a continuación:

```
iptables -A INPUT -p tcp --dport 80 -j DROP
```

Argumentos de los criterios

Para las direcciones, puede especificar:

- un anfitrión por su nombre o su dirección IP;
- una red por su nombre o su máscara (192.168.1.0/24, 192.168.1.0/255.255.255.0).

Para los puertos:

- un número;
- un nombre (ver/etc/servicios);
- una gama de puertos: **123:1024**.

En todos estos casos puede usarse el signo de exclamación para definir excepciones.

Para prohibir en la entrada todas las conexiones salvo las de 10.0.0.1:

```
# iptables -A INPUT -s ! 10.0.0.1 -j DROP
```

Usuarios y grupos

Netfilter gestiona los UID y GID de los procesos que emiten paquetes desde el host. Puede realizar un filtrado en la salida para los usuarios y grupos locales, para las reglas OUTPUT y POSTROUTING. El módulo es **owner**. Los usuarios se especifican con **--uid-owner** y los grupos con **-gid-owner**.

Esto puede ser muy útil, por ejemplo, en el caso de un proxy transparente. Se puede redirigir cualquier conexión saliente con destino el puerto 80 hacia el puerto 3128 para todo usuario excepto el proxy.

Para ello hay que ejecutar el siguiente comando:

```
# iptables -t nat -A OUTPUT -m owner ! --uid-owner proxy -p tcp
-m tcp --dport 80 -j REDIRECT --to-ports 3128
```

Para permitir a un usuario pasar por otra regla, puede ejecutar el siguiente comando ANTES (recuerde que el orden es importante) de la redirección:

```
# iptables -t nat -A OUTPUT -p tcp --dport 80 -m owner --uid-owner seb -j
ACCEPT
```

h. Tablas

Además de los puntos de filtrado (INPUT, OUTPUT, etc.) Netfilter define tablas particulares que pueden contener reglas específicas. Puede crear sus propias tablas que contengan sus propias reglas y a continuación, adjuntarlas a un punto de filtrado. Pero Netfilter también dispone de tablas por defecto y especialmente la tabla **nat**, que contiene las reglas que permiten modificar las IP y puertos origen y destino de los paquetes. Se muestra del siguiente modo:

```
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
```

```
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere
tcp dpt:www owner UID match root
ACCEPT     tcp  --  anywhere             anywhere
tcp dpt:www owner UID match seb
ACCEPT     tcp  --  anywhere             anywhere
tcp dpt:www owner UID match esteban
REDIRECT   tcp  --  anywhere             anywhere
! owner UID match proxy tcp dpt:www redir ports 3128
...
```

La siguiente regla modifica dinámicamente la IP de destino de un cierto tipo de paquetes. Los paquetes con destino el puerto 80 de la IP 192.168.1.2, se modificarán de tal modo que se reenviarán a la 192.168.10.10:

```
# iptables -t nat -A OUTPUT --dst 192.168.1.2 -p tcp --dport 80 -j
DNAT --to-destination 192.168.10.10
```

i. Guardar las configuraciones

Las reglas definidas con **iptables** se cargan en memoria y se transmiten de manera dinámica al núcleo. Pero al arrancar de nuevo la máquina, se pierden. **Red Hat** permite guardar el conjunto de las reglas de manera que se vuelvan persistentes.

```
# service iptables save
```

Las reglas están guardadas en el archivo `/etc/sysconfig/iptables` .

Los comandos siguientes hacen lo mismo:

```
# iptables-save > /etc/sysconfig/iptables
# ip6tables-save > /etc/sysconfig/ip6tables
```

En Debian y Ubuntu, se pueden utilizar las herramientas **iptables-save** e **iptables-restore**, guardando las reglas por ejemplo en `/etc/iptables.rules` . Puede entonces emplear las líneas siguientes en `/etc/network/interfaces` :

```
pre-up iptables-restore < /etc/iptables.rules
post-down iptables-restore < /etc/iptables.downrules
```

Para las demás distribuciones, consulte la documentación. En algunos casos, puede ser útil crear su propio script.

Se deberían cargar las reglas **iptables ANTES** de la activación de la red. Así, no hay ningún riesgo de seguridad, porque las reglas serán directamente válidas en el momento en que se active la red.

5. UFW

Partiendo de la premisa que iptables es un poco complicado para el usuario medio, los desarrolladores de la distribución Ubuntu han decidido crear una herramienta en línea de comandos que permita configurar de forma más sencilla Netfilter. UFW (*Uncomplicated Firewall*) es una capa frontal de Netfilter e iptables. No los reemplaza, sino que permite crear en su lugar reglas Netfilter. Utiliza y es un complemento de iptables. Sólo hay que usar un comando: **ufw**.

UFW está disponible para Ubuntu y también puede instalarse en Debian. Si no está disponible por defecto en otras distribuciones, se puede recompilar, las fuentes están disponibles. Se están dedicando esfuerzos para crear paquetes, especialmente para Fedora. Los comandos siguientes se deben ejecutar directamente como root o desde sudo.

a. Activación y estado

Active ufw con el parámetro **enable**, como se muestra a continuación:

```
# ufw enable
El cortafuegos está activo y habilitado en el arranque del sistema
```

En Ubuntu, ufw se inicia mediante init o más recientemente con systemd. El script de init comprobará el estado de activación de ufw para determinar si debe cargar las reglas o no hacerlo. De este modo, el script init siempre debe estar activo, pero es esta llamada al comando ufw la que determina si debe funcionar o no.

El estado actual se obtiene con el parámetro **status**. En el ejemplo siguiente ya hay una regla presente (la de acceso a través del puerto ssh):

```
# ufw status
Estado: activo

Hasta                      Acción     Desde
-----                      -------    -------
22                         ALLOW      Anywhere
```

Se puede añadir el parámetro **verbose** para obtener más detalles, especialmente para mostrar el estado por defecto de las distintas tablas:

```
# ufw status verbose
Estado: activo
Acceso: on (low)
Por defecto: deny (Entrada), allow (Salida)
```

ufw se desactiva con disable. Las reglas creadas con ufw se descargarán. Sin embargo, las reglas creadas por iptables no se modificarán.

```
# ufw disable
El cortafuegos está detenido y desactivado en el arranque del sistema
```

b. Reglas por defecto

Con ufw, siendo voluntariamente sencillo, sólo el tráfico entrante y saliente tiene reglas por defecto: allow o deny para los permisos, como en los ejemplos siguientes:

```
# ufw default deny incoming
La política incoming predeterminada cambió a «deny»
(asegúrese de actualizar sus reglas consecuentemente)
# ufw default allow outgoing
La política outgoing predeterminada cambió a «allow»
(asegúrese de actualizar sus reglas consecuentemente)
```

El registro se activa y se desactiva así:

```
# ufw logging off
Registro desactivado
# ufw logging on
Registro activado
```

c. Gestión de reglas

Reglas simples

La forma más sencilla consiste en permitir o prohibir conexiones entrantes o salientes en función de los puertos. Por ejemplo, puede permitir el acceso al servicio ssh de su servidor:

```
# ufw allow 22
Regla añadida
```

Mejor que el número del puerto, puede utilizar el nombre del servicio tal y como se define en /etc/services.

