Capítulo 2
Estado del arte de la seguridad informática.

"It is easy to run a secure computer system. You merely have to disconnect all dial-up connections and permit only direct-wired terminals, put the machine and its terminals in a shielded room, and post a guard at the door"

F.T. Grampp.

Sed quis custodiet ipsos custodes? (But who will guard the guards themselves?)

Juvenal.

En este Capítulo se hace un análisis del estado del arte en la seguridad informática. Se presentan los puntos más relevantes sobre la seguridad en el sistema operativo UNIX y en algunos de sus servicios. Se exponen los conceptos más relevantes sobre criptología, firewalls y herramientas existentes.

2.1 La seguridad en UNIX. Prevención

UNIX en sus inicios no fue diseñado para ser un sistema seguro [RIT79], dado que en principio fue concebido para uso académico y, por tanto, estaba abierto a todo el mundo. Una vez que se fue extendiendo su utilización, aparecieron distintas marcas que empezaron a ofrecer en sus distribuciones el nivel de seguridad necesario para los actuales requerimientos que se le suponen a este sistema. Las implicaciones de seguridad se pueden dividir en tres áreas o dominios dependiendo de la parte del sistema que se vea involucrado:

- la seguridad en las cuentas,
- la del sistema de archivos, y
- la de red.

A continuación se presentan las medidas de control que se introducen, los posibles fallos si no se hace un manejo adecuado de ellos y algunas líneas a seguir para conseguir una protección adecuada.

2.1.1 Seguridad en el sistema de cuentas

Seguridad en el sistema de cuentas

La forma más fácil para llevar a cabo un acceso no autorizado en un sistema es a través de la cuenta de otra persona, para evitar esto la medida de seguridad que presenta el sistema es el nombre de usuario y la contraseña. Es importante, para que la seguridad no se vea comprometida, controlar viejas cuentas, evitar contraseñas fácilmente "adivinables", y supervisar todo lo que pueda facilitar un acceso a una cuenta por parte de una persona que no sea su propietario [CUR90][TOM94].

Control de los passwords

El password es la medida más importante de seguridad en UNIX. Si se consigue romper esta barrera el posible intruso adquiere la identidad del propietario. Esto es crucial en el caso de que se vea involucrada la de root. Es fundamental un buen control de los passwords y de las posibles cuentas que no estén siendo usadas. Es fundamental una buena política de elección. Algunas de las líneas que los administradores de seguridad deben dar a los usuarios son las siguientes[CUR90]:

- No utilizar el nombre de la cuenta(login), tal cual, al revés, en mayúsculas.
- No utilizar nuestro nombre, ni apellidos.
- No utilizar nombres de familiares.
- No emplear información personal que pueda ser obtenida fácilmente.
- No emplear un password sólo con números o con la misma letra.
- No emplear palabras que se encuentren en los diccionarios.
- No utilizar palabras de menos de seis letras.
- Mezclar letras mayúsculas y minúsculas.
- Utilizar palabras que sean fáciles de recordar para que no haya que escribirlas.
- Emplear un password que pueda ser tecleado rápidamente sin necesidad de mirar al teclado.
- No escribirlos en ningún sitio, ni siquiera archivos.

El problema de no seguir estas reglas es que existen programas que van probando passwords hasta que encuentran el correcto, basados en un diccionario y combinando algunas palabras. Una vez escogido un password es importante seguir una política adecuada, unas reglas generales pueden ser:

- Su único lugar debe ser la memoria de su dueño.
- No decirlos nunca a otra persona.
- Establecer un tiempo máximo de uso para cada uno.

