Capítulo 11

Comunicaciones seguras

En el interior y exterior de las redes de nuestro sistema pueden existir múltiples peligros acechandonos detras de cada switch o router, es necesario proteger nuestras conexiones y las de nuestros usuarios para que no sean escuchadas por usuarios no autorizados o si esto no lo podemos asegurar, al menos que les sean incomprensibles.

11.1. Shell seguro: OpenSSH

SSH es una herramienta de acceso remoto que nos permite iniciar la sesión en un sistema remoto de una forma segura. El talón de Aquiles de la mayoría de las redes es el hecho de que normalmente las comunicaciones entre sistemas se pasan sobre una red en texto plano. Por lo tanto, podemos fortalecer las máquinas individuales todo lo que deseemos, pero si iniciamos una sesión en ellas remotamente con un programa de terminal inseguro, los ladrones pueden robar nuestras credenciales de registro con un *sniffer* de red (escucha clandestina de paquetes de red). Después pueden iniciar una sesión sin ningún problema. Una de las herramientas de acceso remoto más populares, Telnet, sufre esta deficiencia.

SSH soluciona el problema utilizando tanto la criptografía simétrica como la clave pública para cifrar la sesión desde la primera pulsación de tecla. De este modo, todo el que esté escuchando su conexión obtiene un sonido aleatorio. SSH no sólo proporciona confidencialidad para nuestros datos utilizando el cifrado sino que además proporciona una sólida autenticación, que impide la suplantación de identidad, esto lo hace utilizando certificados digitales para autenticar a los usuarios.

No hay que confundir SSH con SSL, es estándar de cifrado Web. Aunque ambos realizan la misma función, SSH funciona con cualquier protocolo, mientras que SSL está diseñado principalmente para las comunicaciones Web.

SSH también incluye SCP, un reemplazo seguro para RPC, la herramienta de copia remota, y SFTP un reemplazo seguro para FTP. SSH también puede utilizarse para crear un túnel con otros protocolos entre máquinas, como HTTP y SMTP. Al utilizar esta familia de programas en lugar de sus homólogos más antiguos, nos aseguramos de que no se están leyendo nuestras comunicaciones remotas con los servidores.

Eliminar el uso de Telnet y FTP en nuestra red puede ser difícil, pero cuanto más lo hagamos, más seguros estaremos.

11.1.1. Cliente OpenSSH

Para acceder al sistema remoto con SSH, necesitamos un cliente SSH en nuestro lado y tiene que existir un servidor SSH ejecutándose en el lado remoto. Aunque SSH no está tan difundido como Telnet, poco a poco se está haciendo más popular. Cisco está instalando SSH en sus enrutadores, aunque todavía deja activado el servidor Telnet de forma predeterminada, mientras que SSH es opcional.

Debemos asegurarnos de estar utilizando una versión 3.6 o más actual; algunas versiones anteriores tenían fallos en su implantación de protocolos criptográficos y son susceptibles de ataques. De hecho,

es recomendable asegurarse de tener la útima versión disponible, ya que el código se está mejorando constantemente y los algoritmos se están ajustando.

SSH tiene un número de usos realmente interesantes distintos a asegurar un inicio de sesión en un sistema remoto. Se puede utilizar para crear un túnel para cualquier servicio a través de un canal cifrado entre servidores.

Para instalarlo en nuestro sistema hay que realizar un apt-get:
`#apt-get install openssh-client-udeb`

La sintaxis SSH básica para iniciar una sesión remotamente es:
`$ssh -l login hotname`

Donde *login* es su nombre de usuario en el sistema remoto y *hotname* es el anfitrión al que está intentando conectar con SSH. También se puede utilizar:
`$ssh login@hostname`

Por lo tanto y a modo de ejemplo, para registrarse en el servidor Web denominado *web.example.com* utilizando el nombre de inicio de sesión de *josan*, tendríamos que escribir:
`$ssh josan@web.example.com`

También podemos utilizar, `$ssh -l josan web.example.com` para iniciar la sesión. Si simplemente escribirmos `$ssh web.example.com`, el servidor supondrá que el nombre del usuario es igual que el del inicio de sesión del sistema.

En la tabla 11.1 podemos encontrar el resto de opciones de SSH:

Cuadro 11.1: Opciones del cliente SSH

Opción	**Descripción**
-c protocol	Utiliza un protocolo criptográfico específico (tiene que ser soportado por la versión que utilizamos de SSH)
-p port#	Se conoceta a un número de puerto específico en lugar de la puerto SSH predeterminado (puerto 22)
-P port#	Usa un puierto específico que no forma parte de la lista estándar de puertos propietarios, lo que normalmente significa un número por encima de 1024. Esto puede ser útil si tenemos un cortafuegos que imposibilita las comunicaciones en números de puertos inferiores.
-v	Muestra la salida larga. Útil para la depuración
-q:	Informa en modo silencioso, cdontrario del modo largo
-C:	Utiliza compresión del tráfico cifrado. PUde ser útil para conexiones demasiado lentas, como las telefónicas, pero es mejor tener un procesaro más potente para realizar la compresión o ralentizará mucho el rendimiento.

Si queremos personalizar nuestras conexiones, en el directorio */etc/ssh* encontraremos los archivos de configuración del servicio SSH. El archivo de configuración del cliente es */etc/ssh/ssh_config*.

11.1.2. Servidor OpenSSH

Para utilizar SSH debemos tener un cliente SSH ejecutándose en la máquina que deseamos conectar y un servidor SSH en la máquina a la que deseamos conectarnos. Los clientes FTP y Telnet normales no se conectarán a un servidor SSH. El cliente se encuentra integrado en la mayoría de los sistemas operativos Linux actuales, aunque puede que tengamos que seleccionar esta opción al instalar el sistema. El servidor SSH es normalmente opcional. Para determinar si ya está instalado, escribimos `$ps` y comprobamos si se está ejecutando el proceso *sshd*. Si no se está ejecutando, tendremos que instalar el servidor para permitir las conexiones de su máquina a través de SSH.

Pasemos a describir el proceso de instalación del servidor de SSH.

1. Lo primero es instalar el paquete si no lo tenemos en el sistema:

   ```
   #apt-get install openssh-server-udeb
   ```

2. Después hay que revisar los archivos de configuración que se encuentran en el directorio */etc/ssh* para asegurarse de que coincide con los parámetros de nuestro sistema. El archivo de configuración para el servidor es */etc/ssh/sshd_config*.

   A continuación detallo los campos que hay que revisar dentro de este archivo:

   - Port: Es el puerto que escucha SSH para las conexiones entrantes. Su valor predeterminado es 22. Si lo cambiamos, las personas que intenten conectarse con su sistema tendrán que cambiar manualmente el número de puerto en sus clientes SSH.
   - Protocols: Le indican al servidor los protocolos SSH que debe aceptar. El valor predeterminado es aceptar conexiones de tipo SSH1 y SSH2.
   - Hostkey: Claves para aceptar conexiones de clientes, proporciona la ubicación de las claves utilizadas para generar la autenticación de clientes. Éstas no son las mismas claves que las claves del servidor generadas en la instalación.

3. Antes de poder utilizar un servidor SSH tiene que generar sus distintas claves. Esto proporciona un identificador único para las claves de servidor. Para ello hay que escribir el siguiente comando:

   ```
   #ssh make-host-key
   ```

4. Ahora se puede iniciar el servidor SSH en la línea de comandos escribiendo:

   ```
   #sshd &
   ```

   Así se ejecutara el demonio del servidor SSH en segundo plano y escuchará continuamente las conexiones. Si se desea ejecutar *sshd* automáticamente al inicio (opción muy recomendable), hay que colocar dicha línea al final de un archivo al estilo de *rc.local* de otras distribuciones (véase apéndice D para componer este tipo de archivos).

También es posible facilitar la configuración mediante un módulo para la herramienta de administración web Webmin. Para la instalación se utiliza el siguiente apt:

```
#apt-get install webmin-sshd
```

Puerto de envío con OpenSSH

Aunque SSH se diseñó en principio para una interacción de línea de comandos tipo Telnet, también se puede utilizar para configurar un túnel entre dos máquinas para cualquier aplicación. Podemos crear una conexión segura entre dos servidores con la opción de puerto de envío integrada en SSH. Para realizar esta tarea, debemos tener SSH ejecutándose en ambos extremos de la conexión.

Con la siguiente declaración emitida en el extremo del cliente podemos realizar cualquier servicio sobre cualquier puerto:

```
#ssh -L local_port:local_host:remote_port remote_hostname -N &
```

Donde debemos reemplazar:

- local_port: Por un número aleatorio, mayor de 1024, elegido para realizar la nueva conexión cifrada
- local_host: Por la máquina local
- remote_port: Por el puerto del servicio con el que deseamos abrir un túnel en el extremo remoto

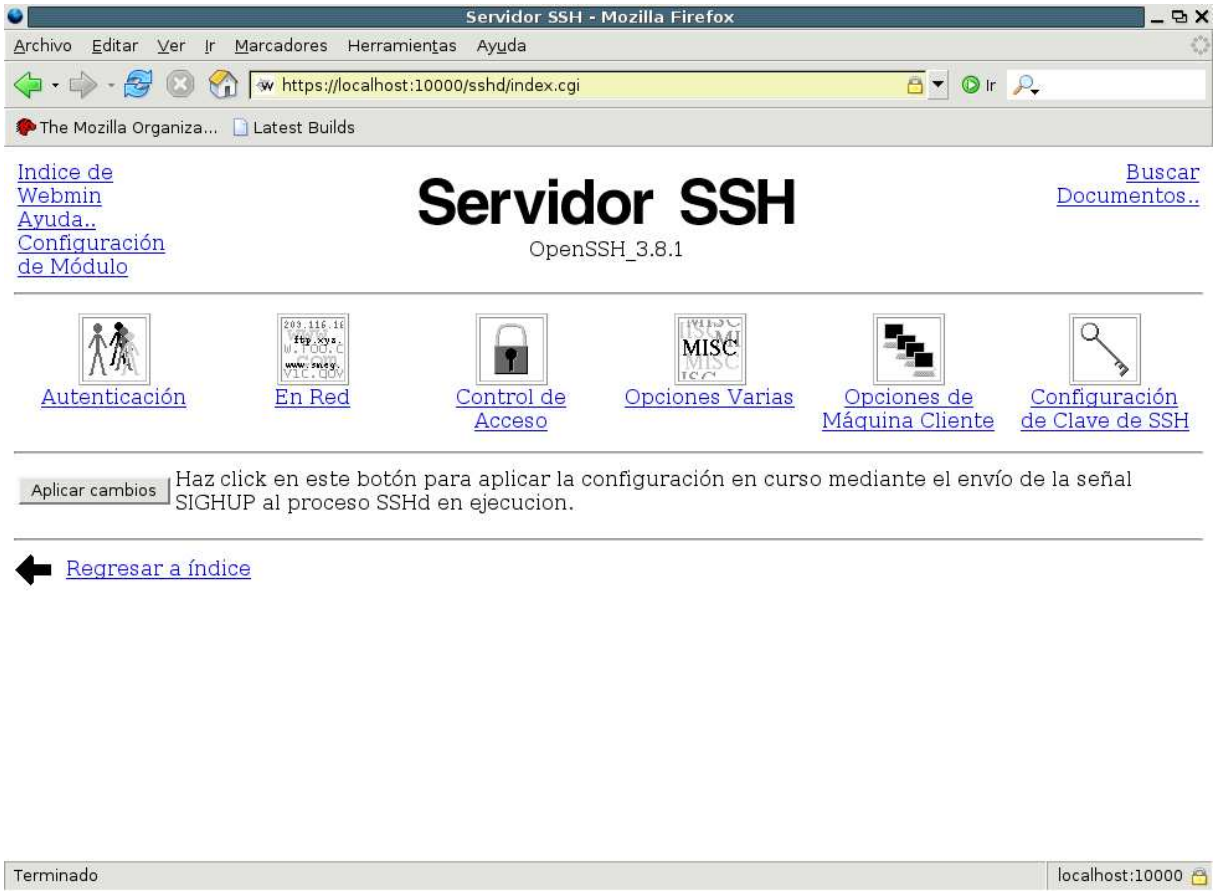

Figura 11.1: Interfaz gráfica Webmin para el servidor SSHD

- remote_hostname: Por la dirección IP o nombre del servidor en el otro extremo de la conexión

La opción *-L* le indica a SSH que debe escuchar el local_port en local_host y enviar cualquier conexión a remote_port en remote_host.

La opción *-N* le indica a SSH que no intente iniciar la sesión, sólo mantener la conexión abierta para el tráfico enviado.

Si se desea que esta configuración del sistema se mantenga en siguientes reinicios de la máquina hay que añadirla a un archivo de *rc.local* como se especifica en la sección D.

Esto podrían ser ejemplos de este tipo de usos:

- Si se necesita cifrar una conexión de correo electrónico:

  `#ssh -L 5000:localhost:25 192.168.0.1 -N &`, . . . donde la IP corresponde al servidor de correo de la red

- Si se necesita crear una conexión web segura con SSH en vez de SSL:

  `#ssh -L 5000:localhost:80 192.168.0.1 -N &`, . . . utilizando en el navegador *localhost:5000* se realizaran envíos a través de túnel seguro con el puerto web(80) en la máquina remota.

- También podemos tener varios puertos al mismo servidor, como es nuestro caso:

  ```
  #ssh -L 5000:localhost:25 -L 5001:localhost:80 192.168.0.1 -N &
  ```

Como se puede observar, SSH funciona extraordinariamente bien para crear una conexión segura entre dos máquinas para casi cualquier protocolo.