Para prohibir el acceso con deny:

```
# ufw deny 22
Regla actualizada
```

Eliminación

Hay veces en que no nos es suficiente, ya que este comando no elimina la regla antigua, la actualiza. Para eliminar definitivamente una regla, muéstrelas todas de forma numérica, como se muestra a continuación:

```
# ufw satus numbered
Estado: activo

     Hasta                  Acción      Desde
     -----                  ------      -----
[1] Anywhere               ALLOW IN    212.27.38.253/udp
[2] 22                     DENY IN     Anywhere
```

Elimine la regla 2:

```
# ufw delete 2
Borrando:
 deny 22
¿Continuar con la operación (s|n)? s
Regla eliminada
```

También puede borrar la regla explícitamente:

```
# ufw delete deny 22
Regla eliminada
```

Aplicaciones

Debido a que ufw tiene que ser sencillo a ojos del usuario, las aplicaciones instaladas pueden ofrecer reglas predefinidas para simplificar la integración de reglas asociadas. Por ejemplo, el uso de un servidor Samba requiere la apertura de algunos puertos. En la instalación del paquete asociado, los scripts de postinstalación llamarán a ufw para definir un perfil que podrá activar a continuación. La lista de aplicaciones para las que hay reglas se muestra con **app list**:

```
# ufw app list
Aplicaciones disponibles:
  CUPS
  OpenSSH
  Postfix
  Postfix Submission
  Samba
  Squid
```

Los detalles de una aplicación se muestran con app info. Por ejemplo, consultamos los detalles de Samba:

```
# ufw app info Samba
Perfil: Samba
Título: LanManager-like file and printer server for Unix
Descripción: The Samba software suite is a collection of programs that
implements the SMB/CIFS protocol for unix systems, allowing you to serve
files and printers to Windows, NT, OS/2 and DOS clients. This protocol is
sometimes also referred to as the LanManager or NetBIOS protocol.

Puertos:
  137,138/udp
  139,445/tcp
```

Las reglas se activan o desactivan fácilmente, tal y como se muestra a continuación:

```
# ufw allow Samba
Regla añadida
# ufw deny Samba
Regla actualizada
```

Reglas más complejas

Para cada puerto, puede especificar el protocolo:

```
# ufw allow 22/tcp
```

También puede utilizar una sintaxis más extendida, como la que prohíbe todo el tráfico entrante en TCP desde la subred 192.168.1.0/24 hacia cualquier IP de su servidor el puerto 22:

```
# ufw deny proto tcp from 192.168.1.0/24 to any port 22
```

La regla siguiente permitirá el acceso a los puertos 80, 443 y del 8080 al 8090 en su ordenador:

```
# ufw allow proto tcp from any to any port 80,443,8080:8090
```

Para prohibir cualquier conexión desde su equipo hacia cualquier equipo al puerto 80:

```
# ufw reject out to any port 80
```

d. Limitaciones

UFW es práctico para casos relativamente simples: servidores básicos, puestos de trabajo, ordenadores personales. Desafortunadamente, para un uso avanzado, UFW es insuficiente. Por ejemplo, los protocolos gestionados están limitados a TCP y UDP. Si desea autorizar multicast para protocolos como VRRP u OSPF, siga buscando. En ese caso deberá utilizar netfilter y los comandos vistos previamente.

6. firewalld

Desde que Ubuntu apareció con sus comandos simplificados, Red Hat siguió la misma estela con firewalld. Este es también un demonio que se pone en lugar de netfilter y que creará las reglas en su lugar. Sin embargo, donde UFW es (demasiado) simple, `firewalld` permite hacer cosas muy complejas. Nos limitaremos aquí a los casos simples y más corrientes.

a. Activación

Firewalld arranca mediante systemd:

```
# systemctl status firewalld
  firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled;
vendor preset: enabled)
   Active: active (running) since sab. 2017-02-04 10:16:23 CET; 3s ago
     Docs: man:firewalld(1)
Main PID: 2974 (firewalld)
   CGroup: /system.slice/firewalld.service
           2974 /usr/bin/python -Es /usr/sbin/firewalld --nofork -nopid
```

El comando central es `firewall-cmd`. Puede obtener el estado actual como sigue:

```
# firewall-cmd --state
running
```

b. Zones

Las zonas son un concepto básico de firewalld: un conjunto de reglas que describen lo que debe estar autorizado y lo que no según el nivel de confianza que proporcione a las diferents redes a las que su equipo se conecte. Sus interfaces de red se asignan a esas zonas.

Por defecto, en un ordenador personal, firewalld ofrece una zona llamada public, que se aplica a su interfaz de red principal.

```
# firewall-cmd --get-default-zone
public
```

Para conocer las reglas aplicadas en una zona:

```
# firewall-cmd --list-all
```

```
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  sourceports:
  icmp-blocks:
  rich rules:
```

Existen otras varias zonas definidas por defecto. La zona home puede utilizarse en un equipo personal. Vemos aquí abajo que los servicios autorizados son los clientes DHCP (para obtener una dirección IP dinámica), samba (para acceder a los recursos compartidos de Windows) y el servicio SSH, para poder conectarse:

```
# firewall-cmd --get-zones
FedoraServer FedoraWorkstation block dmz drop external home dmz internal libvirt
# firewall-cmd --zone=home --list-all
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  sourceports:
  icmp-blocks:
  rich rules:
```