Control de las cuentas

a) Cuentas con vida limitada. Es posible que en algunos sitios, donde haya multitud de usuarios, existan viejas cuentas de personas que hayan abandonado la empresa. Estas cuentas suelen suponer un agujero de seguridad importante, no sólo porque alguien pueda entrar en ellas, sino porque si se produce un acceso no autorizado es muy difícil de detectar. Para evitar esto hay que poner fecha de expiración en todas las cuentas. La fecha puede almacenarse fácilmente en el archivo /etc/shadow. Este archivo está pensado para no dejar en el /etc/passwd los passwords encriptados de los usuarios. Puesto que UNIX requiere que sobre este archivo tengan derechos de lectura, los passwords se almacenan en el /etc/shadow. La línea completa que puede incluirse, para cada usuario, es la siguiente:

username:passwd:lastchg:min:max:warn:inactive:expire

lastchg indica el número de días desde el 1 de enero de 1970 hasta el último cambio de password.
min es el menor número de días entre cambios de password.

max máximo número de días entre cambios de password.
warm se da un aviso antes de que el password haya expirado.
inactive es elnúmero de días que se permiten antes de que se bloquee la cuenta.
expire indica el momento en el que la cuenta va a expirar.

Mediante el comando *passwd*, se puede, cerrar una cuenta, bloquear un password, hacer que en el próximo acceso al sistema se cambie, o asignarles un tiempo de vida.

b) Cuentas multiusuario: Hay que tener también un especial cuidado con las cuentas públicas [TOM94]. A ellas acceden usuarios que permanecen poco tiempo en el sistema. Lo mejor es crearlas para un fin concreto y borrarlas después. No deben tener palabras de acceso tan simples como ``guest'' o ``visitor'' y nunca deben permanecer en el archivo de passwords del sistema. También suelen plantear problemas de seguridad algunas cuentas que se instalan para ejecutar ciertos comandos sin acceder a la máquina. No tienen password, por tanto pueden ser usadas por cualquiera. Algunas de ellas tienen como identificador de usuario el cero. Estas cuentas prácticamente ``invitan'' a los crackers a entrar en ellas. La forma de acceder es sencilla, si un usuario puede acceder al sistema, reemplazando los comandos por unos suyos, podrá ejecutarlos con permisos de root.

2.1.2 Seguridad en el sistema de archivos

La seguridad en el sistema de archivos es fundamental para la protección de los datos de acceso no autorizados. Las barreras de entrada que nos encontramos en el sistema de archivos son los derechos sobre los mismos[RIT79].

Cada archivo o directorio de nuestro sistema tiene asociado 3 conjuntos de bits asociados a los permisos: uno del usuario, dueño del archivo, otro para el grupo al que pertenece y otro para el resto de los usuarios. Cada grupo contiene los tres bits conocidos de permisos, uno de lectura, otro de escritura y otro de ejecución. El significado varía si hace referencia a un directorio o a un archivo. La primera norma de seguridad en el sistema de archivos es dar los permisos adecuados a cada archivo y usuario. Normalmente los archivos se crean sin tener en cuenta los derechos que se les da [Tom94]. Una forma sencilla para controlar esto es utilizar el comando *umask*, así pues:

Con umask 022 los archivos tendrán los permisos: **- r w - r - - r - -**
Con umask 077 crea los archivos con los permisos: **- r w - - - - - - -**

Hay que asegurar que estas líneas están incluidas en el .cshrc o en el .profile.

2.1.2.1 GUID, SGID y sticky bit.

Existe además para cada conjunto de permisos[CUR90], un cuarto bit, que tiene un significado distinto en cada uno de ellos:

setuid Si el conjunto es el de derechos de propietario, este bit controla si el archivo es o no. Este estado permite que cuando ejecutemos el programa adquiramos los derechos del propietario del mismo. Nuestro indicador de usuario efectivo(EUID), se cambia al del propietario. Esta facilidad la usan las aplicaciones multiusuario para acceder archivos que pertenecen al resto. Por ejemplo el sendmail tiene el setuid asignado al root, permitiendo al programa escribir en los archivos de correo a los que el usuario que lo ejecuta no tiene permiso.

setgid Es análogo al anterior pero para los grupos.

sticky Si está activo indica al sistema operativo trate de manera especial el texto imagen de un archivo ejecutable. Tiene poco significado hoy en día y es más para compatibilidad con antiguas versiones de UNIX.