11.2. Criptografía y cifrado

Hoy en día la solidez del cifrado normalmente se mide por el tamaño de su clave. Independientemente de la solidez del algoritmo, los datos cifrados pueden estar sujetos a ataques por la fuerza en los que se prueban todas las combinaciones posibles de claves. Al final, el cifrado se puede romper. Para la mayoría de los códigos modernos con longitudes decentes, el tiempo para romper la clave a la fuerza se mide en milenios. Sin embargo, un fallo inadvertido en un algoritmo o el avance en la tecnología informática o en los métodos matemáticos pueden reducir este tiempo considerablemente.

Normalmente se cree que la longitud de la clave debe ajustarse para mantener seguros los datos durante una cantidad razonable de tiempo. Si el elemento es muy local, como las comunicaciones del campo de batalla o la información diaria sobre las acciones, un código que proteja estos datos durante semanas o meses está bien. Sin embargo, algo como número de tarjeta de crédito o los secretos de seguridad nacional tienen que mantenerse seguros durante un periodo de tiempo más prolongado y de forma eficaz para siempre. Por lo tanto, utilizar algoritmos de cifrado más débiles o longitudes de clave más cortas para algunas cosas está bien, siempre que la utilidad de la información para un intruso expire en un breve periodo de tiempo.

11.2.1. Tipos de cifrado

Criptografía simétrica

El primer tipo de cifrado, denominado criptografía simétrica, o cifrado de secreto compartido, se ha estado utilizando desde la época de los antiguos egipcios. Esta forma de cifrado utiliza una clave secreta, denominada secreto compartido, para cifrar los datos en un galimatías inteligible. La persona que se encuentra en el otro extremo necesita la clave compartida para desbloquear los datos (el algoritmo de cifrado). Podemos cambiar la clave y los resultados del cifrado. Se denomina criptografía simétrica porque se utiliza la misma clave para ambos extremos tanto para cifrar como para descifrar los datos.

El problema que surge con este método es que tenemos que comunicar la clave secreta de una forma segura al destinatario pretendido. Si nuestro enemigo intercepta la clave, puede leer el mensaje. Se han inventado todo tipo de sistemas para intentar solucionar esta fragilidad básica, pero el hecho sigue existiendo: tendremos que comunicar la clave secreta de alguna forma al destinatario pretendido antes de iniciar una comunicación segura.

Criptografía asimétrica

Una revolución en el cifrado fue la iniciada cuando Whitfield Diffie, Martin Hellman y Ralph Merkle inventaron la criptografía de calve pública. (En realidad, todavía existe algún debate sobre si el británico James Ellis en realidad inventó esta clave antes y la mantuvo en secreto, pero Diffie, Hellman y Merkle fueron los primeros en publicarla en 1976).

Estaban tratando de resolver el antiguo problema del intercambio de clave. Diffie se preguntaba cómo dos individuos que deseaban realizar una transacción financiera por una red electrónica podían hacerlo con seguridad. Llevaba mucho tiempo pensando en ello porque Internet estaba naciendo en aquél momento y el comercio electrónico todavía no existía. Si los gobiernos tenían muchos problemas tratando con el problema del intercambio de clave, ¿cómo podría controlarlo una persona media? Quería llegar a crear un sistema por el que las dos partes pudiesen mantener conversaciones protegidas y realizar transacciones seguras sin tener que intercambiarse una clave cada vez. Sabía que si resolvía el problema del intercambio de claves, sería un gran avance en la criptografía.

Diffie se asoció con Martin Hellman y Ralph Merkle. Tardaron algunos años, pero finalmente consiguieron crear un sistema denominado *Cifrado de clave pública* (PKE, Public Key Encription), también conocido como *Criptografía asimetrica*.

La criptografía asimétrica utiliza un cifrado que divide la clave en dos claves más pequeñas. Una de las claves se hace pública y otra sigue siendo privada. El mensaje lo ciframos con la clave pública del

destinatario. Éste puede descifrarla a continuación con su propia clave privada. Y lo mismo pueden hacer por nosotros, cifrando un mensaje con nuestra clave pública para poderlo descifrar con nuestra clave privada. La diferencia es que nadie necesita la clave privada de nadie para enviar un mensaje seguro. Utilizamos su clave pública, que no tiene que mantenerse segura (de hecho, se publican en repositorios como si fueran una guía telefónica). Al utilizar la clave pública del destinatario, sabemos que sólo esa persona puede descifrarlo utilizando su propia clave privada. Este sistema permite que dos entidades se comuniquen con seguridad sin ningún intercambio anterior de claves.

Normalmente la criptografía asimétrica se implanta mediante el uso de funciones de un sentido. En términos matemáticos, éstas son funciones fáciles de calcular en una dirección pero muy difíciles de calcular a la inversa. De esta forma podemos publicar nuestra clave pública, derivada a partir de nuestra clave privada, una tarea muy difícil de llevar a cabo al revés para determinar la clave privada. Una función común de un sólo sentido utilizada actualmente, es la descomposición en factores de números primos grandes. Es fácil multiplicar dos números primos y obtener un producto. Sin embargo, determinar cuál de las muchas posibilidades son los dos factores de un producto es uno de los problemas matemáticos más complejos (su tiempo de computación es NP-Completo[1]).

Si alguien inventase un método para deducir fácilmente factores de números primos grandes[2] en tiempo de computación lineal o polinómico, daría al traste con el mecanismo de cifrado de claves públicas actual, haciendo que cualquier tipo de comunicación basada en este tipo de algoritmos resultase insegura. Afortunadamente, otras funciones de un solo sentido funcionan bien para este tipo de aplicación, como los cálculos sobre curvas elípticas o el cálculo de logaritmos inversos sobre un campo finito.

Poco después del lanzamiento de Diffie, Hellman y Merkle, otro grupo de tres hombres desarrolló una aplicación práctica de la teoría. Su sistema para el cifrado público se denominó RSA, por sus nombres: Ronald Rivest, Adi Shamir y Leonard Adleman. Formaron una empresa que empezó a regular su sistema. La velocidad de adopción era lenta y su empresa estuvo a punto de quebrar, hasta que llegó el momento de aprovechar el emergente campo comercial de Internet con una empresa, entonces pequeña denominada Netscape. El resto es historia y ahora RSA es el algoritmo de cifrado de clave pública más utilizado. Diffie y Hellman finalmente lanzaron una aplicación práctica de su propiedad, pero sólo útil para intercambios de clave, mientras que RSA puede realizar la autenticación y el no reconocimiento.

Panorama actual

Hoy en día, el cifrado de clave pública se encuentra detrás de cada servidor web que nos ofrece una compra segura. Nuestra transacción se cifra sin tener que dar ni obtener una clave secreta y todo se produce en segundo plano. Como usuarios, lo único que conocemos es ese pequeño símbolo de candado SSL que se muestra en nuestro explorador para sentirnos seguros. No se puede imaginar el efecto que tendría sobre el comercio de Internet si cada vez que deseáramos comprar algo online tuviésemos que pensar en una clave secreta, cifrar el mensaje y comunicar posteriormente de alguna forma dicha clave a la otra parte. Evidentemente, el comercio electrónico no existiría tal y como existe actualmente si no existiese la criptografía de clave pública.

Por lo general las aplicaciones, combinan los dos tipos de criptografía. Utilizan primeramente criptografía de clave pública para acordar una clave simétrica aleatoria. Esta clave simétrica es una clave de sesión y sirve para realizar más rápidamente el cifrado y descifrado de la información que se mandan, ya que tiene un coste computacional mucho mas bajo.

Existen muchos algoritmos de cifrado, protocolos y aplicaciones diferentes basadas en estos dos tipos principales de cifrado. Las siguientes secciones explican algunos de estos tipos.

11.2.2. Estándares generales

El estándar de cifrado de datos (DES, Data Encryption Standar) es el estándar original que el gobierno de los Estados Unidos empezó a promover para su uso gubernamental y comercial. Pensado originalmente para se prácticamente inquebrantable en los años setenta, el aumento en potencia y el descenso en costo

[1] Denominación que se le da en informática teórica a los problemas más dificeles de calcular, que necesitan un tiempo exponencial para su resolución).

[2] Esto no es algo descabellado, hace muy pocos años se ha conseguido desarrollar un algoritmo que resuelve el problema de si un n.º tiene factores, en tiempo de computación polinomico.

de la informática hicieron que su funcionalidad de clave de 56 bits quedase obsoleta para información muy confidencial. Sin embargo, todavía se sigue utilizando en muchos productos comerciales y se considera aceptable para las aplicaciones de seguridad inferior. También se utiliza en productos que tienen procesadores más lentos, como las tarjetas inteligentes y los dispositivos que no procesan un tamaño de clave más largo.

TripleDES

También llamado 3DES es la versión DES más moderna y actualizada, su nombre implica lo que hace. Ejecuta DES tres veces en los datos en tres fases: cifrado, descifrado y cifrado de nuevo. En realidad no multiplica por tres la solidez de su código (la primera clave de código se utiliza dos veces para cifrar los resultados del proceso), pero sigue teniendo una longitud de clave efectiva de 168 bits, bastante solidez para la mayoría de usos.

RC4, RC5 y RC6

Este algoritmo de cifrado fue desarrollado por Ronald Rivest, uno de los desarrolladores de RSA, la primera aplicación gráfica de criptografía pública. Se han realizado mejoras a lo largo del tiempo para hacerla más sólida y solucionar problemas menores. La versión actual, RC6, permite una clave de 2.040 bits de tamaño y un tamaño de bloque variable de hasta 128 bits.

AES

Cuando el gobierno de los Estados Unidos se dio cuenta de que DES terminaría llegando al final de su vida útil, empezó a buscar un sustituto. Hubo muchos competidores, incluyendo RC6, Blowfish del reconocido criptógrafo Bruce Schneier y otros algoritmos meritorios. El concurso se resolvió a favor de AES, basado en un algoritmo denominado Rijindael diseñado por dos criptógrafos belgas. Este hecho es importante porque se utilizó una competición abierta para decidir el estándar. Asimismo, al seleccionar un algoritmo de dos desarrolladores no norteamericanos sin intereses comerciales significativos ayudaba a legitimar esta selección por todo el mundo. AES se está convirtiendo rápidamente en el nuevo estándar del cifrado. Ofrece una clave de hasta 256 bits, algo más que suficiente para el futuro previsible. Normalmente, AES se implanta en modo de 128 o 192 bits.

11.2.3. Aplicaciones de la criptografía

Hash

Los hash son funciones especiales de sentido único que proporcionan autenticación y verificación utilizando el cifrado. Una función hash recoge un archivo y lo coloca en una función para que se produzca un archivo de tamaño, en conjunto, mucho más pequeño. Al hacerlo, se produce una huella digital única, lo que nos proporciona una forma segura de saber que el archivo no se ha alterado de ninguna manera. Al utilizar una función hash para un archivo sospechoso y comparar su huella digital con la huella digital correcta, podremos saber si se ha producido algún cambio en el archivo. Es muy poco probable que un archivo con una estructura diferente produzca una huella idéntica. Incluso si cambia uno de los caracteres, se cambia la huella digital significativamente. Las posibilidades de que dos archivos diferentes produzcan el mismo hash son infinitesimales.

Normalmente los hash se proporcionan en versiones descargadas de software para asegurarse de que está obteniendo el software real, algo importante, especialmente con el software de libre distribución que se puede descargar desde réplicas de los servidores oficiales. El sitio web oficial normalmente incluye el hash correcto de la última versión. Si ambos no coinciden, sabremos que se han producido cambios, posiblemente sin el permiso o el conocimiento de los desarrolladores del software. El algoritmo hash más conocido se denomina MD5.

Cuando se instala el sistema, se pregunta al usuario que algoritmo hash quiere usar para las contraseñas. Es muy recomendable establecer MD5 como predeterminado, ya que actualmente se le considera uno de los mejores algoritmos de hash.

Certificados digitales

Los certificados digitales son las "firmas" del mundo comercial en Internet. Utilizan una combinación de tipos de cifrado para proporcionar autenticación y comprueban que quien se está conectando es realmente quien dice ser. En resumen, un certificado es una "certificación" expedida por una autoridad de la que nos fiemos y que permite fiarnos de la veracidad de lo que nos esta contando el titular del certificado.

Un certificado contiene la clave pública de la organización cifrada con la clave privada o la clave pública de una autoridad de firmas. El uso de una autoridad de certificados o firmas se considera el método más seguro de los dos. Si podemos descifrar el certificado con su clave publica, podremos suponer razonablemente que el sitio web pertenece a dicha organización.

Normalmente, los certificados se unen a un dominio determinado. Pueden ser emitidos por una entidad central, o creados y firmados localmente. Existen varias organizaciones de este tipo, entre ellas *VeriSign*, la empresa que además se encarga del sistema de nombres de dominio en Internet. Estas organizaciones han sancionado a otras muchas empresas por ofrecer certificados de su parte, sin regulación de ningun tipo. Obtener un certificado de *VeriSign* o de una de las empresas autorizadas es como si respondieran por nosotros. Generalmente, no emitirán un certificado hasta que verifiquen la información incluida en él, bien por vía telefónica o bien por otro medio de documentación en papel, como un contrato corporativo en el que se pide autenticación en persona. Cuando han "certificado" que nuestra información es correcta, cogen esta información, incluyendo los URL que vamos a utilizar para el certificado y la "firman" digitalmente cifrándola con su clave privada. Después, un servidor Web o cualquier programa podrán utilizar este certificado.