Las zonas están vinculadas a las interfaces de red; si queremos asignar las interfaces

eth0 y eth1 del equipo de prueba a la zona home, estos son los comandos que deberemos ejecutar:

```
# firewall-cmd --zone=home --change-interface=eth1
success
# firewall-cmd --zone=home --change-interface=eth0
success
```

Para mostrar la asignación de las interfaces de red a las diferentes zonas, ejecutamos el comando:

```
# firewall-cmd --get-active-zones
home
 interfaces: eth0 eth1
```

La zona por defecto se modifica como sigue:

```
firewall-cmd --set-default-zone=home
```

Para hacer permanentes los cambios en sus interfaces, los archivos **/etc/sysconfig/network-scripts /ifcfg-*** deben modificarse:

```
ZONE=home
```

c. Servicios

Al igual que con UFW, firewalld dispone de perfiles que permiten aplicar reglas de filtrado por defecto. Estos perfiles dependen de los paquetes instalados. Podemos obtener la lista de ellos con estos comandos:

```
# firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client amqp amqps apcupsd audit bacula
bacula-client bb bgp bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc
```

```
bittorrent-lsd ceph ceph-mon cfengine cockpit condor-collector ctdb dhcp dhcpv6
dhcpv6-client distcc dns dns-over-tls docker-registry docker-swarm
dropbox-lansync elasticsearch etcd-client etcd-server finger freeipa-4
freeipa-ldap freeipa-ldaps freeipa-replication freeipa-trust ftp ganglia-client
ganglia-master git grafana gre high-availability http https imap imaps ipp
ipp-client ipsec irc ircs iscsi-target isns jenkins kadmin kdeconnect kerberos
kibana klogin kpasswd kprop kshell kube-apiserver ldap ldaps libvirt
libvirt-tls lightning-network llmnr managesieve matrix mdns memcache minidlna
mongodb mosh mountd mqtt mqtt-tls ms-wbt mssql murmur mysql nfs nfs3 nmea-0183
...
# firewall-cmd --get-services | tr -s " " "\n" | wc -l
168
```

Estos servicios se describen en **/usr/lib/firewalld/services/*.xml**. Por ejemplo, para SSH es relativamente simple, porque solo describe el puerto 22. Si usted activó este servicio, firewalld abrirá el puerto 22 en las interfaces de la zona activa:

```
# cat /usr/lib/firewalld/services/ssh.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>SSH</short>
  <description>Secure Shell (SSH) is a protocol for logging into and
executing commands on remote machines. It provides secure encrypted
communications. If you plan on accessing your machine remotely via SSH
over a firewalled interface, enable this option. You need the openssh-
server package installed for this option to be useful.</description>
  <port protocol="tcp" port="22"/>
</service>
```

Agregue el servicio HTTP a la zona home:

```
# firewall-cmd --zone=home --add-service=http
# firewall-cmd --zone=home --list-all | grep services
  services: dhcpv6-client http mdns samba-client ssh
```

Este cambio no es permanente y se perderá al reiniciar. Para hacerlo permanente, utilice el

parámetro idóneo:

```
# firewall-cmd --zone=home --add-service=http --permanent
success
```

Un servicio puede eliminarse con `--remove-service` siguiendo la misma sintaxis.

d. Reglas personalizadas

Además de los servicios, se pueden proporcionar otras reglas, como los puertos:

```
# firewall-cmd --zone=home --permanent -add-port=443/tcp
success
# firewall-cmd --zone=home --permanent --list-ports
443/tcp
```

El parámetro `--remove-port` elimina el puerto siguiendo la misma sintaxis.

e. Reglas ricas

Hasta el momento, aparte de las zonas que son, de hecho, los perfiles de las reglas, firewalld se aproxima un poco a UFW. Pero UFW no permite, a través de su comando, implementar reglas complejas. Firewalld ofrece las Rich Rules, que permiten definir reglas iptables:

```
# firewall-cmd --zone=home --add-rich-rule 'rule family="ipv4" source
address=192.168.0.14 accept'
success
# firewall-cmd --zone=home --list-all
home (active)
...
  rich rules:
    rule family="ipv4" source address="192.168.0.14" accept
```

Se hace posible, por ejemplo, crear reglas para VRRP:

```
# firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 \
--in-interface eth0 --destination 224.0.0.18 --protocol vrrp -j ACCEPT
success
```

Luego vuelva a cargar las reglas:

```
# firewall-cmd --reload
```

7. GPG

a. Un clon de PGP

GPG (*Gnu Privacy Guard*) es un clon libre de **PGP** (*Pretty Good Privacy*). Implementa el algoritmo de cifrado RSA. PGP es objeto de una norma que GPG respeta. Esto significa que las dos implementaciones son compatibles: las claves generadas por uno u otro se pueden intercambiar.

El objetivo de GPG es cifrar una comunicación gracias a un cifrado por claves asimétricas. Una clave permite firmar el texto, otra clave sirve para encriptar el texto.

El cifrado por claves asimétricas utiliza dos claves, una pública y otra privada:

- Su **clave pública**, difundida públicamente, permite al destinatario descifrar el mensaje que ha firmado (con la clave privada).
- Su **clave privada** le permite firmar un mensaje. La persona que lo recibe comprueba su firma con la ayuda de la clave pública que le ha proporcionado, demostrando así que usted es realmente el autor del mensaje.

A continuación, vamos a presentar todo lo necesario para crear y gestionar un almacén de claves: sus claves públicas y privadas, las claves públicas de sus amigos, las firmas, etc.

Se presentan los comandos en modo consola. Existen herramientas gráficas, como **kgpg**, para gestionar las claves. Asimismo, numerosos clientes de mensajería integran GPG (Kmail lo integra por defecto). Gracias al plugin Enigmail se puede incorporar facilmente

GPG a Thunderbird. Se utilizan también las firmas para comprobar el origen de los paquetes RPM, por ejemplo.

b. Generar claves

Utilice `gpg` con el parámetro `--gen-key`. GPG empieza creando, si no existen ya, el directorio raíz de GPG propio de cada usuario y los archivos que contendrán los elementos gestionados por gpg.

```
$ gpg --gen-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: anillo «/home/jolivares/.gnupg/secring.gpg» creado
gpg: anillo «/home/jolivares/.gnupg/pubring.gpg» creado
```