En ocasiones se abusa de esta facilidad. Si un hacker llegar a ser un determinado usuario, puede ser posible que sea capaz de ejecutar programas como ese usuario. Si gana el EUID 0 podrá editar el archivo de passwords y crearse una cuenta de root.

Habitualmente, los permisos de los archivos en Unix corresponden con un número octal que varía entre 000 y 777; sin embargo existen unos permisos especiales que hacen variar esta definición y van desde 0000 y 7777; se trata de los bits de permanencia (1000), SGID (2000) y SUID (4000) [VIL00]

El bit de SUID o setuid se activa sobre un archivo añadiéndole a la representación octal de los permisos de un archivo y otorgándole además permiso de ejecución al propieratio del mismo, al hacer esto, en lugar de la x en la primera terna de los permisos, aparecera una s o una S si no se ha otorgado el permiso de ejecución correspondiente (en ese caso el bit no tiene efecto)

```
laboper# chmod 4777 /tmp/file1
laboper# chmod 4444 /tmp/file2

laboper# ls -la /tmp/file1
-rwsrwxrwx   1 root     other          0 Apr  2 14:15 /tmp/file1

laboper# ls -la /tmp/file2
```

```
-r-Sr--r--   1 root     other         0 Apr  2 14:16 /tmp/file2
```

El bit SUID activado sobre un archivo indica que todo aquél que ejecute el archivo va a tener durante la ejecución los mismos provilegios que quién lo creó. (Si root crea un achivo y le pone este bit, todo aquél usuario que lo ejecute va a disponer, hasta que el programa finalice, de un nivel de privilegio total en el sistema)

Todo lo mencionado sobre el setuid es aplicable al bit setgid, pero a nivel del grupo del usuario en lugar de propietario: en lugar de trabajar con el EUID del propietario, todo usuario que ejecute un programa con setgid tendrá los priviletios del grupo al que pertenece el usuario. Para activarlo, se suma 2000 a la representación octal del permiso del archivo y además se necesita permiso de ejecución en la terna del grupo (aparecerá una s o S en la terman del grupo). Si fuera un directorio, el setgit afectará a los archivos y subdirectorio que se crean en él.

Los bits de setuid y setgis dan a Unix una gran flexibilidad, pero constituyen al mismo tiempo la mayor fuente de ataques internos al sistema (entendiendo por ataques internos aquellos realizados por un usuario - autorizado o no - con el objeto de aumentar su nivel de privilegio en la máquina).

Es necesario el riesgo que se corre al no eliminar todos los archivos con este tipo de permisos. Para unos archivos de Unix es estrictamente necesario hacerlo así.

Ejemplo:

Un archivo con setuid clásico en cualquier clon de Unix es /bin/passwd , la instrucción para que cada usuario pueda cambiar su passwd de entrada al sistema. Sus principales funciones consisten en modificar el archivo */etc/passwd* y/o */etc/shadow* . Se resalta el punto que el usuario por si mismo no tiene el nivel de privilegio necesario para hacer esto, por lo tanto existen varias soluciones: se puede asignar permiso de escritura para todo el mundo sobre dicho(s) archivo(s), se puede denegar a los usuarios el cambiar su passwd o se puede obligar al usuario a pasar por el Departamento de Operación cada vez que quieran cambiarlo. En este caso, se asume que el bit de setuid en */bin/passwd* es imprescindible. Así como este caso, en una instalación pura de Unix existen mas de cincuenta archivos con estos permisos y es responsabilidad del administrador el conocerlos y determinar si los elimina o no según las necesidades.

Tambien se debe estar atento a nuevos archivos con estas características; demasiadas aplicaciones de Unix se instalan con esos permisos aún cuando no se

necesitan. Y todos aquellos archivos que misteriosamente aparecen en la máquina, son seguramente causa de que un intruso está instalado en ella.