Las entidades de certificación descentralizan su misión en entidades de certificación locales en las que confian. En Cataluña esta entidad es la Agencia Catalana de certificaciones, CATCERT.

En su página web podemos encontrar más información: http://www.catcert.net

Cuando los usuarios externos reciben datos, como una página web del servidor, y adjunta un certificado, pueden utilizar la criptografía de clave pública para descifrar el certificado y verificar nuestra identidad. Se utilizan principalmente en los sitios de comercio electrónico, pero también se utilizan en cualquier otra forma de comunicación. Programas como SSH y Nessus pueden utilizar certificados para la autenticación. Las VPN o las redes inalámbricas también pueden utilizar certificados para la autenticación en lugar de contraseñas.

11.2.4. Protocolos de cifrado

Un hecho bien conocido es que el protocolo IP tal y como se diseño originalmente no era muy seguro. La versión 4 de IP (IPv4), utilizado por casi todo el mundo de la comunicación con IP, no proporciona ningún tipo de autenticación ni confidencialidad. Las cargas útiles del paquete se envían al descubierto y los encabezados del paquete se pueden modificar fácilmente ya que no se verifican en el destino. Muchos ataques de Internet se basan en esta inseguridad básica de la infraestructura de Internet. Para proporcionar autenticación y confidencialidad a través del cifrado, se ha desarrollado un nuevo estándar IP denominado IPv6. Además, amplía el espacio de dirección IP utilizando una dirección de 128 bits en lugar de 32 bits y mejora además otros elementos.

La implantación completa del estándar IPv6 requiere actualizaciones de hardware a amplia escala, por lo que el despliegue de IPv6 esta siendo bastante lento. Mientras no se acabe de implantar existen una serie de protocolos que nos permiten tener una cierta seguridad.

IPsec

Para solventar los problemas de IPv4 se desarrollo una implantación de seguridad para IP, denominada IPsec, que no requería cambios importantes en el esquema del direccionamiento. Los suministradores de hardware se aferraron a ello, convirtiéndose IPsec poco a poco en un estándar de hecho para crear VPNs en Internet.

No es un algoritmo de cifrado específico, sino una estructura para cifrar y verificar paquetes dentro del protocolo IP. Puede utilizar diversos algoritmos y puede implantarse total o parcialmente. Para codificar el contenido del paquete se utiliza una combinación de clave pública y privada y además los hash añaden

también autenticación. Esta función se denomina encabezado de autenticación (AH, Autentication Header). Con AH, un hash se crea a partir del encabezado IP y éste pasa adelante. Cuando el paquete llega a su destino, se crea un nuevo hash a partir de cada encabezado. Si no es comparable al enviado, sabrá que el encabezado se ha alterado de alguna manera durante el tránsito, lo que nos proporciona un alto nivel de garantía de que el paquete proviene de donde dice. Podemos elegir descifrar la carga útil pero no ejecutar un AH ya que puede ralentizar el procesamiento. AH también puede estropearse en algunos entornos con NAT o cortafuegos.

Existen otros dos modos de operación diferentes en los que podemos ejecutar IPsec: en modo túnel o en modo transporte.

En modo túnel, todo el paquete (encabezados incluidos) se encapsula y se cifra, se coloca en otro paquete y se remite al procesador VPN central. Los extremos finales descifran los paquetes y después los envían al IP correcto. Una de las ventajas de utilizar este método es que los usuarios externos pueden saber incluso cuál es el destino final del paquete cifrado. Otra ventaja es que VPN puede controlarse y administrarse desde pocos puntos centrales. El inconveniente es que requiere un hardware dedicado en ambos extremos para abrir el túnel.

En modo de transporte sólo se cifran las cargas útiles del paquete; los encabezados se envían intactos, lo que produce un despliegue más facil y requiere menos infraestructura. Podemos seguir ejecutando AH cuando utilicemos el modo de transporte y verificar la dirección de origen de los paquetes.

PPTP: Protocolo de túnel punto a punto

El protocolo PPTP (Point-to-Point Tunneling Protocol) es un estándar desarrollado por Microsoft, 3Com y otras grandes empresas que proporcionan cifrado. En PPTP se han descubierto algunos fallos importantes que limitan su aceptación. Cuando Microsoft implemento IPsec en Windows 2000, parecía una admisión tácita que IPsec había ganado como nuevo estándar de cifrado. Sin embargo, PPTP sigue siendo un protocolo útil y económico para configurar VPN entre los PC más antiguos de Windows.

L2TP: Protocolo de túnel de dos capas

El protocolo L2TP (Layer Two Tunneling Protocol) es otro protocolo desarrollado para la industria y firmado por Microsoft y Cisco. Aunque se utiliza frecuentemente en dispositivos de cifrado basados en hardware, su uso en software es relativamente limitado.

SSL: Capa segura de Sockets

El protocolo SSL (Secure Socket Layer) se diseñó específicamente para su uso en la Web (ApacheSSL que explico en el proyecto lo contempla), aunque puede utilizarse en cualquier tipo de comunicación TCP. Originalmente lo diseño Netscape para que su explorador ayudase a la simulación de comercio electrónico. SSL proporciona cifrado de datos, autenticación en ambos extremos e integridad de mensajes utilizando certificados.

Normalmente, SSL se utiliza cuando se realiza una conexión a un servidor web para que sepa que la información que enviamos se protege a lo largo del trayecto. La mayoría de las personas ni siquiera se dan cuenta de que SSL se está ejecutando en segundo plano. Normalmente sólo autentica un extremo, la parte del servidor, ya que la mayoría de usuarios finales no tienen certificados.

11.2.5. OpenPGP: Aplicación de cifrado

El estandar PGP (Pretty Good Privacy o privacidad bastante buena), en el que esta basado OpenPGP, se creo para proteger la información de los usuarios ante miradas indiscretas.

Historia del PGP

Phil Zimmerman es un programador muy implicado en los derechos humanos. Le preocupaba que el uso creciente de los ordenadores y de las redes de comunicación facilitase a las agencias de seguridad estatales de regímenes represivos la interceptación y recopilación de información sobre los disidentes. Phil quería escribir algún software que ayudase a dichas personas a mantener su información privada y segura frente

a los brutales regimenes que los controlaban. Este software podía salvar literalmente la vida de algunas personas. Tampoco creía que su propio gobierno no observara sus datos personales cuando se desplazaban por redes interconectadas. Sabía lo fácil que podía ser para el gobierno crear sistemas para buscar cada línea de los mensajes de correo electrónico para determinadas palabras clave. Deseaba proporcionar a las personas una forma de proteger y garantizar su derecho constitucional a la privacidad.

Este software lo denominó Pretty Good Privacy (PGP), algo así como una Privacidad bastante buena, ya que creía que hacía una buena labor a la hora de proteger los datos ante los servicios de inteligencia de los países más pequeños. Sin embargo, la agencia de la información de los Estados Unidos, NSA, no lo veía de esa forma. Zimmerman fue investigado por infringir las leyes federales de exportación de munición por permitir que su software se descargase fuera del país. Originalmente, Phil pretendía buscar una empresa que vendiera su innovación. Sin embargo, cuando el gobierno inició su persecución, distribuyó libremente el software por Internet para que se distribuyese por todas partes. Posteriormente formó una empresa para vender las versiones comerciales del software pero existen implantaciones de libre distribución de PGP por todo Internet. Algunas de éstas son mas populares que otras y algunas son aplicaciones que tienen funciones específicas como el cifrado de mensajes de correo electrónico. Se pueden encontrar una lista de todas estas implementaciones de PGP en *http://www.cypherspace.org/openpgp/*.

GnuPG: GNU Privacy Guard

En el servidor he utilizado GnuPG, una de las implementaciones bajo licencia GPL basada en el estandar OpenPGP.

La gran ventaja de la versión Gnu a la versión PGP comercial es que se encuentra bajo licencia GPL y por lo tanto podemos utilizarla para cualquier aplicación, comercial o personal, asi como ampliarla o insertarla como queramos. El inconveniente es que se trata de una herramienta de línea de comandos, por lo que no tiene los bonitos complementos que ofrece la versión comercial de PGP. No obstante hay que tener cuidado, GnuPG probablemente no sea la mejor opción para usuarios no técnicos, a no ser que añadamos su propia interfaz (Gnome dispone de una).

Para comprobar la versión:
`$gpg --version`

Si no lo tenemos instalado, hay que ejecutar algunas sentencias apt:
`#apt-get instal gnupg`, ... Programa GnuPG
`#apt-get instal gpgp`, ... interfaz Gnome para GnuPG

Crear pares de claves

- Si ya las tiene creadas y quiere importarlas, `#gpg --import path/filename`. Si tiene separadas la clave pública y privada, el formato de archivo suele ser *pubring.pkr* y *secring.skr*
- Si se han de crear:
  1. Escribimos: `#gpg --gen-key` y seguimos las instrucciones de pantalla
  2. GnuPG pide el tamaño de bits de sus claves. El predeterminado es 1.024, que generalmente se considera suficiente para una criptografía sólida de clave pública. Podemos aumentar el tamaño hasta 2.048 para una seguridad mucho mas fuerte.
  3. Generalmente no se desea que las claves expiren, pero si tiene un caso especial en el que sólo utilizará esta clave durante un tiempo limitado, se puede establecer una fecha de finalización.
  4. Por último se pide que se introduzca una contraseña que sirva de frase de paso.

Atención: Es muy importante conservar copias de seguridad del par de claves en un lugar seguro, en caso de perderlas, los datos cifrados se no se podrían recuperar.

`$gpg --list-keys`, ... nos dara la lista de claves instaladas en el sistema.

En el *home* del usuario se crea un directorio oculto *.gnupg* donde se almacenarán los archivos.

Crear un certificado de revocación:

Una vez creadas las claves, también podemos crear un certificado de revocación que se utiliza si perdemos nuestras claves o si alguien obtiene acceso a nuestra clave privada. Después podemos utilizar este certificado para revocar nuestra clave de los servidores públicos. Podemos seguir descifrando mensajes recibidos utilizando la clave pública antigua (suponiendo que no la hayamos perdido) pero nadie podrá descifrar ningún mensaje con las claves públicas incorrectas. Para crear un certificado:

```
$gpg --output revoke.asc --gen-revoke user
```

Donde se debe reemplazar *user* con la frase secreta de dicho usuario, asignada cuando generamos el par de claves. Este certificado hay que eliminarlo del disco y guardarlo en lugar seguro, ya que si alguien se hace con él, también podría hacerse con el certificado de revocación.

Publicar la clave pública

Se puede colocar la clave pública en un servidor de claves para que cualquiera pueda encontrarla fácilmente y enviar mensajes. Para ello es necesario ejecutar el siguiente comando:

```
$gpg --keyserver server --send-keys user
```

Donde se debe reemplazar *server* con el nombre de un servidor de claves públicas y *user* con la dirección de correo electrónico de la clave que desea publicar. Puede utilizar cualquier servidor de claves públicas PGP ya que todos se sincronizan con frecuencia. Existen muchos servidores de claves públicas, como por ejemplo: pgp.mit.edu, certserver.pgp.com y usa.keyserver.net.

Cifrar archivos con GnuPG

Para cifrar un archivo se utiliza el comando:

```
$gpg --output file.gpg --encrypt --recipient friend@example.com file.doc
```

Reemplanzando *file.gpg* con el nombre de archivo resultante deseado, *friend@example.com* con la dirección de correo electrónico del usuario al que está realizando el envío y *file.doc* por el arcivo que se desea cifrar. Tenga en cuenta que tiene que tener la clave pública del destinatario en su repositorio para poder hacerlo.

También se puede utilizar GnuPG para cifrar archivos con una criptografía simétrica, que puede utilizar para sus archivos locales que desea proteger o para alguien del que no tiene su clave pública. Para ello, utilice el comando:

```
$gpg --output file.gpg --symmetric file.doc
```

Reemplazado *file.gpg* con el archivo de salida deseado y *file.doc* con el nombre del archivo que desea cifrar.

Descifrar archivos

Para utilizar GnuPG para descifrar archivos recibidos, utilizamos el siguiente comando:

```
$gpg --output file.doc --decrypt file.gpg
```

Reemplazando *file.doc* con el nombre del archivo resultante deseado y siendo *file.gpg* el archivo cifrado. Tenemos que tener la clave pública del usuario para el que se ha realizado el cifrado en nuestro repositorio. Un mensaje solicitará la frase de contraseña y, una vez introducida correctamente, GnuPG producirá el archivo descifrado.

Firmar archivos

Tal y como he mencionado anteriormente, otro uso de GnuPG y PGP es firmar documentos para verificar su integridad, algo que podemos hacer con el siguiente comando:

```
$gpg --output signed.doc --sign unsigned.doc
```

Siendo *signed.doc* el nombre de archivo de salida resultante deseado y *unsigned.doc* el archivo a firmar. Este comando firma y cifra el documento y después procesa el archivo de salida *signed.doc*. Cuando se descifra, GnuPG también verificará el documento.