En las versiones recientes. GPG ya no propone esta elección. Para forzar este comportamiento use el parámetro`--full-generate-key`. Seleccione el tipo de clave deseada. La primera elección incluye las otras dos; por lo tanto, es la que debería efectuarse:

```
Seleccione el tipo de clave deseado:
  (1) RSA y RSA (por defecto)
  (2) DSA y ElGamal (por defecto)
  (3) DSA (sólo firmar)
  (4) RSA (sólo firmar)
¿Su elección?
```

Luego debe elegir el tamaño, en bits, de la clave que hay que generar. Cuánto más larga sea la clave, más complejo es el cifrado. Pero en máquinas más antiguas, el descifrado puede tardar entonces más tiempo. Puede elegir una clave de 2048 bits por ejemplo:

```
las claves RSA pueden tener entre 1024 y 4096 bits de longitud.
¿De qué tamaño quiere la clave? (2048)
```

```
El tamaño requerido es de 2048 bits
```

Luego indique la duración de validez de las claves. Por defecto, si prevé utilizarla el mayor tiempo posible, elija cero para una duración infinita. Si no, precise la duración deseada según el formato indicado. Valide su elección. Debe saber que es posible invalidar una clave generada por error con una clave de revocación de la que hablaremos un poco más adelante.

```
Especifique cuánto tiempo debería ser válida esta clave.
         0 = la clave no expira
      <n>  = la clave expira en n días
      <n>w = la clave expira en n semanas
      <n>m = la clave expira en n meses
      <n>y = la clave expira en n años
¿La clave es válida para? (0)
La clave no expira.
¿Es correcto? (s/N) s
```

Para generar la clave, GPG le pide introducir su nombre, su dirección de correco electrónico (e-mail) y un comentario. En el nombre, no dude en poner sus apellidos y su nombre. Se podrá utilizar luego el comentario como alias para determinados comandos GPG. Estos elementos no se podrán cambiar más adelante. Por el contrario, le será posible añadir direcciones de email con el fin de asociar varias de ellas a una sola clave.

```
Necesita un identificador de usuario para identificar su clave. El programa
construye el identificador a partir del Nombre Real, Comentario y Dirección
de Correo electrónico de esta forma:
   "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Nombre y apellidos: javier olivares
Dirección de correo electrónico: jolivares@JJoSoft.com
Comentario: Soft
Ha seleccionado este ID de usuario:
   «javier olivares (Soft) <jolivares@JJoSoft.com>»

¿Cambia (N)ombre, (C)omentario, (D)irección o (V)ale/(S)alir? V
```

La **passphrase** o frase de contraseña complementa la clave privada. Se pedirá esta frase a cada envío o recepción de mensaje privado encriptado. También asegura, al menos mientras no la encuentren, una cierta seguridad de la clave privada, incluso si la roban, ya que esta clave no se puede utilizar sin esta frase. Elija una frase (o palabra) bastante compleja, que recuerde siempre, pero no muy larga, ya que tendrá que teclearla de manera regular.

```
Necesita una frase de contraseña para proteger su clave secreta.
Inserte la frase de contraseña:
Repita la frase de contraseña:
```

GPG efectúa la última etapa, la generación de las propias claves. GPG va a utilizar información introducida y genera números aleatorios. Le damos un consejo: durante el cálculo, haga trabajar su ordenador para que el generador de números aleatorios «espabile».

```
gpg: /home/jolivares/.gnupg/trustdb.gpg: se ha creado base de datos de confianza
gpg: clave F2729638 marcada como de confianza absoluta
claves pública y secreta creadas y firmadas.

gpg: comprobando base de datos de confianza
gpg: 3 dudosa(s) necesaria(s), 1 completa(s) necesaria(s),
    modelo de confianza PGP
gpg: nivel: 0  validez:   1  firmada:   0
    confianza: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/F2729638 2017-08-09
     Huella de clave = 4CB5 8D90 2EAD 2522 DE9E  F380 719E 2A3D F272 9638
uid                  javier olivares (Soft)
<jolivares@JJoSoft.com>
sub   2048R/C4489841 2017-08-09
```

Las últimas cuatro líneas resumen toda la información relativa al trabajo efectuado por GPG:

- **pub**: da la longitud de la clave (2048 bits), de número F2729638, generada el 09/08/2017.

- **huella**: es el fingerprint que permite determinar la validez de la clave pública, un poco como la suma md5 de cualquier archivo. Es casi imposible (pero no del todo), que claves de gran tamaño tengan una huella idéntica.
- **uid**: información nominativa (nombre, comentario o alias/seudo, dirección e-mail).
- **sub**: es el tamaño (2048 bits), el número y la fecha de generación de la clave privada.

c. Generar una clave de revocación

Generar una clave de revocación le permitirá revocar su clave, o sea, cancelar su validez. Está firmada por su clave privada. Piense en dos casos: si le han robado su clave privada, hay que cancelarla para que deje de ser válida. Si ha perdido su passphrase, también tendrá que revocar su clave. En ambos casos, tendrá que volver a generar una clave. La clave de revocación permite a todas las personas que disponen de una clave pública comprobar que la cancelan. Una vez generada la clave de revocación, consérvela en un sitio seguro, no la publique ni tampoco la pierda.

Por lo tanto, es muy importante crear una clave de revocación justo después de haber creado el par de claves pública/privada y guardarla en un lugar seguro. Eso es así por varias razones: si pierde por ejemplo su clave privada, no tendrá ya la oportunidad de generar la clave de revocación. Si se olvida de su passphrase, no podrá utilizar más su clave privada; por lo tanto, no podrá crear la clave de revocación.

En contraposición, tenga cuidado en no divulgar su clave de revocación: si se conociera, cualquiera podría revocar una clave que usted utiliza.