Por otra parte, el stiky bit o bit de permanencia se activa agregándole 1000 a la representación octal de los permisos de un determinado archivo y otorgándole además permiso de ejecución, así en lugar de una x en la terna correspondiente al resto de usuarios, aparece una t (si no se ha dado permiso de ejecución, será T) :

```
laboper# chmod 1777 /tmp/file1
laboper# ls -la /tmp/file1
-rwxrwxrwt   1 root     other          0 Apr  2 14:15 /tmp/file1

laboper# chmod 1774 /tmp/file2
laboper# ls -la /tmp/file2
-rwxrwxr-T   1 root     other          0 Apr  2 14:16 /tmp/file2
```

Si el bit de permanencia está activado, se está indicando al sistema operativo que se trata de un archivo muy utilizado, por lo que es conveniente que prermanezca en memoria principal el mayo tiempo posible, esta opción se utilizaba en sistemas anteriores que disponían de muy poca RAM, pero hoy en día prácticamente no se utiliza. Lo que si sigue vigente es el efecto del stiky bit activado sobre un directorio: en este caso se indica al sistema operativo que, aunque los permisos "normales" digan que cualquier usuario puede crear y eliminar archivos, sólo el propietario de cierto archivo y el administrador pueden borrar un archivo guardado en un directorio con estas características. Esto se utiliza principalmente en archivos donde interesa que todos puedan escribir pero no borrar, como es el caso de /tmp : si el equivalente octal de los permisos de éste fueran solo 777 en lugar de 1777, cualquier usuario podria borrar los archivos del resto.

Shells scripts con setuid

Los programas [TOM94] que tienen los bits de setuid o setguid activos no son seguros. Hay que tener especial cuidado cuando se escriben dichos programas. Existen numerosos paquetes de software que intentan que estos shells sean seguras, pero no se ha conseguido resolver del todo. El problema de un script con este bit activo es que puede ser interrumpido, dejando al usuario que lo utiliza con los privilegios del propietario. Esto se puede hacer, por ejemplo, para el .profile. Para evitar esto hay que usar la orden trap. También hay que tener cuidado cuando se llama a un programa ejecutable desde un shell, puesto que se ejecutará sobre su propio shell, que puede interrumpirse también. Para prevenir esto, es preciso lanzar el programa con el comando exec. Los distintos shells tratan de forma distinta el EUID [TOM94]:

Bourne Shell. Esta shell por defecto protege contra esto e ignora el EUID y el EGID, activándolos a UID y GID respectivamente. Este comportamiento puede deshabilitarse con la opción -p.

Korn Shell. Esta shell no tiene la posibilidad anterior. En cualquier, caso va a leer el /etc/suid_profile si el EUID no es igual al UID, y lo mismo para EGID y el GID.

C-Shell. No permite que se ejecuten shell scripts con el bit setuid activo. Si se invoca con la opción -b deshabilita esta característica.

El Sticky bit en los directorios: Cuando el sticky bit está activo en los directorios, quiere decir que los usuarios no pueden borrar o cambiar el nombre de archivos de otros usuarios que estén en el mismo. Esto es útil para el directorio /tmp.

2.1.2.2 Seguridad en las bibliotecas compartidas

Las bibliotecas compartidas son muy comunes en Solaris [SUN88]. Permiten ahorrar memoria a la hora de ejecutar los procesos. Los programas que usan las mismas funciones, no precisan, merced a ellas, tened su propia copia de las mismas en memoria. Se enlazan en tiempo de ejecución. La mayor parte de los comandos de /etc/bin usan esta característica. El comando **ldd** permite conocer qué bibliotecas usa cada programa. Por ejemplo:

```
 % ldd /usr/bin/cat
libintl.so.1 => /usr/lib/libintl.so.1
libw.so.1 = > /usr/lib/libw.so.1
libc.so.1 => /usr/lib/libc.so.1
libdl.so.1 => /usr/lib/libdl.so.1
```

Un usuario puede crear su propia versión de una de ellas y alterar el camino de la misma. Por ejemplo:

```
% cp /usr/lib/libc.so.1 /tmp
% touch /tm/libc.so.1
% setenv LD_LIBRARY_PATH /tmp
% ldd /usr/bin/cat
libintl.so.1 => /usr/lib/libintl.so.1
libw.so.1 = > /usr/lib/libw.so.1
libc.so.1 => /tmp/libc.so.1
```

```
libdl.so.1 => /usr/lib/libdl.so.1
```