Para verificar el archivo se utiliza el comando:

```
$gpg --verify signed.doc
```

Siendo *signed.doc* el archivo cifrado que desea verificar. También podemos crear firmas separadas del archivo para poder acceder a usuarios sin GnuPG e incluir la firma. Éstos son los comandos para conseguirlo:

```
$gpg --clearsign file.doc
```

Crea un apéndice de texto al archivo con la firma. Si no desea alterar el archivo, puede crear un archivo de firma separado, para dejar almacenada la firma o mandarla por separado, con el comando:

```
$gpg --output sig.doc --detached-sig file.doc
```

Web de modelo de confianza de PGP/GnuPG

Tal y como hemos mencionado anteriormente, en lugar de utilizar un sistema de confianza jerárquico como los certificados digitales y su autoridad de certificados central, PGP y GnuPG utilizan una web de modelo de confianza. Al firmar las claves de las personas que conoce, puede verificar que sus claves son de confianza. Y si firman las claves de otras personas que no conoce directamente, se crea una cadena de confianza. El modelo se basa en la idea de que "cualquier amigo suyo es un amigo mío". En realidad este modelo no funciona perfectamente; en algún lugar en la parte inferior de la cadena podría haber una manzana podrida, pero la idea que se esconde detrás del sistema es que se propaga orgánicamente y no requiere ninguna estructura. Debido a ello puede desmantelarse o formarse fácilmente a gran escala. La forma de establecer esta web de confianza es firmar las claves de las personas y dejar que ellos firmen la suya propia.

Firmar claves y administrar claves de confianza

En GnuPG, firmamos y administramos claves de confianza entrando en el modo de edición de la clave con el siguiente comando:

```
gpg --edit-key friend@example.org
```

Donde *friend@example.org* coincide con la dirección de correo electrónico de la clave que desea firmar o administrar, tiene que ser una de la claves que tenemos en el repositorio. Este comando imprime información básica sobre la clave. Dentro de este modo escriba *fpr* para imprimir la huella dactilar de la clave. Igual que las personas, la huella dactilar de la clave es un identificador específico para dicha clave. Asegúrese de que se trata de la clave de la persona en concreto comparándola con dicha persona por teléfono o por cualquier otro medio de comunicación. También puede comprobar si alguien más ha firmado esta clave escribiendo *check*. Este comando imprime una lista de otros firmantes de esta clave y puede ayudarla a decidir la validez de la misma.

Cuando se quiere asegurar de que se trata de la clave de una persona en concreto, escriba *sign*. Este comando firma la clave de dicha persona para que cualquiera que la esté buscando sepa que confía en ella. En este modo también puede editar los niveles de las diferentes claves en su repositorio. Se introduce este modo dentro del modo de edición de la clave escribiendo *trust*.

Así se muestra el siguiente menú:

```
1=Don't know (No la conozco).
2=I do NOT trust (No confio en ella).
3=I trust marginally (Confio en ella un poco).
4=I trust fully (confio en ella plenamente).
d=Please show me more information (Mostrar mas detalles).
m=Back to the main menu (Volver al menu principal).
```

Al escoger uno de los elementos, dicha clave se marcará como realizada por nosotros. Ésta es otra forma de poderse comunicar con otros a cerca de los usuarios que tienen su nivel más alto de confianza y cuáles son usuarios en los que confía poco.

PGP y GnuPG son extraordinarios para cifrar archivos. Sin embargo, ¿y si desea cifrar todas las comunicaciones entre dos puntos? PGP no es en realidad viable para esta función, esto es realizado por el SSH, explicado en la sección anterior.

Existen muchísimas más opciones que se pueden consultar con: `$man gpg`

Capítulo 12

Herramientas de seguridad

12.1. Herramientas básicas

Existen varios comandos de sistema operativo que utilizaremos con frecuencia en nuestra labor de asegurar la seguridad. No son programas de seguridad completamente independientes, sino utilidades del sistema operativo que se pueden utilizar para generar información de seguridad.

12.1.1. Ping

Ping (Packet Internet Groper, que se pude traducir como buscador de paquetes en Internet) es una herramienta de diagnóstico TCP/IP. Muchos creen que el ping es como un radar submarino: un ping sale, rebota en un destino y vuelve. Aunque se trate de una buena analogía general, no describe exactamente lo que sucede cuando incluimos un ping en una máquina. Los ping utilizan un protocolo de red denominado ICMP (Internet Control Message Protocol, o Protocolo de mensajes de control de Internet). Estos mensajes se utilizan para enviar información sobre redes. Ping utiliza los tipos de mensaje ICMP 8 y 0, conocidos también como *Solicitud de eco* y *Contestación de eco* respectivamente. Cuando utilizamos el comando ping, la máquina envía una solicitud de eco a otra máquina. Si se puede acceder a la máquina que se encuentra en el otro extremo y se ejecuta una pila TCP conforme, responderá con una contestación de eco.

Básicamente, las comunicaciones en un ping tienen la siguiente apariencia:

- El sistema A envía un ping al sistema B: solicitud de eco, "¿Estás ahí?"
- El sistema B recibe la solicitud de eco y envía una contestación de eco: "Si, estoy aquí"

En una sesión ping típica, este proceso se repite varias veces para comprobar si la máquina de destino o la red están bajando paquetes. Tambieén se puede utilizar para determinar la latencia, el tiempo que tardan los paquetes entre dos puntos.

También podemos obtener estos otros tipos de mensajes ICMP cuando mandamos un ping. Cada uno tiene us propio significado.

- Red inalcanzable
- Anfitrión inalcanzable

Con un ping podemos saber algo más sobre un anfitrión que si simplemente está activo o no. La forma en que una máquina responde a un ping, normalmente, identifica el sistema operativo que está ejecutando. También podemos utilizar ping para generar una solicitud de búsqueda DNS, que nos proporciona el nombre del anfitrión de destino (si lo tiene), que a veces puede decirnos si esta máquina es un servidor, un enrutador o quizá una conexión telefónica o conexión de ancho de banda. Podemos mandar un ping a una dirección IP o a un nombre de dominio.

Cuadro 12.1: Opciones del comando ping

Opción	Descripción
`-c count`	Establece un número de pings deteminado, por defecto es infinito
`-f`	Flujo de los ping. Envía tantos paquetes como puede, tan rápido como puede. Útil para comprobar si un anfitrión esta bajando paquetes porque mostrará gráficamente a cuántos ping responde. Hay que tener cuidado con esta opción porque puede producir la caida de la red por DoS (denegación de servicio)
`-n`	No ejecuta DNS en la dirección IP. Puede aumentar la velocidad de una respuesta y cancelar los problemas de DNS cuando existen problemas de diagnóstico de la red
`-s size`	Envía paquetes de longitud *size*. Útil para probar cómo manipula los paquetes grandes una máquina o un enrutador. Irregularmente, los paquetes grandes se utilizan en ataques de denegación de servicios para hacer caer una máquina o desbordarla
`-p pattern`	Envía un patrón específico en el paquete ICMP de carga útil. También es útil para probar cómo responde una máquina a un estímulo ICMP inusual

La tabla 12.1 incluye los modificadores y las opciones adicionales para el comando ping, que pueden resultar útiles.

Un ejemplo de este comando podría ser el siguiente:

```
# ping www.fib.upc.es
PING www.fib.upc.es (147.83.41.7): 56 data bytes
64 bytes from 147.83.41.7: icmp_seq=0 ttl=241 time=66.3 ms
64 bytes from 147.83.41.7: icmp_seq=1 ttl=241 time=130.4 ms
64 bytes from 147.83.41.7: icmp_seq=2 ttl=241 time=103.8 ms
64 bytes from 147.83.41.7: icmp_seq=3 ttl=241 time=62.0 ms
64 bytes from 147.83.41.7: icmp_seq=4 ttl=241 time=76.8 ms
64 bytes from 147.83.41.7: icmp_seq=5 ttl=241 time=61.5 ms

--- www.fib.upc.es ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 61.5/83.4/130.4 ms
```

12.1.2. Traceroute

Si no lo tenemos instalado basta con hacer:
`#apt-get install traceroute`

Este comando es similar a ping pero proporciona más información sobre el anfitrión remoto. Básicamente, los pings que rastrean rutas (tracerotue) son anfitriones, pero cuando envían fuera el primer paquete, establecen la configuración TTL (tiempo de vida) del paquete en uno. Esta configuración controla la cantidad de puntos de conexión en el ordenador de la red que obtendrá antes de morir por lo que el primer paquete sólo irá al primer enrutador o máquina más allá de la nuestra en Internet y después devolverá un mensaje indicando que el paquete ha "expirado". A continuación el siguiente paquete se establece con un TTL de 2, y así sucesivamente hasta llegar a nuestro objetivo, mostrándonos el trazado virtual (la ruta) que siguen los paquetes. Se resuelve el nombre de cada anfitrión encontrado a lo largo del camino para que podamos ver cómo cruza Internet el tráfico. Puede ser muy interesante comprobar cómo un paquete pasa por diferentes paises antes de llegar a su destino una fracción de segundo después y a veces, por caminos impensables y lejanos.

Esta herramienta es muy práctica cuando estamos intentando localizar el origen o la ubicación de un intruso que hemos encontrado en nuestro logs o alertas. Podemos rastrear la ruta de la dirección IP y saber varias cosas sobre dicha dirección. La salida puede indicarnos si se trata de un usuario doméstico

o de un usuario que se encuentra dentro de una empresa, cuál es su ISP (para poder enviarle una queja sobre el abuso), qué tipo de servicio tiene y lo rápido que es y dónde se encuentra geográficamente (a veces depende de la capacidad de descripción de los puntos intermedios).

Para ver como funciona podemos ejecutar el siguiente ejemplo:

```
# traceroute www.fib.upc.es
traceroute to www.fib.upc.es (147.83.41.7), 30 hops max, 38 byte packets
1   (192.168.0.1)  0.480 ms  0.459 ms  0.408 ms
2  213.0.184.252 (213.0.184.252)  46.732 ms  47.016 ms  116.462 ms
3  213.0.190.22 (213.0.190.22)  48.833 ms  51.344 ms  47.850 ms
4  tbvia1-tbest1-1.nuria.telefonica-data.net (213.0.248.146)  47.108 ms  45.986 ms
5  213.0.254.242 (213.0.254.242)  47.335 ms  52.532 ms  51.392 ms
6  montseny-catnix.catnix.net (193.242.98.2)  61.530 ms  66.452 ms  61.101 ms
7  upc-anella.cesca.es (84.88.18.18)  63.990 ms  129.435 ms  64.099 ms
8  * * *
```

Como se puede observar, la lectura de traceroute es mas un arte que una ciencia, pero con el tiempo se aprende a reconocer mejor el significado de las abreviaturas.

Traceroute nos ofrece mucha información para el seguimiento de una IP, si es el origen de una intrusión o un ataque. Si identificamos el ISP, con el sitio web de la empresa podemos encontrar un número de telefono o una dirección de correo y quejarnos. Los ISPs habitualmente finalizan su contrato de suministro con el cliente malintencionado. También podemos utilizar el comando whois, para buscar contactos técnicos específicos para la empresa y organización.

12.1.3. Whois

El comando whois es útil para intentar localizar el contacto de alguien que está causando problemas en nuestra red. Este comando consulta los servidores de nombre de dominio principales y devuelve toda la información que tiene el registrador de nombres que le corresponda. Con esto podemos averiguar quién es el propietario de un dominio.

Este comando es útil para ataques que provienen tanto de dentro de las redes de empresas como de los ISP. De cualquier modo, podemos averiguar quién es la persona responsable de dicha red y comunicarle nuestro problema. Esta solución no siempre es muy práctica, pero por lo menos podemos probar.

Su sintaxis es:

`$whois domain-name.com`, . . . donde *domain-name.com* es el nombre del dominio sobre el que estamos buscando información.

Podemos observar esto en el siguiente ejemplo:

```
$whois www.google.com

Whois Server version 1.3

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

   Server Name: WWW.GOOGLE.COM.TR
   Registrar: TUCOWS INC.
   Whois Server: whois.opensrs.net
   Referral URL: http://domainhelp.tucows.com

   Server Name: WWW.GOOGLE.COM.MX
   Registrar: ENOM, INC.
   Whois Server: whois.enom.com
   Referral URL: http://www.enom.com

   Server Name: WWW.GOOGLE.COM.BR
   Registrar: ENOM, INC.
   Whois Server: whois.enom.com
   Referral URL: http://www.enom.com

   Server Name: WWW.GOOGLE.COM.AU
   Registrar: MELBOURNE IT, LTD. D/B/A INTERNET NAMES WORLDWIDE
   Whois Server: whois.melbourneit.com
   Referral URL: http://www.melbourneit.com

>>> Last update of whois database: Wed, 25 May 2005 08:39:49 EDT <<<
```

También nos puede pasar lo siguiente:

```
$whois www.fib.upc.es
Este TLD no dispone de servidor whois, pero puede acceder a la informacion de whois en
https://www.nic.es/ingles/
```

Con lo que, la única solución es buscar a mano en esa dirección. Solamente los dominios *.com*, *.net* y *.edu* se encuentran incluidos en whois.

El comando whois normalmente muestra una lista de direcciones de correo electrónico, direcciones de correo postal y, a veces, números telefónicos. Nos dice cuándo se creó el dominio y si se han hecho cambios recientes en sus listados whois. Tambien muestra a los servidores de nombre de dominio responsables de ese nombre de dominio. Se puede ampliar más esta información con el siguiente comando: *dig*.

Si administra dominios propios, debe asegurarse de que su listado whois está actualizado y es tan genérico como pueda serlo. Al colocar direcciones de correo electrónico y nombre reales en los campos de información, estamos proporcionando información que alguien del exterior puede aprovechar, ya sea para una labor social como para atacar nuestros sistemas. Es mejor utilizar direcciones de correo electrónico genéricas, dejando que los responsables reciban los mensajes enviados a esas direcciones de correo electrónico y evitar proporcionar una información valiosa sobre la estructura de nuestra organización técnica.