Inicie el comando **gpg** con `--gen-revoke` y el nombre o el comentario (alias, seudo) asociado a la clave. Teclee la razón por la cual quiere generar una clave de revocación. Dé la razón que quiera y el comentario asociado con la elección y confirme.

```
jolivares@ubuntu:~$ gpg --gen-revoke soft

sec  2048R/F2729638 2017-08-09 javier olivares (Soft)
<jolivares@JJoSoft.com>
```

```
¿Crear un certificado de revocación para esta clave? (s/N) s
Elija una razón para la revocación:
  0 = No se dio ninguna razón
  1 = La clave ha sido comprometida
  2 = La clave ha sido reemplazada
  3 = La clave ya no está en uso
  Q = Cancelar
(Aquí seguramente desee elegir 1)
¿Su decisión? 1
```

Introduzca una descripción opcional; termínela con una línea vacía:

```
> Ejemplo público para libro
>
Causa de revocación: Se comprometió la clave.
Ejemplo público para libro.
¿Está de acuerdo? (s/N) s
```

Luego, tiene que teclear, como cada vez que se debe utilizar la clave privada, su frase de contraseña:

```
Necesita una contraseña para desbloquear la clave secreta
del usuario: "javier olivares (Soft) <jolivares@JJoSoft.com>"
clave RSA de 2048 bits, ID F2729638, creada el 2017-08-09
```

Ya se ha generado la clave. El sistema le muestra las instrucciones de uso: proteja su clave.

```
se fuerza salida con armadura ASCII.
Certificado de revocación creado.

Consérvelo en un medio que pueda esconder; si alguien consigue
acceso a este certificado podrá usarlo para inutilizar su clave.
Se recomienda imprimir este certificado y guardarlo en otro lugar, por
si acaso su soporte resulta imposible de leer. Pero cuidado: ¡el sistema
```

```
de impresión de su máquina podría almacenar los datos y hacerlos accesibles
a otras personas!
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1
Comment: A revocation certificate should follow

iQE5BCABAgAjBQJZiyXmHB0CRWplbXBsbyBwdWJsaWNvIHBhcmEgbGlicm8ACgkQ
cZ4qPfJyljhdYgf+ORfa2IE5B+9LBjWmUlJD5h9JvJurtswCgfnEHItvh7Zh4N2W
cqJAB2iO89NXmPGYe3a2qiAYHeYBOXcH7ApiK1Hn6fRz8u8ELyupCcnMrJD+a1OY
A99ZWEBoMOQEUqmoS3eA3+YHlKMe4jT97VB8j32+OeCk25vdpxPJa1sC6H+XytLL
h//0zeDWSGcHqlQURso1tOhomF4WNL/Ctuj4909YGpgDM/3BtPYGEXwsfJejw6Mp
rhCkA/ENV8tCpXvYorZTarymTU2Bmvq4pfnK9MIUbBIubm7Ormmo6ROgtwmXf0zP
IlrHr/53Q+FJ1BtmNyvjSxMlhNhQP2cT8tfgrA==
=viWn
-----END PGP PUBLIC KEY BLOCK-----
```

Deje su clave GPG en un soporte seguro: imprímala, póngala en un cd grabado (incluso de forma exclusiva), en un pendrive (no confíe en este soporte), en una caja fuerte o aprendrásela de memoria (si es buena).

d. Gestionar el almacén de claves

Haga la lista de las claves generadas por GPG con el parámetro `--list-key`. Para cada clave, puede ver la información relativa a las claves públicas y privadas, su propietario y la posible fecha de expiración.

```
jolivares@ubuntu:~$ gpg --list-keys
/home/jolivares/.gnupg/pubring.gpg
---------------------------------
pub   2048R/F2729638 2017-08-09
uid                  javier olivares (Soft) <jolivares@JJoSoft.com>
sub   2048R/C4489841 2017-08-09
```

Se muestran las firmas, o más bien la lista de las firmas de cada una de sus claves, por clave pública y clave privada, con la `--list-sigs` que incluye la `--list-key`. Más adelante se explicarán las firmas.

```
jolivares@ubuntu:~$ gpg --list-sigs
/home/jolivares/.gnupg/pubring.gpg
---------------------------------
pub   2048R/F2729638 2017-08-09
uid                  javier olivares (Soft) <jolivares@JJoSoft.com>
sig 3       F2729638 2017-08-09  javier olivares (Soft)
<jolivares@JJoSoft.com>
sub   2048R/C4489841 2017-08-09
sig         F2729638 2017-08-09  javier olivares (Soft)
<jolivares@JJoSoft.com>
```

e. Exportar la clave pública

Para que los demás puedan utilizar su clave pública para mandar mensajes, les debe proporcionar la clave y, por lo tanto, exportarla en un formato legible. Eso se hace con `gpg --export --armor` :

```
jolivares@ubuntu:~$ gpg --export --armor
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQENBFmLIZsBCADUbN2MLDbk6b+JE48D6al1UvSEEtlGvYsfe886aU981Om5Yb9P
8Xttf7JX6I4nDG1CUtfkLvxaySOMEcLEAR49gyO4mblXfjn8TqXazbFdM9Xk653+
NOX9406QgCbTl5nLCa8vH7/SaDMb+YtjtmtNGz6LDwJTP9Xv+KIkrJrumTBxvJO6
4SB8yFG9Yp624cF4nQ+TdaVGTG0TVv0xF8jOGjizxnz3gsO0YvGHmEu10NSpzxfF
WaDQAJFWEOOxLsaRFtPmux6ozRodfDcuTT9Dw0DPZOhQxS7ISzvgV2wroj7sTNMJ
0cOVHAp+i5OmHmj15bQ6BBzYrBxqWwR73RY3ABEBAAG0LmphdmllciBvbGl2YXJl
cyAoU29mdCkgPGpvbGl2YXJlc0BKSm9Tb2Z0LmNvbT6JATgEEwECACIFAlmLIZsC
GwMGCwkIBwMCBhUIAgkKCwQWAgMBAh4BAheAAAoJEHGeKj3ycpY45UQH/jj+4W+s
...
H878snA/Wuo7IsyEh+pN1/z79ufbuH2Vv26BuAyg/rHZHlbW1DsAEQEAAYkBHwQY
AQIACQUCWYshmwIbDAAKCRBxnio98nKWOBpvB/9nZMr+1ivG3C/jz+dx/Kq7o4v+
6c+WZKNbEYKVwA6187PgylfO9El/x7NL9OMiZwgI7IvoAaSAhUkoha+tQOlK7PYs
/R+DJpKVa8kyf7304RYZwpEllU1M0NRDacXuX1SxxnPr4rpnI3rKAa91vluQM7Zl
GPy+EDWGNeCfnGeckKMkO3k14gFw9kSc8tj7WBjkVQ03fbKleFVM75buTSZY0t3/
sOekXw3ZlWFQHysZAYHE43y0mv8HyV7H7LkNeqkZ5G6fAY20xoB9mDh4jNW973vO
...
dAJntEJmMu78MWaFXcNpdXSWuLE6ZMAL1SUIFzBKb6C64B6Up07XhnktIe8t
```

```
=jJUm
-----END PGP PUBLIC KEY BLOCK-----
```

La salida está truncada porque es demasiado larga. Basta con copiar este bloque completo y mandarlo, con el método que desee, a la persona a la cual quiere facilitársela (archivo adjunto de mail, servidores de claves, publicada en un sitio, etc.).