Esto, para el comando **cat**, no es importante, pero si fuese un programa setuid, el creador de la nueva biblioteca podría ejecutar la parte de código que quisiese, como un usuario más privilegiado. Para evitar esto, es posible, hacer que un programa ignore la variable LD_LIBRARY_PATH usando una nueva variable denominada LD_RUN_PATH:

```
% setenv LD_RUN_PATH /usr/lib
% unsetenv LD_LIBRARY_PATH
% cc -o prog prog.c
```

Si ahora ``prog'' tiene el bit setuid activo se ignorará el valor de LD_LIBRARY_PATH. Todos los programas de /usr/bin están compilados de esta forma. No se debe perder de vista cúales de las aplicaciones que corren con setuid activo pueden ignorar LD_LIBRARY_PATH.

2.1.2.3 Encriptación de archivos

La última barrera de protección de nuestros ficheros es la posibilidad de encriptarlos. En Solaris existen dos programas para llevarlo a cabo: el crypt y el DES. Pero para distribuciones fuera de Estados Unidos sólo se incluye el primero. El crypt sólo puede prevenirnos de que un ``espía'' casual vea el contenido de nuestro archivo, pero no puede protegerlo de un cracker.

Este programa implementa una máquina rotor que sigue las líneas del Enigma Alemán (deshabilitado en la Segunda Guerra Mundial). Se conocen varios métodos para desencriptar archivos bastante largos en pocas horas sin saber qué es lo que contienen. El DES, sin embargo, es mucho más seguro, de todas formas nunca se ha probado que lo sea totalmente.

El comando ***crypt*** permite cifrar y descifrar archivos en diferentes sistemas Unix, si no recibe parámetros lee los datos de la entrada estándar y los escribe en la salida estándar. [VIL00]

Un ejemplo simple de su uso es el siguiente:

```
laboper# cat /tmp/file1
archivo de prueba. contenido peligroso
laboper# crypt </tmp/file1> /tmp/file1.crypt
```

```
Enter key:
```

En el anterior ejemplo se cifró el archivo /tmp/file1 y guardó el resultado en el archivo /tmp/file1.crypt , el original en texto plano se mantiene en el directorio, y si se desea se puede borrar.

```
laboper# file /tmp/file1.crypt
/tmp/file1.crypt:       data
laboper# cat /tmp/file1.crypt
GÁ
%B\÷Q¶¶Ìä
```

Para descifrar el archivo, se utiliza el mismo comando:

```
laboper# crypt </tmp/file1.crypt> /tmp/file1.bueno
Enter key:
laboper# cat /tmp/file1.bueno
archivo de prueba. contenido peligroso
```

Crypt no se debe utilizar para cifrar información confidencial, la seguridad del algoritmo de cifrado utilizado por este comando es mínima para estos tiempos, ya que se basa en una máquina con un rotor de 256 elementos similar en muchos aspectos a la alemana Enigma, y con unos métodos de ataque rápidos y conocidos es posible violarla.

Es posible incluir la clave de cifrado en la línea de comandos:

```
laboper# crypt key </tmp/file1> /tmp/file1.crypt
```

y con ello, es posible espiarla con el comando ***ps*** directamente.

Este comando no tiene relación con la función crypt() utilizada para cifrar los passwords de los usuarios, ésta está basada en una variante del algoritmo DES, la cuál se puede considerar segura en la mayoría de los entornos.

2.1.2.4 ACL's

Las listas de control de acceso (ACLs Access Control List) proveen un nivel de seguridad adicional a los archivos extendiendo el esquema clásico de Unix. [VIL00] Mientras que los últimos se puede especificar permisos para los tres grupos de usuarios habituales (dueño, grupo y mundo) las ACLs permiten asignar permisos a usuarios o grupos específicos. Por ejemplo, se pueden otorgar ciertos permisos a dos usuarios sobre

los mismos archivos sin necesidad de incluirlos en el mismo grupo. Estas listas de acceso existen en los sistemas Unix (Solaris, AIX, HP-UX, excepto LINUX)desde hace más de diez años.