12.1.4. Dig

El comando dig consulta en el servidor de nombres determinada información sobre un dominio. Dig es una versión actualizada del comando *nslookup*, que ha quedado desfasado. Podemos utilizarlo para determinar los nombre de máquinas utilizados en una red, qué direcciones IP se unen a dichas máquinas, cuál es su servidor de correo y otro tipo de información útil.

La sintaxis general es:
`$dig @ server domain type`

Donde *server* es el servidor DNS al que deseamos consultar, *domain* es el dominio sobre el que estamos preguntando y *type* es el tipo de información deseada.

Como ejemplo podemos poner una dirección que viene en el manual:

```
$dig www.isc.org

; <<>> DiG 9.2.4 <<>> www.isc.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25173
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 3

;; QUESTION SECTION:
;www.isc.org.                   IN      A

;; ANSWER SECTION:
www.isc.org.            600     IN      A       204.152.184.88

;; AUTHORITY SECTION:
isc.org.                3599    IN      NS      ns-ext.isc.org.
isc.org.                3599    IN      NS      ns-ext.lga1.isc.org.
isc.org.                3599    IN      NS      ns-ext.nrt1.isc.org.
isc.org.                3599    IN      NS      ns-ext.sth1.isc.org.

;; ADDITIONAL SECTION:
ns-ext.lga1.isc.org.    3599    IN      A       192.228.91.19
ns-ext.nrt1.isc.org.    3599    IN      A       192.228.90.19
ns-ext.sth1.isc.org.    3599    IN      A       192.228.89.19

;; Query time: 696 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Wed May 25 21:30:18 2005
;; MSG SIZE  rcvd: 192
```

En la tabla 12.2 podemos encontrar las opciones más usuales del comando dig.

Cuadro 12.2: Opciones del comando dig

Opción	Descripción
AXFR	Intenta obtener todo el archivo del archivo del dominio o de la "zona". Actualmente algunos servidores se han configurado para no admitir transferencias de archivos de zona, por lo que puede que tengamos que preguntar por registros específicos.
A	Devuelve cualquier registro "A". Los registros "A" son nombres de anfitrión individuales sobre la red
MX	Devuelve el nombre del anfitrión de correo registrado para dicho dominio. Es útil para contactar con un administrador
CNAME	Devuelve cualquier anfitrión *CNAMED*, conocidos como alias
ANY	Devuelve cualquier información que puede generarse en el dominio. Algunas veces funciona cuando falla AXFR

12.1.5. Finger

Finger es un antiguo comando Unix que ya no se utiliza pero que se sigue ejecutando en muchas máquinas como servicio heredado. Originalmente se diseñó cuando Internet era un lugar cómodo y los usuarios permitían que otras personas al otro lado del mundo conociesen sus datos.

La mayoría de los administradores lo eliminan de sus sistemas porque es una fuente de brechas de seguridad, pero todavía existen muchos enrutadores que lo incluyen y algunos Unix lo mantienen por defecto. Además a esto se une que muchos administradores se les olvida desinstalarlo o no saben cómo hacerlo.

El comando *finger* permite consultar, al sistema remoto, acerca de información de sus usuarios. La sintaxis sería la siguiente:

```
$finger user@hotname.example.com
```

También podemos especificar una dirección IP como dominio. El resultado del finger, podría ser usada por una persona malintencionada para conseguir información más relevante mediante ingeniería social.

Otro uso de finger es enviar el comando sin un nombre de usuario. Así se genera una lista de todos los usuarios conectados actualmente. Con esto podemos saber quién está conectado y sus nombres reales e incluso podemos saber si están inactivos y durante cuanto tiempo. Por último presenta una lista de las estaciones y de donde provienen (si son locales o remotas), un usuario malicioso puede intentar secuestrar una de esas sesiones inactivas. Los resultados del comando los podemos ver en el siguiente ejemplo:

```
$finger
Login    Name                    Tty      Idle  Login Time   Office     Office Phone
josan    Jose Antonio Escartin  *:0              May 26 12:18
josan    Jose Antonio Escartin   pts/1           May 26 12:34 (:0.0)
```

Y después se puede utilizar para para averiguar más sobre un usuario:

```
$finger josan
Login: josan                            Name: Jose Antonio Escartin Vigo
Directory: /home/josan                  Shell: /bin/bash
On since Thu May 26 12:18 (CEST) on :0 (messages off)
On since Thu May 26 12:34 (CEST) on pts/1 from :0.0
No mail.
No Plan.
```

En este caso no hay demasiada información, pero otras veces podemos encontrar su correo electrónico, su plan de trabajo e incluso proyectos en los que este trabajando actualmente.

También podemos hacer consultas sobre todos los que esten conectados con la siguiente opción:

```
$finger -l
```

12.2. Firewall o cortafuegos

Forman la primera línea de defensa frente a los intrusos que quieren entrar en nuestra red corporativa. Sin embargo, debido a la complejidad creciente y a la sofisticación de los atacantes, pueden ser unos mecanismos de defensa insuficientes si no se han configurado correctamente. Una línea de configuración errónea puede negar la protección que ofrece un cortafuegos. Un administrador muy ocupado que esté intentando establecer el acceso para los empleados errará normalmente más en la parte de acceso en lugar de realizar ahí un mejor esfuerzo para configurar la seguridad. Se necesita mucha revisión y paciencia a la hora de establecer las reglas.

Los cortafuegos, se sitúan en la parte en la parte superior del sistema operativo y por lo tanto, pueden ser vulnerables a todos los ataques normales a nivel de sistema operativo. Muchos cortafuegos utilizan un servidor web como interfaz con otros usuarios y así pueden aprovecharse también de las brechas en las interfaces web. Asegurar estas defensas de primera línea es crítico y debe ser una de nuestras primeras prioridades.

Por otra parte un cortafuegos solo nos protege de los ataques exteriores, contra ataques interiores no tiene nada que hacer. Debemos de asegurarnos de que nuestros sistemas están vigilados y no depender del cortafuegos para toda la seguridad de nuestra red.

12.2.1. Políticas de seguridad

En algún momento, preferiblemente antes de instalar el cortafuegos, debemos comentar por escrito su proceso de actuación. Esto nos será muy útil para planificar la instalación y configuración. Este plan documenta los procesos y procedimientos subyacentes para asegurarnos de que obtenemos un beneficio. La instalación de un cortafuegos está muy bien, pero sin instalar los procesos apropiados, puede que no esté ofreciendo a la organización la seguridad prometida.

Los siguientes pasos perfilan un proceso para la implantación y funcionamiento de un cortafuegos.

- Desarrollar una política sobre el uso de la red
- Trazar un mapa de los servicios internos y externos necesarios
- Convertir la política sobre uso de la red y los servicios necesarios en reglas para el cortafuegos
- Implantar y probar la funcionalidad y la seguridad. Después de esto podemos activar el cortafuegos y sentarnos a esperar las quejas.
- Revisar y probar las reglas del cortafuegos periódicamente.

Diseñar y utilizar un proceso como éste nos ayudará a garantizar que obtenemos mucho más de la implantación de nuestro cortafuegos.

12.2.2. Modos de configuración

Existen dos formas de configurar un cortafuegos:

- Permitir todo y bloquear lo que no deseemos
- Denegar todo y añadir lo permitido

El método habitual, usado por la gran mayoría de administradores es empezar con “denegar todo” y después añadir lo que deseamos permitir a los usuarios. Automáticamente bloqueamos todo el tráfico, a no ser que se admita en la configuración específicamente.

Para la mayoría de los sitios, la solución “denegar todo” es mucho mas seguro. Sin embargo, sólo porque optemos por esta solución no significa que nuestra red sea totalmente segura. Los ataques pueden provenir a través de cualquier brecha que hayamos creado, como la web y el correo electrónico.

12.2.3. IPTables

Esta sección describe cómo se configura un cortafuegos con IPTables, que es la utilidad integrada en la mayoría de los sistema Linux 2.4 y posteriores. Esta utilidad nos permite crear un cortafuegos empleando comandos de nuestro sistema operativo.

Es una herramienta muy eficaz, pero compleja, y normalmente se recomienda para usuarios que estén familiarizados con los cortafuegos y con el arte de configurarlos. Si es nuestro primer cortafuegos, es mejor utilizar una de las herramientas de configuración automática disponibles para crear la configuración, al menos al principio. Estas herramientas utilizan IPTables para crear un cortafuegos utilizando nuestras entradas. Sin embargo, es recomendable tener un conocimiento básico de lo que está sucediendo "bajo cubierta" con IPTables antes de empezar a configurar con una de las herramientas gráficas.

Instalar IPTables

La mayoría de los sistemas Linux con kernel 2.4 o superior tendrán integrado IPTables, por lo que no es necesario instalar ningún programa adicional, el servicio lo proporciona el propio kernel.

Para comprobar si lo tenemos instalado hay que ejecutar:
`$iptables -L`, ... debería mostrar una lista con el conjunto actual de reglas.

Si tenemos problemas lo más probable es que no tengamos habilitado IPTables en el kernel, tenemos que recompilar el kernel.

Módulos del kernel para IPTables

Para poder usar IPTables es necesario tener instalado en el kernel el *netfilter* y cargados una serie de módulos, depende para que usemos IPTables necesitaremos más o menos.

Las opciones de IPTables se encuentran en: *Networking-suport* -> *Networking-options* -> *Network-packet-filtering.*

1. Módulos básicos
   - CONFIG_NETFILTER
   - CONFIG_PACKET
   - CONFIG_IP_NF_CONNTRACK
   - CONFIG_IP_NF_FTP
2. Tabla *filter* (actúa como filtro)
   - CONFIG_IP_NF_IPTABLES
   - CONFIG_IP_NF_FILTER
   - CONFIG_IP_NF_MATCH_LIMIT
   - CONFIG_IP_NF_MATCH_MAC
   - CONFIG_IP_NF_MATCH_MARK
   - CONFIG_IP_NF_MATCH_MULTIPORT
   - CONFIG_IP_NF_MATCH_TOS
   - CONFIG_IP_NF_MATCH_TCTPMSS
   - CONFIG_IP_NF_MATCH_STATE
   - CONFIG_IP_NF_TARGET_REJECT

3. Tabla *Nat* (actúa como router)
   - CONFIG_IP_NF_NAT
   - CONFIG_IP_NF_NAT_NEEDED
   - CONFIG_IP_NF_NAT_FTP
   - CONFIG_IP_NF_TARGET_MASQUERADE
   - CONFIG_IP_NF_TARGET_REDIRECT
4. Tabla *mangle* (altera paquetes especiales)
   - CONFIG_IP_NF_MANGLE
   - CONFIG_IP_NF_TARGET_TOS
   - CONFIG_IP_NF_TARGET_MARK
   - CONFIG_IP_NF_TARGET_LOG
   - CONFIG_IP_NF_TARGET_TCPMSS
5. Compatibilidad con versiones anteriores
   - CONFIG_IP_COMPAT_IPCHAINS
   - CONFIG_IP_COMPAT_IPFWADM

Utilizar IPTables

La idea que se esconde detrás de IPTables es crear canales de entradas y procesarlas de acuerdo con un conjunto de reglas (la configuración del cortafuegos) y enviarlas a continuación a canales de salida. En IPTables, estos canales se denominan tablas.

Las tablas básicas empleadas en IPTables son:

- Input: Tráfico que entra en la máquina
- Forward: Tráfico que pasa por la máquina
- Prerouting: Enrutamiento previo
- Postrouting: Enrutamiento posterior
- Output: Trafico que sale de la máquina

El formato que genera una declaración IPTables es:
`#iptables command rule-specification extensions`

Donde *command*, *rule-specification* y *extensions* son una o más opciones válidas. La tabla 12.3 incluye un resumen de las especificaciones de reglas y la tabla 12.4 de los comandos IPTables.

Existen otros comandos y opciones pero éstos son los más comunes. Para obtener una lista completa de listados, consulte el manual de IPTables escribiendo:
`$man iptables`

Cuadro 12.3: Especificaciones de reglas de IPTables

Regla	Descripción
`-s address/mask!port`	Especifica una determindada dirección de red origen a comparar. Para designar un rago de direcciones IP se usa la notación estándar de barra oblicua. También se pude especificar un número de puerto o un rango de números de puerto colocándolos después de un signo de exclamación de apertura
`-d address/mask!port`	Especifica una determindada dirección de red destino a comparar. Para designar un rago de direcciones IP se usa la notación estándar de barra oblicua. También se pude especificar un número de puerto o un rango de números de puerto colocándolos después de un signo de exclamación de apertura
`--sport`	Puerto de origen
`--dport`	Puerto de destino
`--tcp-flags`	Flags permitidos y flags activos. Los flags son:SYN ACK FIN RST URG PSH ALL NONE (hay que especificarlos separados por comas)
`-f`	Especifica paquetes fragmentados
`-m <modulo>`	Especifica un módulo de opciones especiales
`-m multiport`	Especifica que se usará el módulo de multipuertos
`-p protocol`	Especifica un determinado protocolo con el que se compara la regla. Los tipos de protocolo válidos son icmp, tcp, udp o todos (all)
`-i interface`	Especifica una interfaz de entrada
`-o interface`	Especifica una interfaz de salida
`-j target`	Indica lo que se tiene que hacer con el paqute si coincide con las especificaciones. Las opciones válidas para *target* son:
	DROP — Coloca el paquete sin ninguna acción posterior
	REJECT — Coloca el paquete y devuelve un paquete de error
	LOG — Registra el paquete en un de error
	MARK — Marca el paquete para una acción posterior
	TOS — Cambia el bit TOS (Tipo de servicio)
	MIRROR — Invierte las direcciones de origen y de destino y las envía de nuevo, básicamente "rebotándolas" de nuevo al origen
	SNAT — NAT estática. Esta opción se utiliza cuando se está realizando una Traducción de dirección de red (NAT, Network Address Translation). Convierte la dirección de origen en otro valor estático
	DNAT — NAT dinámica. Similar a la anterior pero usando un rango dinámico de direcciones IP
	MASQ — Enmascara la IP usando una IP pública
	REDIRECT — Redirecciona el paquete

Crear un cortafuegos IPTables

La mejor forma de ver como funciona es mediante un ejemplo, que situaremos dentro de un script ejecutable.