Puede exportar su clave pública hacia un servidor de claves GPG, por defecto keys.gnupg.net. Así, una persona que desea obtener su clave pública, para firmarla por ejemplo como veremos más adelante, podrá recuperarla en el servidor correspondiente.

```
jolivares@ubuntu:~$ gpg -k
/home/jolivares/.gnupg/pubring.gpg
------------------------------------
pub   2048R/F2729638 2017-08-09
uid                  javier olivares (Soft) <jolivares@JJoSoft.com>
sub   2048R/C4489841 2017-08-09
$ gpg --send-keys 515D9D90
gpg: send key 515D9D90 to server hkp keys.gnupg.net
```

Si luego se dirige al servirdor http://keys.gnupg.net/ y teclea **soft** (por ejemplo, el e-mail o el identificador de clave funcionan también), obtendrá la lista de las claves correspondientes y, entre ellas, la anterior.

f. Importar una clave

Puede importar una clave en su almacén con el parámetro `--import`. El segundo parámetro puede ser un archivo que contiene la clave. Si no lo tiene, efectúe un copiar-pegar de la clave en la consola, con la última línea que debe ser `-----END PGP PUBLIC KEY BLOCK-----` ; luego teclee [Ctrl] **D**.

```
jolivares@ubuntu:~$ gpg --import
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1
```

```
mQENBFmLIZsBCADUbN2MLDbk6b+JE48D6al1UvSEEtlGvYsfe886aU981Om5Yb9P
8Xttf7JX6I4nDG1CUtfkLvxaySOMEcLEAR49gyO4mblXfjn8TqXazbFdM9Xk653+
NOX9406QgCbTl5nLCa8vH7/SaDMb+YtjtmtNGz6LDwJTP9Xv+KIkrJrumTBxvJO6
4SB8yFG9Yp624cF4nQ+TdaVGTG0TVv0xF8jOGjizxnz3gsO0YvGHmEu10NSpzxfF
WaDQAJFWEOOxLsaRFtPmux6ozRodfDcuTT9Dw0DPZOhQxS7ISzvgV2wroj7sTNMJ
0cOVHAp+i5OmHmj15bQ6BBzYrBxqWwR73RY3ABEBAAG0LmphdmllciBvbGl2YXJl
cyAoU29mdCkgPGpvbGl2YXJlc0BKSm9Tb2Z0LmNvbT6JATgEEwECACIFAlmLIZsC
GwMGCwkIBwMCBhUIAgkKCwQWAgMBAh4BAheAAAoJEHGeKj3ycpY45UQH/jj+4W+s
kTVoir68jrlFyQovFKBGGGD5tKwg9HLRYGboIi3G9piTWdCdv61y7d52aofNKH9Q
Qh90Qw+YPrJaRmh6MWJi1t0vYPnJ0XxJtAKZfpnKCWbS0IktZVZYVgMP+DuzlVSs
UQCIOeLwE1ZzzbMqbvlbOSuuqKPCCUDi7dsMqFXmDvvviPoO9HEn+Aa99cNCRsAs
9bBCqs+5rM1ccKodtD0O/oTjXRDmebbCUlmNqTH7umXQw/mfoFVomikf9wk+5hIG
YqOQVtgVaMj7hBx7BFSmyRXScsBfUELDIMDlUxCWxnyAdnAAK/2EToTfJkZq54sM
ul8qx+eTVBXQIQe5AQ0EWYshmwEIALW6OaOmi1tDRcRKoqL1O2ekE7G7svPW9QTX
M4nCaND4VqcgU6uIsZaKHmII/0u3TEnETWPTJ57kIDMqOByt1wppEqC4ykj+nFWc
FaTHsUFstPHz42XfdhKqWiWgbKyEHuA8vC93zj+koGat0fPrqw9DbYHJG6qWrUnB
wiip5I69z+L3lnvdjehhuuWO0cSjbCRib+F2zqzNMK05u8f8Wi+u/+Qzbjt6684b
UDzYqEDkTzakk23KKqVs2vo1QvlR+csHsmy9SlrJreAI5gMzTM8p1nbHKY9stpeS
H878snA/Wuo7IsyEh+pN1/z79ufbuH2Vv26BuAyg/rHZHlbW1DsAEQEAAYkBHwQY
AQIACQUCWYshmwIbDAAKCRBxnio98nKWOBpvB/9nZMr+1ivG3C/jz+dx/Kq7o4v+
6c+WZKNbEYKVwA6187PgylfO9El/x7NL9OMiZwgI7IvoAaSAhUkoha+tQOlK7PYs
/R+DJpKVa8kyf7304RYZwpEllU1M0NRDacXuX1SxxnPr4rpnI3rKAa91vluQM7Zl
GPy+EDWGNeCfnGeckKMkO3k14gFw9kSc8tj7WBjkVQ03fbKleFVM75buTSZY0t3/
sOekXw3ZlWFQHysZAYHE43y0mv8HyV7H7LkNeqkZ5G6fAY20xoB9mDh4jNW973vO
dAJntEJmMu78MWaFXcNpdXSWuLE6ZMAL1SUIFzBKb6C64B6Up07XhnktIe8t
=jJUm
-----END PGP PUBLIC KEY BLOCK-----
gpg: clave F2729638: «javier olivares (Soft) <jolivares@JJoSoft.com>» sin cambios
gpg: Cantidad total procesada: 1
gpg:              sin cambios: 1
```

También puede recibir una clave de un servidor:

```
jolivares@ubuntu:~$ gpg --keyserver keys.gnupg.net --recv-key D0FE7AFB
gpg: solicitando clave D0FE7AFB de hkp servidor keys.gnupg.net
gpg: clave D0FE7AFB: clave pública "Josh Triplett <josh@joshtriplett.org>"
importada
gpg: clave D0FE7AFB: clave pública "Josh Triplett <josh@joshtriplett.org>" importada
```

```
gpg: 3 dudosa(s) necesaria(s), 1 completa(s) necesaria(s),
modelo de confianza PGP
gpg: nivel: 0  validez:   1  firmada:   0  confianza: 0-, 0q, 0n, 0m, 0f, 1u
gpg: Cantidad total procesada: 2
gpg:              importadas: 2  (RSA: 1)
```