Bajo el esquema tradicional, si se quieren ver los permisos de un archivo basta hacer un listado largo:

```
sandra : > ls -l /usr/local/sbin/sshd
lrwx------  1 root           5 Aug 26  2000 /usr/local/sbin/sshd -> sshd2
```

Getfacl.

Viendo el resultado, se sabe directamente que el archivo sshd puede ser leído por el administrador, pero por nadie más, sin embargo no se conoce el estado de la lista de control de acceso asociada al archivo. Para ver esa lista, en Solaris se utiliza la instrucción ***getfacl***:

```
sandra : > getfacl /usr/local/sbin/sshd

# file: /usr/local/sbin/sshd
# owner: root
# group: other
user::rwx
group::---              #effective:---
mask:---
other:---
```

Aquí se muestra en primer lugar el nombre del archivo, su dueño y su grupo, todos precedidos por # . Lo que se muestra a continuación es la lista de control: los campos **user**, **group** y **other** son básicamente la interpretación que getfacl hace de los permisos del archivo. El campo mask es muy similar al umask clásico: define los permisos máximos que un usuario (a excepción del pripietario) o grupo pueden tener sobre el archivo. Finalmente el campo effective indica para cada usuario (excepto el propietario) o grupo el efecto que la máscara tiene sobre los permisos: es justamente el campo que se tiene que analizar si se quiere ver quién puede accesar al archivo y de qué forma.

Se puede notar que la estructura de la lista de control de acceso otorga los mismos permisos que las ternas clásicas. Esto es algo normal en todos los Unix, si no se indica lo contrario al crear un archivo se le asocia un ACL que coincide con los permisos que ese archivo tiene en el sistema (cada archivo tendra una lista asociada, igual que tiene unos

permisos); de esta forma, el resultado anterior no es mas que la visión que getfacl tiene de los bits **rwx** del archivo.

Setfacl.

Lo interesante de cara a la protección de archivos es extender los permisos clásicos del archivo, modificando su lista asociada. Esto se consigue con la instrucción ***setfacl***:

```
sandra : > setfacl -m user:enrique:r-x /usr/local/sbin/sshd
sandra : > getfacl /usr/local/sbin/sshd

# file: /usr/local/sbin/sshd
# owner: root
# group: other
user::rwx
user:enrique:r-x      #effective:---
group::---            #effective:---
mask:---
other:---
```

Como puede verse, se acaba de modificar la lista de control de acceso al archivo para asignarle a enrique permiso de lectura y ejecución sobre el mismo. La instrucción setfacl se utiliza principalmente de tres formas:

Añadir entradas a la ACL, mendiante la opción **-m** seguida de las entradas que se desean añadir separadas por comas. Utilizando la opcion **-s** para reemplazar la ACL completa (indicando todas las entradas, separadas por comas) o bien borrar entradas de la lista con la opción **-d** (de sintaxis igual a -m)

Cada entrada de la ACL tiene el siguiente formato:

tipo:UID|GID:permisos

El tipo indica a quién aplicar los permisos (por ejemplo: *user* para el propietario o *mask* para la máscara), el UID indica el usuario al que queremos asociar la entrada (como se ha visto, se puede utilizar también el login), y el GID hace los mismo con el grupo. Por último, el campo de permisos hace referencia a los permisos a asignar, y puede ser especificado mediante símbolos rwx- o de forma octal.

Se acaba de indicar que el usuario enrique tenga permiso de lectura y ejecución sobre el archivo /usr/local/sbin/sshd , no obstante, si ahora este usuario intenta acceder al archivo en estos modos obtendrá un error.

```
sandra : > /usr/local/sbin/$ ./sshd
sandra : > ./sshd: Permission denied
```

No está sucediendo nada anormal, simplemente está actuando la máscara sobre sus permisos (antes se dijo que se prestara atención al campo effective, y aquí se puede ver que no se ha modificado). Para solucionar esto se debe modificar el campo mask:

```
sandra : > setfacl -m mask:r-x  /usr/local/sbin/sshd
```

Ahora el usuario *enrique* podrá accesar al archivo para leerlo o ejecutarlo.