Se toman las siguientes premisas: Se supone que la LAN local es 192.168.0.1 - 192.168.0.254, que la interfaz *eth1* es la conexión LAN local y que la interfaz *eth0* es la conexión WAN o Internet.

1. Empieza eliminando cualquier regla existente con el comando *Flush*:

   ```
   iptables -F FORWARD
   ```

   Así eliminamos cualquier regla para la cadena FORWARD, que es el "conducto" principal para cualquier paquete que desea pasar por el cortafuegos.

Cuadro 12.4: Comandos IPTables

Comando	Descripción
`-A chain`	Añade una o más reglas al final de la declaración
`-I chain rulenum`	Inserta una cadena en la ubicación *rulenum*. Es útil cuando deseamos que una regla reemplace a las anteriores
`-D chain`	Elimina la cadena indicada
`-R chain rulenum`	Reemplaza la regla en la ubicación *rulenum* con la *chain* proporcionada
`-L`	Lista todas las reglas en la cadena actual
`-F`	Purga todas las reglas en la cadena actual, eliminando básicamente la configuración de nuestro cortafuegos. Es recomendable su uso cuando iniciamos una configuración para asegurarnos de que ninguna regla existente entrará en conflicto con una regla nueva
`-Z chain`	Pone a cero todas las cuentas de paquetes y bytes en la cadena denominada
`-N chain`	Crea una nueva cadena con el nombre de *chain*
`-X chain`	Elimina la cadena especificada. Si no se especifica ninguna cadena, se eliminan todas las cadenas
`-P chain policy`	Establece la política para la cadena especificada en *policy*

2. Eliminamos las otras cadenas:

```
iptables -F INPUT
iptables -F OUTPUT
```

   Así eliminamos cualquier regla en su máquina local y en su cadena de salida, también podiamos haber usado: `iptables -F`, afectando a todas las cadenas a la vez.

3. Inserta la declaración "denegar todo" estándar justo al principio:

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

4. Para aceptar paquetes fragmentados en IPTables, es necesario que se escriba lo siguiente explícitamente:

```
iptables -A FORWARD -f -j ACCEPT
```

5. Existen dos tipos de ataques comunes que debemos bloquear en seguida. Uno es el conocido como *spooofing*, que se produce cuando alguien falsifica los encabezados de los paquetes IP para que parezcan paquetes externos que tienen direcciones internas. Así, alguien puede enrutar hacia nuestra LAN incluso aunque tengamos idrecciones IP privadas. El otro tipo de ataque se lleva a cabo enviando una gran cantidad de paquetes a las direcciones LAN para sobrecargar la red. Este tipo de ataque se denomina ataque *smurf*, ataca sobre el protocolo de transmisión de archivos. Podemos bloquear este tipo de ataques con dos sencillas declaraciones.

```
iptables -A FORWARD -s 192.168.0.0/24 -i eth0 -j DROP
iptables -A FORWARD -p icmp -i eth0 -d 192.168.0.0/24 -j DROP
```

   La primera declaración rechaza cualquier paquete que provenga de la interfaz eth0 de Internet con la dirección interna 192.168.0.0/24. Por definición, ningún paquete debería provenir de una interfaz de no confianza con una dirección de fuente privada e interna. La segunda declaración retira cualquier paquete del protocolo ICMP que provenga de la dirección exterior a la interior.

   También se podrían evitar las respuestas a pings externos mediante:

```
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

6. Generalmente aceptaremos el tráfico entrante basado en conexiones iniciadas desde el interior, por ejemplo, alguien que explora una página Web. Siempre que la conexión esté en curso y se haya iniciado internamente, probablemente sea correcta. Sin embargo, podemos limitar el tipo de tráfico permitido. Supongamos que sólo deseamos permitir a los empleados el acceso a la Web y al correo electrónico. Podemos especificar los tipos de tráfico para permitir sólo el que esté en una conexión ya iniciada. Podemos saber si es una conexión existente comprobando si se ha establecido la parte ACK, es decir, si se ha producido la conexión TCP de tres vías. Las siguientes declaraciones permiten el tráfico web basado en este criterio.

```
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 192.168.0.0/24 --dports www,smtp
         --tcp-flags SYN,ACK ACK -j ACCEPT
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 192.168.0.0/24 --sports www,smtp
         --tcp-flags SYN,ACK ACK -j ACCEPT
```

   La declaración *--dports* indica que sólo se permite el correo electrónico y la Web y la declaración de indicadores *–tcp* indica que sólo deseamos paquetes con el campo ACK establecido

7. Para poder aceptar conexiones de entrada desde el exterior sólo en determinados puertos, como un mensaje correo electrónico entrante en nuestro servidor de correo, usamos este tipo de declaración:

```
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 192.168.0.0/24 --dports smtp
         --syn -j ACCEPT
```

   El indicador de múltiples puertos *-m* indica a IPTables que vamos a emitir una declaración de coincidencia para los puertos. La declaración *-syn* le indica que se permiten los paquetes SYN, lo que significa que se deben iniciar las conexiones TCP. Y el indicador *--dports* permite sólo el tráfico de correo SMTP.

8. Podemos permitir que los usuarios inicien conexiones de salida pero sólo en los protocolos que deseamos que usen. Aquí podemos evitar que los usuarios usen FTP y otros programas no esenciales. Las direcciones que contienen todo ceros son una abreviatura de “cualquier dirección”.

```
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 0.0.0.0 --dports www,smtp
         --syn -j ACCEPT
```

9. Necesitaremos permitir determinados paquetes UDP entrantes. UDP se usa para DNS y si lo bloqueamos, los usuarios no podrán resolver direcciones. Como no disponen de un estado como los paquetes TCP, no podemos fiarnos de la revisión de los indicadores SYN o ACK. Deseamos admitir UDP sólo en el puerto 53, por lo que especificaremos: *domain* (una variable integrada para el puerto 53), como único puerto admisible. Las declaraciones que debemos usar son:

```
iptables -A FORWARD -m multiport -p udp -i eth0 -d 192.168.0.0/24 --dports
         domain -j ACCEPT
iptables -A FORWARD -m multiport -p udp -i eth0 -s 192.168.0.0/24 --sports
         domain -j ACCEPT
iptables -A FORWARD -m multiport -p udp -i eth1 -d 0.0.0.0 --dports
         domain -j ACCEPT
iptables -A FORWARD -m multiport -p udp -i eth1 -s 0.0.0.0 --sports
         domain -j ACCEPT
```

   Las dos primeras declaraciones permiten los datagramas UDP entrantes y las otras dos declaraciones permiten las conexiones salientes.

10. También podemos especificarlos para los paquetes ICMP. Lo que pretendemos es permitir todo tipo de ICMP interno hacia el exterior, pero sólo determinados tipos como la contestación de eco hacia el interior, algo que podemos conseguir con las siguientes instrucciones:

```
iptables -A FORWARD -m multiport -p icmp -i eth0 -d 192.168.0.0/24
         --dports 0,3,11 -j ACCEPT
iptables -A FORWARD -m multiport -p icmp -i eth1 -d 0.0.0.0
         --dports 8,3,11 -j ACCEPT
```

    Es mas simple si lo controlamos a la entrada, haciendo un DROP de los 'echos de icmp', así estas dos instrucciones no son necesarias.

11. Por último, vamos a establecer el inicio de sesión para poder ver en los registros lo que se ha rechazado. Es mejor revisar estos registros de vez en cuando, incluso aunque no exista ningún problema, para tener una idea de los tipos de tráfico que se han rechazado. Si observa paquetes rechazados repetidamente de la misma red o dirección, puede que esté siendo atacado. Existe una declaración para registrar cada tipo de tráfico:

```
iptables -A FORWARD -m tcp -p tcp -j LOG
iptables -A FORWARD -m udp -p udp -j LOG
iptables -A FORWARD -m icmp -p icmp -j LOG
```

Con esto conseguiríamos una protección a nivel de cortafuegos ante los ataques más comunes de internet. El siguiente código es el ejemplo en un script.

Cuadro 12.5: Ejemplo de IPTables

```
#! /bin/bash

# Punto 1
iptables -F FORWARD

# Punto 2
iptables -F INPUT
iptables -F OUTPUT

# Punto 3
iptables -P FORWARD DROP
iptables -A INPUT -i eth0 -j DROP

# Punto 4
iptables -A FORWARD -f -j ACCEPT

# Punto 5
iptables -A FORWARD -s 192.168.0.0/24 -i eth0 -j DROP
#iptables -A INPUT -p icmp --icmp-type echo-request -j DROP  - OMITIDO , para evitar el punto 10
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

# Punto 6
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 192.168.0.0/24 --dports www,smtp --tcp-flags SYN,ACK ACK -j ACCEPT
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 192.168.0.0/24 --sports www,smtp --tcp-flags SYN,ACK ACK -j ACCEPT

# Punto 7
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 192.168.0.0/24 --dports smtp --syn -j ACCEPT

# Punto 8
iptables -A FORWARD -m multiport -p tcp -i eth0 -d 0.0.0.0 --dports www,smtp --syn -j ACCEPT

# Punto 9
iptables -A FORWARD -m multiport -p udp -i eth0 -d 192.168.0.0/24 --dports domain -j ACCEPT
iptables -A FORWARD -m multiport -p udp -i eth0 -s 192.168.0.0/24 --sports domain -j ACCEPT
iptables -A FORWARD -m multiport -p udp -i eth1 -d 0.0.0.0 --dports domain -j ACCEPT
iptables -A FORWARD -m multiport -p udp -i eth1 -s 0.0.0.0 --sports domain -j ACCEPT

# Punto 10 - OMITIDO
#iptables -A FORWARD -m multiport -p icmp -i eth0 -d 192.168.0.0/24 --dports 0,3,11 -j ACCEPT
#iptables -A FORWARD -m multiport -p icmp -i eth1 -d 0.0.0.0 --dports 8,3,11 -j ACCEPT

# Punto 11
iptables -A FORWARD -m tcp -p tcp -j LOG
iptables -A FORWARD -m udp -p udp -j LOG
iptables -A FORWARD -m icmp -p icmp -j LOG
```

Es necesario darle permisos de ejecución: `#chmod 700 nombre_script`

El resultado de nuestro cortafuegos después de ejecutar el script sería el siguiente:

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
DROP       icmp --  anywhere             anywhere            icmp echo-request

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     all  -f  anywhere             anywhere
DROP       all  --  192.168.0.0/24       anywhere
ACCEPT     tcp  --  anywhere             192.168.0.0/24      multiport dports www,smtp tcp flags:SYN,ACK/ACK
ACCEPT     tcp  --  anywhere             192.168.0.0/24      multiport sports www,smtp tcp flags:SYN,ACK/ACK
ACCEPT     tcp  --  anywhere             192.168.0.0/24      multiport dports smtp tcp flags:SYN,RST,ACK/SYN
ACCEPT     tcp  --  anywhere             0.0.0.0             multiport dports www,smtp tcp flags:SYN,RST,ACK/SYN
ACCEPT     udp  --  anywhere             192.168.0.0/24      multiport dports domain
ACCEPT     udp  --  192.168.0.0/24       anywhere            multiport sports domain
ACCEPT     udp  --  anywhere             0.0.0.0             multiport dports domain
ACCEPT     udp  --  0.0.0.0              anywhere            multiport sports domain
LOG        tcp  --  anywhere             anywhere            tcp LOG level warning
LOG        udp  --  anywhere             anywhere            udp LOG level warning
LOG        icmp --  anywhere             anywhere            icmp any LOG level warning

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

Enmascarar IP con IPTables

Cuando se diseño Internet originalmente, se reservaron varios bloques de direcciones para su uso en redes privadas. Estas direcciones nos se van a enrutar a través de Internet y se pueden utilizar sin preocuparse de que vayan a tener conflictos con otras redes. Los rangos de direcciones privadas son:

```
10.0.0.0 - 10.255.255.255
192.168.0.0 - 192.68.255.255
172.16.0.0 - 172.31.255.255
```

Al utilizar estas direcciones en nuestra LAN interna y tener una IP externa enrutable en nuestro cortafuegos, estamos protegiendo con efectividad nuestras máquinas internas ante el acceso desde el exterior. Podemos proporcionar esta capa adicional de protección fácilmente con IPTables utilizando el enmascarado IP. El encabezado IP interno se desprende en el cortafuegos y se reemplaza con un encabezado que muestra el cortafuegos como el IP de origen. A continuación se envía el paquete de datos a su destino con una dirección IP de origen de la interfaz pública del cortafuegos.

Cuando vuelve de nuevo, el cortafuegos recuerda el IP interno al que se dirige y vuelve a dirigirlo para una entrega interna. Este proceso también se conoce como Traducción de dirección de red (NAT, Network Address Translation). Con las siguientes declaraciones podemos hacerlo en IPTables:

```
iptables -t nat -P POSTROUTING DROP
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

El indicador *MASQUERADE* se puede abreviar como *MASQ*.