Puede suprimir una clave del almacén de la manera siguiente:

```
jolivares@ubuntu:~$ gpg --delete-keys D0FE7AFB
gpg (GnuPG) 1.4.20; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


pub  1024R/D0FE7AFB 2014-06-16 Josh Triplett <josh@joshtriplett.org>
¿Eliminar esta clave del almacén de claves? (s/N) s
```

Para terminar, puede volver a generar la fingerprint de una clave, o sea, su huella. Como vimos durante la generación de las claves, sólo tendrá la prueba de que una clave pública pertenece realmente a una persona determinada si su huella es idéntica a la que dicha persona le ha suministrado.

```
jolivares@ubuntu:~$ gpg --fingerprint jolivares@JJoSoft.com
pub   2048R/F2729638 2017-08-09
      Huella de clave = 4CB5 8D90 2EAD 2522 DE9E  F380 719E 2A3D F272 9638
uid                  javier olivares (Soft) <jolivares@JJoSoft.com>
sub   2048R/C4489841 2017-08-09
```

g. Firmar una clave

Firmar una clave pública es certificar que esta clave pertenece realmente a la persona indicada por el identificador. Por eso hay que comprobar la huella antes de firmar una clave. Para firmar la clave que le ha transmitido, por ejemplo la de su amigo Esteban, tiene que importarla primero a su propio almacén:

```
jolivares@ubuntu:~$ gpg --import
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.9 (GNU/Linux)

mQGiBEoB63QRBACrfwOSo2EKJRYF0lh0Y06X2ZNNEq8+50IJmXYTXLx4b+niYzjf
V1pW5/0Of1/iM80RFU8NjkIKuxKnK0jji/WQbwAyWcTVMQ8CNrH2Fgbahi+8mn/Z
...
dAIbDAUJAA0vAAAKCRD1IZKWFJB27BVSAJoCQ1i49aTNi3NopE+zgiZYwYSXaACe
LFxBG3QJUkmLpSX8L/9vzjAZ+6Y==QMLk
-----END PGP PUBLIC KEY BLOCK-----
gpg: clave 149076EC: clave pública «Esteban (Clave de prueba)
<esteban@tele2.es>» importada
gpg:       Cantidad total tratada: 1
gpg:                   importada: 1
```

Compruebe que todo se ha desarrollado correctamente haciendo la lista de las claves:

```
jolivares@ubuntu:~$ gpg -k Esteban
pub   1024D/149076EC 2017-02-042017-02-04  [expira: 2017-02-042017-02-04]
uid                  Esteban (Clave de prueba) <esteban@tele2.es>
sub   2048g/ABC8FBE2 2017-02-042017-02-04 [expira: 2017-02-042017-02-04]
```

Recupere la huella de la clave de Esteban para compararla con la que le ha facilitado:

```
jolivares@ubuntu:~$ gpg --fingerprint Esteban
pub   1024D/149076EC 2009-05-06 [expira: 2017-02-04]
  Huella de la clave = B50E 5C76 C0A6 4BDC F83E  FB8F F521 9296 1490 76EC
uid                  Esteban (Clave de prueba) <esteban@tele2.es>
sub   2048g/ABC8FBE2 2017-02-042017-02-04 [expira: 2017-02-042017-02-04]
```

Si la huella es correcta, edítela: eso abre un pequeño intérprete de comandos para trabajar en el almacén o la clave correspondiente. Observe el estado de confianza y validez: está en "desconocido" porque la clave no está firmada:

```
jolivares@ubuntu:~$ gpg --edit-key Esteban
```

```
gpg (GnuPG) 2.0.9; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


pub  1024D/149076EC  creado: 2017-02-04  expira: 2017-02-04  utilización: SC
                confianza: desconocida       validez: desconocida
sub  2048g/ABC8FBE2  creado: 2017-02-04  expira: 2017-02-04  utilización: E
[ desconocida] (1). Esteban (Clave de prueba) <esteban@tele2.es>

Comando>
```

Teclee **trust** para dar su nivel de confianza:

```
Comando> trust
pub  1024D/149076EC  creado: 2017-02-04  expira: 2017-02-04  utilización: SC
                confianza: desconocida       validez: desconocida
sub  2048g/ABC8FBE2  creado: 2017-02-04  expira: 2017-02-04  utilización: E
[ desconocida] (1). Esteban (Clave de prueba) <esteban@tele2.es>

Decida ahora hasta qué punto confía en este
usuario para que compruebe las claves de los demás usuarios (puede
usted comprobar su pasaporte, las huellas de varias
fuentes diferentes, etc.)

  1 = no sé o no diré
  2 = NO confío
  3 = confío un poco
  4 = confío totalmente
  5 = doy una confianza última
  m = volver al menú principal

¿Su decisión? 4

pub  1024D/149076EC  creado: 2017-02-04  expira: 2017-02-04  utilización: SC
                confianza: entera      validez: desconocida
sub  2048g/ABC8FBE2  creado: 2017-02-04  expira: 2017-02-04  utilización: E
[ desconocida] (1). Esteban (Clave de prueba) <esteban@tele2.es>
```

> Observe que la validez mostrada para la clave no es necesariamente
> correcta hasta que usted no vuelva a arrancar el programa.

Observe los niveles de confianza: de 1 a 5, por orden ascendente. La traducción española es un poco sorprendente. La opción 3, por ejemplo, corresponde a «confío un poco». Si elige 5, se le pedirá una confirmación.