Si en esta ocasión se obtiene un error, este ya no depende de la protección de los archivos sino de la configuración del programa. enrique no podrá utilizar recursos a los que no le está permitido el acceso (como puertos bien conocidos, otros archivos o procesos). Hay que recordar que si un usuario accesa un archivo que le pertenece a root, los privilegios del usuario no cambian.

Para el ejemplo citado, no es posible que enrique de acceso a nuevos usuarios. Esto es, el no pude crear nuevos ACL sobre el archivo al que se le permitió el acceso.

En Solaris, para indicar que una lista de control de acceso otorga permisos no reflejados en los bits rwx se situa un símbolo + a la derecha de los permisos en un listado largo.

```
sandra : > ls -l /usr/local/sbin/sshd
-rwx------+    1  root     bin        95745    Mar 24 2000  /usr/local/sbin/sshd
```

Otra característica que tiene Solaris es la de leer las entradas de una lista de acceso desde un archivo en lugar de indicarlas en la línea de comandos, mediante la opción **-f** de setfacl; el formato de este archivo es justamente el resultado general de getfacl, lo que permite copiar ACLs entre archivos de una forma muy cómoda.

```
sandra : > getfacl  /usr/local/sbin/sshd > /tmp/acl_tml
sandra : > setfacl  -f  /tmp/acl_tml   /usr/locla/sbin/miprograma
sandra : > getfacl /usr/locla/sbin/miprograma

# file: /usr/local/sbin/sshd
# owner: root
```

```
# group: other
user::rwx
user:enrique:r-x      #effective:---
group::---            #effective:---
mask:r-x
other:---
```

Antes de finalizar este apartado dedicado a listas de control de acceso, es conveniente comentar el principal problema de estos mecanismos. Está claro que las ACLs son de gran ayuda para el administrador de sistemas Unix, tanto para incrementar la seguridad como para faciliar ciertas tareas; sin embargo, es fácil darse cuenta de que se pueden convertir en gran ayuda para un atacante que desee situar puertas traseras en las máquinas.

2.1.3 Control de los dispositivos

La seguridad de los dispositivos es una característica importante en UNIX. Los programas los emplean para acceder a los datos en los discos o en memoria. Si no están adecuadamente protegidos el sistema está abierto a un posible ataque. Es importante que se sigan las siguientes líneas:

- Los archivos /dev/kmem, /dev/mem y /dev/drum no deben poder leerse por todos los usuarios.
- Los dispositivos de disco como /dev/sd0a y /rxylib deben pertenecer a root y al grupo operator y debe tener modo 640.
- Con muy pocas salvedades, todos los otros dispositivos deben pertenecer al root. Una excepción son los terminales cuya pertenencia cambia al usuario que ha accedido al sistema desde él. Cuando abandona el sistema vuelve automáticamente a root.

2.1.4 Seguridad en x11

Una de las características de los sistemas basados en X-windows es permitir a los usuarios correr sus aplicaciones en un sistema y poder visualizar la salida en otro. Esto conlleva varias consideraciones de seguridad. Open Windows 3.3 tiene dos mecanismos de control de acceso: uno basado en usuario y otro basado en host. El segundo se utiliza sólo por compatibilidad con otros sistemas [SUN90a].

El sistema basado en host tiene deficiencias de seguridad porque si se consigue acceder a otro sistema todos los usuarios de este último podrán acceder a nuestro servidor X. El segundo sistema da un acceso de acuerdo con un determinado identificador de red, NetID.