Conclusiones

Bueno, ahora ya sabemos cómo crear un cortafuegos básico. Se trata de una configuración bastante sencilla y las posibles variaciones son infinitas. Podemos enviar determinados puertos a servidores internos para que no tengan una dirección IP pública. Podemos colocar otra tarjeta de red en el servidor y convertirla en una interfaz DMZ para servidores con dirección pública. Existen libros completos sobre la configuración avanzada de cortafuegos y muchas listas de correo.

Existen otros métodos mas simples y rápidos de crear un cortafuegos, sin introducir comandos y tener que recordar la sintaxis. Muchas herramientas crean las declaraciones del cortafuegos utilizando una interfaz gráfica, es decir, de forma automática.

Para simplificar el método utilizare el módulo Firewall para Webmin, basado en IPTables.

12.3. Squid: Proxy transparente

Lo primero es tener el IPTables y el squid funcionando. Con las opciones NAT de *netfilter* en el kernel. Veamos el proceso:

- Activamos el reenvió de paquetes:

  ```
  echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

- Hacemos que el NAT coja todas las peticiones que vayan a un puerto (por ejemplo el 80) y las redirija al puerto del proxy (por ejemplo 3128):

  ```
  iptables -t nat -A PREROUTING -s $NUESTRA_RED -p tcp --dport 80 -j REDIRECT
           --to-port 3128
  ```

- Instalación del proxy squid:

  ```
  #apt-get install squid webmin-squid
  ```

- El fichero de configuración del squid es: `/etc/squid/squid.conf`. Lo configuramos para permitir el acceso del proxy a nuestra red interna:

  - En *#ACCESS CONTROLS*:
    `acl redInterna src 192.168.1.0/255.255.255.0`
  - En *#Only allow cachemgr access from localhost*, hay que poner antes de las denegaciones:
    `http_access allow redInterna`
  - Y en *#HTTPD-ACCELERATOR OPTIONS*:
    `httpd_accel_host virtual`
    `httpd_accel_uses_host_header on`
    `httpd_accel_with_proxy on`

También podemos configurar el proxy en modo gráfico mediante la herramienta de administración Webmin. Para cargar el modulo realizaremos un apt: `#apt-get install webmin-squid`

Ahora veamos un ejemplo del archivo */etc/squid/squid.conf* donde se limita el ancho de banda, para la conexión:

Cuadro 12.6: Ejemplo del archivo /etc/squid/squid.conf

```
#Sacado de http://debaser.ath.cx/deal/manuales/Limitar-ancho-de-banda-COMO/html/install.html
#Todas las opciones de este archivo se encuentran muy bien documentadas en el
#propio squid.conf asi
#como en http://www.visolve.com/squidman/Configuration%20Guide.html

#Los puertos por los que escuchara nuestro Squid.
http_port 8080
icp_port 3130
#los cgi-bin no se cachearan.
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
#La memoria que usara Squid. Bueno, Squid usara mucha mas que esa.
cache_mem 16 MB
#250 significa que Squid usara 250 megabytes de espacio en disco.
cache_dir ufs /cache 250 16 256

#Lugares en los que iran los archivos de bitacora de Squid.
cache_log /var/log/squid/cache.log
cache_access_log /var/log/squid/access.log
cache_store_log /var/log/squid/store.log
cache_swap_log /var/log/squid/swap.log
#Cuantas veces rotar los archivos de bitacora antes de borrarlos.
#Acuda a la FAQ para m\'as informaci\'on.
logfile_rotate 10

redirect_rewrites_host_header off
cache_replacement_policy GDSF
acl localnet src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255
acl Safe_ports port 80 443 210 119 70 20 21 1025-65535
acl CONNECT method CONNECT
acl all src 0.0.0.0/0.0.0.0
```

```
http_access allow localnet
http_access allow localhost
http_access deny !Safe_ports
http_access deny CONNECT
http_access deny all
maximum_object_size 3000 KB
store_avg_object_size 50 KB

#Configure esto si quiere que su proxy funcione de manera transparente.
#Eso significa que por lo general no tendra que configurar todos los
#navegadores de sus clientes, aunque tiene algunos inconvenientes.
#Si deja esto sin comentar no pasara nada peligroso.
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on

#Todos los usuarios de nuestra LAN seran vistos por los servidores web
#externos como si usasen Mozilla en Linux.
anonymize_headers deny User-Agent
fake_user_agent Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.6+) Gecko/20011122

#Para acelerar aun mas nuestra conexion ponemos dos lineas similares a las
#de mas abajo. Apuntaran a un servidor proxy [parent] que usara nuestro propio
#Squid. No olvide cambiar el servidor por uno mas rapido para usted.
#Puede utilizar ping, traceroute y demas herramientas para comprobar la
#velocidad. Asegurese de que los puerto http e icp son los correctos.

#Descomente las lineas que comienzan por "cache_peer" de ser necesario.
#Este es el proxy que va a usar para todas las conexiones...
#cache_peer w3cache.icm.edu.pl parent 8080 3130 no-digest default

#...excepto para las direcciones e IPs que comiencen por "!".
#No es buena idea usar un mayor
#cache_peer_domain w3cache.icm.edu.pl !.pl !7thguard.net !192.168.1.1

#Esto resulta util cuando queremos usar el Cache Manager.
#Copie cachemgr.cgi al cgi-bin de su servidor web.
#Podra acceder a el una vez lo haya hecho introduciendo en un navegador
#la direccion http://su-servidor-web/cgi-bin/cachemgr.cgi
cache_mgr your@email
cachemgr_passwd secret_password all

#Este es el nombre de usuario con el que trabajara nuestro Squid.
cache_effective_user squid
cache_effective_group squid

log_icp_queries off
buffered_logs on

#####DELAY POOLS
#Esta es la parte mas importante para configurar el trafico entrante con
#Squid. Para una descripcion detallada acuda al archivo squid.conf o a la
#documentacion de http://www.squid-cache.org

#No queremos limitar las descargas en nuestra red local.
acl magic_words1 url_regex -i 192.168

#Queremos limitar la descarga de este tipo de archivos
#Ponga todo esto en una unica linea
acl magic_words2 url_regex -i ftp .exe .mp3 .vqf .tar.gz .gz .rpm .zip .rar .avi .mpeg .mpe .mpg .qt
.ram .rm .iso .raw .wav .mov
#No bloqueamos .html, .gif, .jpg y archivos similares porque por lo general
#no consumen demasiado ancho de banda.

#Queremos limitar el ancho de banda durante el dia permitiendo
#el ancho de banda completo durante la noche.
#Cuidado! con el acl de abajo sus descargas se interrumpiran
#a las 23:59. Lea la FAQ si quiere envitarlo.
acl day time 09:00-23:59

#Tenemos dos delay_pools diferentes
#Acuda a la documentaci\'on de Squid para familiarizarse
#con delay_pools y delay_class.
delay_pools 2

#Primer delay pool
#No queremos retrasar nuestro trafico local
#Hay tres cases de pools; aqui solo hablaremos de la segunda.
#Primera clase de retraso (1) de segundo tipo (2).
delay_class 1 2

#-1/-1 significa que no hay limites.
delay_parameters 1 -1/-1 -1/-1

#magic_words1: 192.168 que ya hemos puesto antes
delay_access 1 allow magic_words1

#Segundo delay pool.
#Queremos retrasar la descarga de los archivos mencionados en magic_words2.
#Segunda clase de retraso (2) de segundo tipo (2).
delay_class 2 2

#Los numeros siguientes son valores en bytes;
#Debemos recordar que Squid no tiene en cuenta los bits de inicio/parada
#5000/150000 son valores para la red al completo
#5000/120000 son valores para la IP independiente
#una vez los archivos descargados exceden los 150000 bytes,
#(o el doble o el triple)
#las descargas proseguiran a 5000 bytes/s