Teclee ahora **sign** para firmar la clave con su clave privada. Tendrá que introducir su passphrase:

```
Comando> sign

pub  1024D/149076EC  creado: 2017-02-04  expira: 2017-02-04  utilización: SC
                     confianza: entera      validez: desconocida
 Huella de la clave principal:
B50E 5C76 C0A6 4BDC F83E  FB8F F521 9296 1490 76EC

     Esteban (Clave de prueba) <esteban@tele2.es>

Esta clave expira el 2017-02-04.
¿Está seguro de que quiere firmar esta clave
con la suya? «Javier Olivares <javier.olivares@dominio.es>»
(13E021A8)

¿Firmar realmente? (s/N) s

Necesita una frase de contraseña para desbloquear la
clave secreta para el usuario: «Javier Olivares
<javier.olivares@dominio.es>»
clave de 1024 bits DSA, ID 13E021A8, creada el 2017-02-04
```

Ahora la clave de Esteban está firmada.

h. Firmar y cifrar

Ahora que dispone de la clave pública de sus amigos, puede firmar un mensaje (de su

parte) a su atención.

```
jolivares@ubuntu:~> gpg --clearsign -u javier.olivares@dominio.es -a

Necesita una frase de contraseña para desbloquear la
clave secreta para el usuario: «Javier Olivares
<javier.olivares@dominio.es>»
clave de 2048 bits DSA, ID 515D9D90, creada el 2017-02-04

Hola, amigos
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hola, amigos
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.22 (GNU/Linux)

iQEcBAEBAgAGBQJYla6uAAoJEAQi5NJRXZ2QGDsH/A5l1P9nlBHD6olZnYRsXVab
IZDvkMS9F9CvwAOm/VZds1ayfjMP0GAUqbdinw8cMzPSz+EZe45GcbkTe8K731P9
zb1w5N2U8SM1VPkLCV7NFvS0fbdlephoyaVQk4RRADT930dmPfZJyANlyCNnNKxM
FcKD1amlOGSdqREqyMlizndu5g/+q6IzcOZStuM6kXIuJQJHDmrUJSKTNcgnhfv+
xFXnxecccZuwd6nf71ph+K2D5FcdCC9KUNGKHGYIif0Vy+lca12TG5xehsXv5/Rc
e+oBkz29Z3d2u4cOZ4rStY6Yx5oZ++iPOHORJqgZjKvh4QeHPFIksV7coY4Jl3g=
=u1nb -----END PGP SIGNATURE-----
```

Mande su mensaje en forma de texto, con la firma, al destinatario. Éste, al recibir el mensaje, hará la operación inversa: comprobará si la firma corresponde realmente a usted y al mensaje (suponiendo que el mensaje haya sido colocado en un archivo mensaje.asc). Es ese caso: gpg confirma que Javier Olivares ha firmado el mensaje.

```
esteban@slyserver:~> gpg --verify message.asc
gpg: Firma
hecha el sab 04 de febrero de 2017 22:24:31 CEST con la clave RSA ID 515D9D90
gpg: Firma correcta de «Javier Olivares
<javier.olivares@dominio.es>»
```

La etapa siguiente consiste en cifrar (encriptar) el mensaje completo, de tal manera que

sólo el destinatario pueda descifrarlo. Para que Javier pueda mandar un mensaje cifrado a Esteban del cual tiene la clave pública, haga lo siguiente:

```
jolivares@ubuntu:~> gpg -r Esteban -e --armor
Hola Esteban, estoy probando el envío de un mensaje encriptado
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.22(GNU/Linux)

hQEMA8PaFKWBgPbqAQgAxt9+P8Umtsee7s/j83i1kBs12Z2rUGr54QE7V38A1735
...
PjGCfxEZj6PHRMpd84x+i2yFCg==
=xiHX
-----END PGP MESSAGE-----
```

Mande el mensaje encriptado (del BEGIN al END) a Esteban. Éste, por su parte, intentará descifrarlo: debe ser el único en poder hacerlo con su clave privada.

```
esteban@slyserver:~> gpg --decrypt message.asc
Necesita una frase de contraseña para desbloquear su clave
secreta para el usuario. «Esteban (Clave de prueba) <esteban@tele2.es>»
clave de 2048 bits ELG, ID 8180F6EA , creada el 2017-02-04 (ID clave
principal 515D9D90 )

gpg: cifrado con una clave de 2048 bits ELG, ID 8180F6EA, creada el 2017-02-04
     «Esteban (Clave de prueba) <esteban@tele2.es>»
Hola Esteban, estoy probando el envío de un mensaje encriptado.
```

El mensaje que se muestra no contiene errores.

Última etapa: cifrar y firmar al mismo tiempo. Si su mensaje se encuentra en un archivo `test.msg`, teclee lo siguiente:

```
jolivares@ubuntu:~> gpg -u javier.olivares@tele2.es -r Esteban -a -e -s
test.msg

Necesita una frase de contraseña para desbloquear la clave
```

```
secreta para el usuario: «Javier Olivares
<javier.olivares@dominio.es>»
clave de 2048 bits DSA, ID 515D9D90 , creada el 2017-02-04
```

Se ha creado un archivo test.msg.asc que contiene el mensaje cifrado y encriptado.

```
jolivares@ubuntu:~> cat test.msg.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.22 (GNU/Linux)

QEMA8PaFKWBgPbqAQgAgWBMCwz905M0zp0LrI/pkdI2d9EkiIj/zJKmEsMVohWw
...
vz0fz1ymXPa5Qqu6U6WK/DaMzj01DkGb5RKTJ5o8cnktog1LRPKXlQ==
=fhqg

-----END PGP MESSAGE-----
```

Por su parte, Esteban va a descifrar y comprobar el mensaje de la manera siguiente:

```
esteban@slyserver:~> gpg -d -v test.msg.asc
Version: GnuPG v2.0.22 (GNU/Linux)
gpg: encabezamiento de armadura:
gpg: la clave pública es 8180F6EA
gpg: utilización de la subclave 8180F6EA en lugar de la clave
principal 515D9D90

Necesita una frase de contraseña para desbloquear la
clave secreta para el usuario: «Esteban (Clave de prueba) <esteban@tele2.es>»
gpg: utilización de la subclave 8180F6EA en lugar de la clave
principal 515D9D90
clave de 2048 bits ELG, ID 8180F6EA, creada el 2009-05-06 (ID clave
principal 515D9D90)

gpg: no running gpg-agent - starting one
gpg: cifrado con una clave de 2048 bits ELG, ID 8180F6EA, creada el 2017-02-04
     «Esteban (Clave de prueba) <esteban@tele2.es>»
gpg: datos cifrados con AES256
```

```
gpg: nombre de archivo original: 'test.msg'
Hola, esto es un mensaje cifrado y encriptado
gpg: Firma
hecha el sab. 04 de febrero de 2017 22:33:47 CEST con la clave DSA ID 515D9D90
gpg: utilización del modelo de confianza PGP
gpg: Firma correcta de «Javier Olivares
<javier.olivares@dominio.es>»
gpg: firma binaria, algoritmo SHA256
```