delay_parameters 2 5000/150000 5000/120000
#Ya hemos configurado antes el dia de 09:00 a 23:59.
delay_access 2 allow day
delay_access 2 deny !day
delay_access 2 allow magic_words2
#EOF
```

12.4. Bastille Linux: Herramienta de seguridad

Una vez instalado nuestro sistema operativo, necesitamos fortalecerlo para utilizarlo como sistema de seguridad. Este proceso implica cerrar los servicios innecesarios, restringir los permisos y, en general, minimizar las partes de la máquina que están expuestas. Los detalles de este proceso serán diferentes dependiendo del uso pretendido de la máquina.

Antes el fortalecimiento solía ser un proceso manual intensivo a medida que se probaban y testeaban las posibles modificaciones. Sin embargo, en Linux, existen herramientas que realizarán automáticamente un fortalecimiento del sistema. Así se puede ahorrar tiempo sin olvidarnos de nada.

En este caso utilizaré la herramienta Bastille Linux, disponible en la distribución Debian. A pesar de lo que pueda parecer, no se trata de un sistema operativo independiente, sino de un conjunto de secuencias de comandos que llevan a cabo determinadas configuraciones del sistema basándose en nuestras indicaciones. Simplifica extraordinariamente el proceso de fortalecimiento y lo reduce a responder a algunas preguntas.

Si necesitamos más información la podemos obtener en la página web oficial: www.bastille-linux.org.

12.4.1. Ejecución

Es muy recomendable instalar esta herramienta, primeramente, en un entorno de pruebas. Este tipo de programas pueden desconectar los servicios necesarios para el funcionamiento de algún servidores, produciendo cortes de servicio o detención del mismo. Cuando haya probado su efecto y verificado su estabilidad, podremos ejecutar la herramienta en nuestro entorno de trabajo.

Para instalar hay que ejecutar:
`#apt-get install bastille`

Para poder usar la aplicación deben de estar instalados los siguientes paquetes:

- Perl 5.5_003 o superior
- Perl TK Module 8.00.23 o superior
- Perl Curses Module 1.06 o superior

Para ejecutar Bastille linux:
`#/usr/sbin/bastille`, ... para el modo gráfico.
`#/usr/sbin/bastille -c`, ... para el modo texto (curses).

También podemos ejecutar Bastille en lo que se denomina modo no interactivo. Este modo ejecuta Bastille automáticamente, sin hacer preguntas, desde un archivo de configuración preasignado. cada vez que ejecutamos Bastille, se crea un archivo de configuración. A continuación podemos utilizarlo para ejecutar Bastille en otras máquinas en modo no interactivo. Esta técnica es útil para cerrar rápidamente múltiples máquinas. Cuando disponga del archivo de configuración con los elementos deseados, simplemente cargue Bastille en las máquinas adicionales y copie el archivo de configuración en dichas máquinas (o deje que accedan al archivo por la red).

Para ejecutarlo escribimos:
`#bastille non-interactive config-file`

Donde *config-file* es el nombre y la ubicación del archivo de configuración que deseamos utilizar.

12.4.2. Modos de funcionamiento

Normalmente ejecutaremos Bastille en modo interactivo. En este modo responderemos a una serie de preguntas sobre cómo vamos a utilizar la máquina. Basándose en las respuestas, Bastille desconecta los servicios innecesarios o restringe los privilegios de los usuarios y servicios.

Nos pregunta cosas como "¿Desea utilizar esta máquina como acceso a máquinas Windows?". Si respondemos negativamente, desconecta el servidor Samba, que permite a nuestra máquina interactuar con las máquinas Windows. Samba podría introducir algunas brechas de seguridad potenciales en nuestro sistema, por lo que es recomendable desconectarlo si no lo necesitamos.

Si tenemos que ejecutar algunos servidores (por ejemplo, SSH), intentará establecerlos con privilegios limitados o utilizando un *chrooted jail*, es decir, si un servidor tiene que ejecutarse con acceso a la raíz del sistema, tiene una capacidad limitada de afectar a otras partes del sistema, lo que suaviza los efectos de cualquier ataque con éxito sobre dicho servicio.

A cada pregunta le acompaña una pequeña explicación de porqué esa configuración es importante, así podemos decidir si es apropiada para nuestra instalación. También existe un botón de detalle con información adicional.

Bastille adopta la solución de intentar educar al administrador mientras está cerrando el sistema. Cuanta más información tenga, mejor armado estará para los deberes de seguridad de su red. Si no está seguro, puede saltarse una pregunta y volver a ella más adelante. No hay que preocuparse, se ofrece una oportunidad final para terminar todas las configuraciones. También puede ejecutar Bastille posteriormente después de investigar sobre la respuesta y cambiar la configuración. Otra ventaja es que Bastille nos ofrece una lista de "cosas para hacer" al final de la sesión de fortalecimiento para cualquier elemento del que Bastille no se haya ocupado.

12.5. Copias de seguridad

Para realizar las copias de seguridad necesarias en nuestro servidor y dependiendo de las necesitades de nuestra empresa, necesitamos disponer de una medio seguro donde poder almacenarlas. Para ello, podemos aplicar una las siguientes soluciones:

- Almacenar las copias en el propio disco físico del servidor (no se recomienda).
- Utilizar cintas de backup.
- Utilizar DVDs de backups.

Si realizamos las copias en el propio disco o en cintas de backup, el proceso puede ser automatizado mediante el *cron* del sistema.

12.5.1. Dispositivos de cinta

Las unidades de cinta SCSI, usan el siguiente esquema de nombres:

- */dev/stX*: Dispositivo de cinta SCSI de rebobinado automático; x es el número de la unidad de cinta. Las unidades de cinta se numeran por su orden en la controladora SCSI.
- */dev/nstX*: Dispositivo de cinta SCSI sin rebobinado automático; x es el número de la unidad de cinta. Las unidades de cinta se numeran por su orden en la controladora SCSI.

Probablemente tendremos en nuestro sistema un dispositivo de enlace simbólico a la cinta: */dev/tape*.

12.5.2. Mt

El programa *mt* proporciona controles simples para la unidad de cinta, como el rebobinado, expulsión o la búsqueda de un archivo. En el contexto de las copias de seguridad, *mt* es muy útil como mecanismo de rebobinado y búsqueda.

Todas las acciones de *mt* se especifican en la línea de comandos. El cuadro 12.7 muestra los parámetros del comando.

Cuadro 12.7: Opciones del comando *mt*, para manipular cintas de backup

Parámetros	**Descripción**
`-f <dispositivo>`	Especifica el dispositivo de cinta.
`fsf <cuenta>`	Avanza un número (*cuenta*) de archivos. La cinta se coloca en el primer bloque del archivo siguiente; por ejemplo, *fsf 1* debería dejar la cabeza preparada para leer el segundo archivo de la cinta.
`asf <cuenta>`	Posiciona la cinta al comienzo del archivo indicado por *cuenta*. El posicionamiento se hace primero con un rebobinado de la cinta y después se avanza *cuenta* archivos.
`rewind`	Rebobina la cinta.
`erase`	Borra la cinta.
`status`	Da el estado de la cinta.
`offline`	Deja la cinta inactiva y, si es aplicable, la descarga.
`load`	Carga la cinta (aplicable a cambiadores de cinta).
`lock`	Bloquea la puerta de la unidad (sólo aplicable a ciertas unidades de cinta).
`unlock`	Desbloquea la puerta de la unidad (sólo aplicable a ciertas unidades de cinta).

Podemos ver los siguientes ejemplos:
`#mt -f /ver/nst0 rewind`,... Para rebobinar la cinta en */dev/nst0*.
`#mt -f /dev/nst0 asf 2`,... Mueve la cabeza lectora, para leer el tercer archivo de la cinta.

12.5.3. Dump y Restore

Las herramientas que utilizaremos serán dos:

- *dump*
- *restore*

Para instalarlas ejecutaremos un apt:
`#apt-get install dump`

La herramienta *dump* trabaja haciendo una copia de un sistema de archivos entero. La herramienta *restore* puede tomar esa copia y restaurarla.

Para soportar backups incrementales, *dump* usa el concepto de niveles de dump. Un nivel de dump de 0 significa una copia de seguridad completa. Cualquier nivel de dump superior a 0 es un incremento relativo a la última vez que se realizó un dump con un nivel de dump menor. Por poner un ejemplo, si consideramos que tenemos tres dump: el primero de nivel 0, el segundo de nivel 1 y el tercero también de nivel 1. El primer dump es una copia completa. El segundo dump contiene todos los cambios hechos desde el primer dump. El tercer dump tiene todos los cambios desde el primer dump.

La utilidad dump almacena toda la información sobre sus operaciones en el archivo */etc/dumpdates*. Este archivo lista cada copia de seguridad de un sistema de archivos, cuándo se hizo y de qué nivel. Dada esta información, podemos determinar que copia debemos restaurar.

En el cuadro 12.8 se muestran los parámetros más usuales del comando dump:

Cuadro 12.8: Parámetros del comando dump

Parámetro	Descripción
`-n`	El nivel de dump, donde *n* es un número entre 0 y 9.
`-b <tam_bloque>`	Configura el tamaño del bloque de dump a *tam_bloque*, el cual se mide en kilobytes. Si hacemos copias de archivos muy grandes, al usar un tamaño de bloque mayor aumentará el rendimiento.
`-B <cuenta>`	Especifica el número (*cuenta*) de registros por cinta. Si hay más datos sobre los que hacer dump, que espacio de cinta, dump muestra un símbolo del sistema pidiendo que se inserte una cinta nueva.
`-f <archivo>`	Especifica una localización (*archivo*) para el archivo dump resultado. Podemos hacer el archivo dump como un archivo normal que reside en otro sistema de archivos, o podemos escribirlo en un dispositivo de cinta.
`-u`	Actualiza el archivo */etc/dumpdates* después de un dump con éxito.
`-d <densidad>`	La *densidad* de una cinta en bits por pulgada.
`-s <tam>`	El tamaño (*tam*) de la cinta en pies.

Para colocar el backup sobre una cinta, comprimiendolo, podemos hacer lo siguiente:
`#dump -0 -f - /dev/hda1 | gzip --fast -c > /dev/st0`

Hay que tener cuidado con dump, se considera peligroso hacer dump de sistema de archivos que estén en uso. Para asegurarse de que no estan en uso, hay que desmontar el sistema de archivos primero. Desafortunadamente, muy poca gente se puede permitir el lujo de desmontar un sistema el tiempo necesario para hacer una copia de seguridad. Lo mejor es realizar la poca atractiva tarea de verificar las copias de seguridad sobre una base normal. La verificación se hace comprobando que el programa *restore* puede leer completamente la copia y extraer los archivos de ella. Es tedioso y nada divertido, pero muchos administradores de sistemas perdieron su trabajo después de copias de seguridad erróneas (y no queremos ser uno de ellos).

Durante mis estudios en la FIB (Facultad de Informática de Barcelona) coincidí con varios profesores que eran, al mismo tiempo, muy buenos administradores de sistemas. Y uno de ellos, Alex Ramirez, nos dijo la siguiente frase, "La primera tarea de un administrador de sistemas es hacer copias de seguridad, nadie quiere hacerlas y por eso nos pagan a nosotros para que las hagamos. Las copias de seguridad, sólo se hechan de menos cuando nadie las ha hecho".

Uso de dump para hacer backup de un sistema entero

La utilidad dump trabaja haciendo un archivo de un sistema de archivos. Si el sistema entero contiene varios sistemas de archivos, necesitaremos ejecutar dump por cada sistema de archivos. Puesto que dump crea su salida como un gran archivo independiente, podemos almacenar varios dumps en una sola cinta mediante el uso de un dispositivo de cinta sin rebobinado automático, otra solución factible es el uso de DVDs grabables. Primero tenemos que decidir sobre qué sistemas de archivos vamos a hacer la copia, está información está en el archivo */etc/fstab*.

Si por ejemplo queremos hacer un backup de: /dev/hda1, /dev/hda3, /dev/hda5 y /dev/hda6. Y grabarlo en /dev/nst0, comprimiendo los archivos dump, debemos ejecutar los siguientes comandos:

```
#mt -f /dev/nst0 rewind
#dump -0uf - /dev/hda1 | gzip --fast -c > /dev/st0
#dump -0uf - /dev/hda3 | gzip --fast -c > /dev/st0
#dump -0uf - /dev/hda5 | gzip --fast -c > /dev/st0
#dump -0uf - /dev/hda6 | gzip --fast -c > /dev/st0
#mt -f /dev/nst0 rewind
#mt -f /dev/nst0 eject
```

Uso de restore

El programa restore lee los archivos dump y extrae archivos y directorios individuales de ellos. Aunque restore es una herramienta de línea de comandos, ofrece un modo interactivo muy intuitivo que le mueve a través de la estructura de directorios de la cinta.

El cuadro 12.9 muestra las opciones de línea de comandos de la utilidad restore.

Cuadro 12.9: Opciones del comando RESTORE

Opción	Descripción
`-i`	Activa el modo interactivo de restore. La utilidad leerá el contenido del directorio de la cinta y nos dará una interfaz parecida a la shell en la cual nos podemos mover entre los directorios y señalar los archivos que queramos, restore irá al dump y los restaurará. Este modo es útil para recuperar archivos individuales, especialmente si no estámos seguro en qué directorio están.
`-r`	Reconstruye un sistema de archivos. en el caso de que pierda todo un sistema de archivos (por ejemplo, por un fallo de disco), puede recrear un sistema de archivos vacío y restaurar todos los archivos y directorios del archivo dump.
`-b tam_bloque`	Configura el tamaño del bloque de dump a *tam_bloque* kilobytes. Si no proporciona esta información, restore se la pedirá.
`-f nom_archivo`	Lee un dump del archivo *nom_archivo*.
`-T directorio`	Especifica el espacio de trabajo temporal (*directorio*) para el restore. Por defecto es */tmp*.
`-v`	La opción verbal; muestra cada paso del restore.
`-y`	En el caso de un error, reintenta automáticamente en lugar de preguntar al usuario si quiere probar de nuevo.

Un restore típico podría ser el siguiente:

```
#restore -ivf /dev/st0
```

Donde tenemos el archivo dump en */dev/st0*, visualizaremos cada paso que tome restore y entraremos en una sesión interactiva donde decidiremos qué archivos se restauran.

Y para un restore completo desde la cinta */dev/st0*, si perdemos el sistema de la unidad SCSI */dev/sda1* que contiene */home*, sustituimos la unidad y lo recrear ayudandonos con *mke2fs*.

```
#mke2fs /dev/sda1
#mount /dev/sda1 /home
#cd /home
#restore -rf /dev/st0
```

12.5.4. Configuración gráfica de backups, interfaz Webmin

Para realizar de una forma mucho más simple las copias de seguridad, podemos utilizar la herramienta de configuración por web: Webmin.

Para instalar el módulo de copias de seguridad ejecutaremos un apt:

```
#apt-get install webmin-fsdump
```

Este módulo permite hacer backups de:

- Sistemas EXT2
- Sistemas EXT3
- Archivos TAR

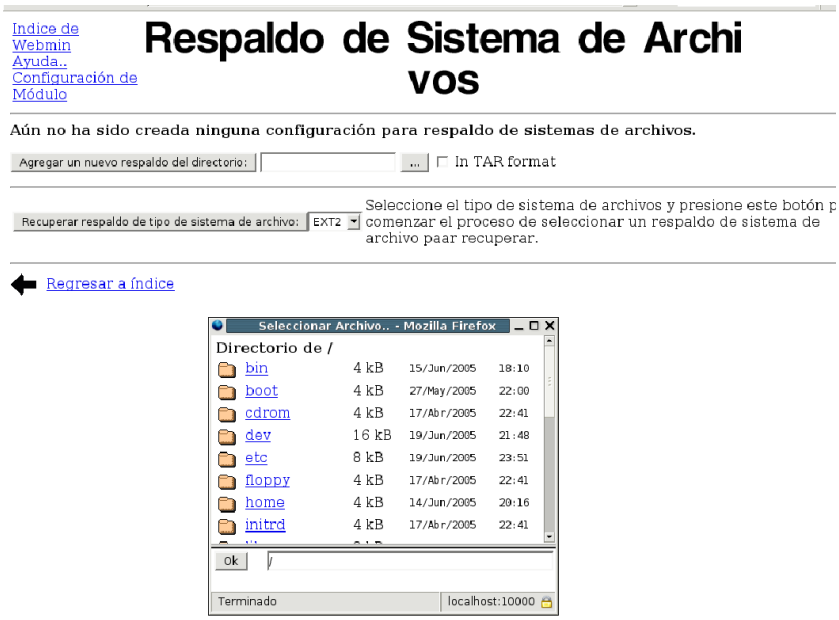

Figura 12.1: Módulo Webmin para realizar Backups del sistema

12.5.5. K3B: Grabación de CDs y DVDs

Si queremos guardar las copias en CDs o DVDs, nos resultará muy útil una interfaz gráfica simple e intuitiva, como la que nos ofrece el programa *K3B* basado en el entorno KDE.

Lo podemos instalar con un apt: `#apt-get install k3b`
Y para ejecutarlo introduciremos el comando: `#k3b`

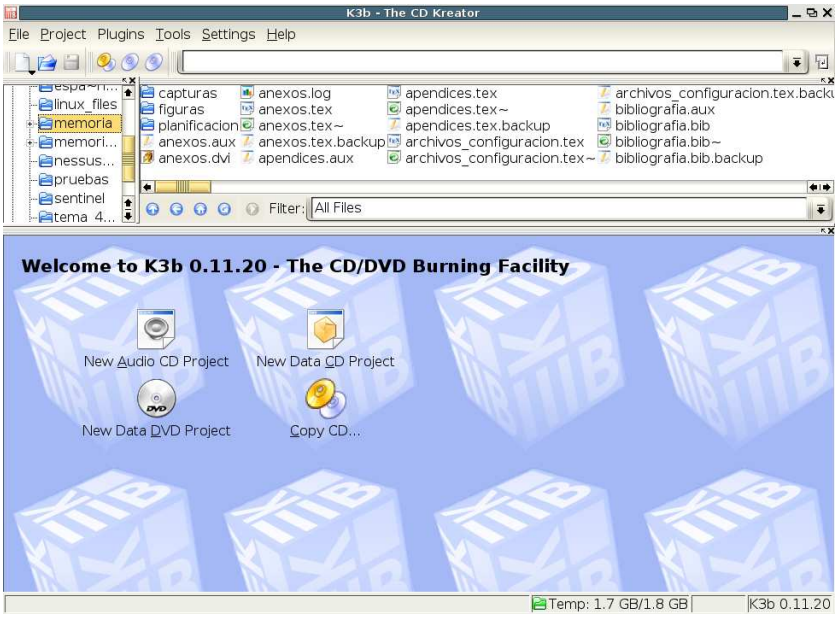

Figura 12.2: Programa K3B para grabar CDs y DVDs