Existen dos tipos de protocolos para la autentificación basada en usuario: MIT-MAGIC-COOKIE-1 y SUN-DES-1 [SUN90a]. La primera se usa por defecto en OpenWindows. El segundo protocolo es mucho más robusto y usa RPCs seguros. Para seleccionar un tipo mecanismo determinado de autentificación basta con invocar al ambiente gráfico solicitando el tipo de autenticación deseado, por ejemplo:

```
``openwin''                 Autentificaci'on MAGIC-COOKIE
``openwin -noauth''         Autentificaci'on basada en host
``openwin -auth sun-des''   Autentificaci'on SUN-DES
```

Con MAGIC-COOKIE la información de acceso se guarda en el archivo .Xauthority en el directorio local de cada usuario. Si alguien consigue acceder a ese archivo podrá ver todo aquello que se escribamos en la pantalla. El archivo .Xauthority se puede modificar con los comandos xauth o con xhost. Con la opción:

```
% xhost +
```

se permite acceder a cualquiera al servidor de X. Con este comando se puede controlar el acceso basado en host apropiado para un entorno monousuario. Si se tienen hosts o terminales multiusuario con sistema X, entonces es conveniente usar una autentificación basada en el usuario.

2.1.5 Puertas traseras.

Las puertas trasceras son fragmentos de código en un programa que permiten saltarse los métodos usuales de autenticación para realizar ciertas tareas. [VIL00]. Generalmente son colocadas por los programadores en la fase de prueba de un desarrollo, del cuál se eliminan en la fase final. Lamentablemente esto no sucede siempre. Supongamos un escenario donde se necesitan de cuatro claves para autentificación y que en la fase de pruebas se dispone de una clave que abarca el poder de las cuatro. Esto es aceptable para dicha fase, pero constituye un grave peligro si el sistema ya está funcionando en la vida real: cualquiera que la conozca puede saltarse el mecanismo de seguridad del programa.

Además de las puertas traseras en los programas es posible situarlas en ciertos archivos vitales para el sistema; generalmente, cuando un atacante consigue acceso a una máquina Unix desea mantener ese acceso aunque su penetración sea detectada. Por ejemplo: algo muy habitual es añadir un usuario con UID 0 en el archivo de passwords de forma que el pirata pueda seguir accediendo al sistema con ese nuevo login, aunque el

administrador cierre la puerta que utilizó para entrar. También es común añadir servicios en puertos no utilizados, de forma que haciendo un *telnet* a ese número de puerto se abra un shell con permisos de root, incluso muchos intrusos utilizan la facilidad del **cron** para checar periódicamente esos archivos e instalar las puertas que hayan sido cerradas.

La prevención de estos ataques consiste en revisar periódicamente la integridad de los archivos más importantes (archivos de passwords, spoolers, configuración de la red, archivos de arranque, etc), tambiés es conveniente rastrear la existencia de nuevos archivos con setuid que puedan "aparecer" misteriosamente. Los más paranoicos no deben olvidar efectuar una búsqueda bajo los dispositivos montados.

2.1.5.1 Crontab

Como administrador, este comando es de suma importancia, pues permite la repetición de comandos a cualquier hora o día [FRI 91]. Cuando el programa cron se ejecuta al mismo tiempo que el inicio del sistema, el programa lee instrucciones para comandos a ejecutar en el archivo crontab. Los usuarios pueden tener su propio crontab archivo, pero deben guardar sus nombres en el directorio */usr/lib/crontab/allow/*.

Hay seis campos en cada entrada del crontab. Estas entradas pueden estar separadas por un espacio en blanco, un tabulador o un salto de línea. Las entradas de izquierda a derecha son:
Minutos.- se especifican los minutos a los que será ejecutado el comando.
Hora.- Aquí se especifica la hora exacta (24 horas) en que el comando debe ser ejecutado.
Día.- Especifica el día exacto en que se ejecutara el comando.
Mes.- Especifica el mes exacto en que se ejecutara el comando.
Día de la semana.- Especifica el día de la semana en que se ejecutará el comando, donde 0 es Domingo y 6 es Sábado.
Comando a Ejecutar.- Este puede ser introducido desde el Bourne Shell

Los asteriscos son usados cuando no se desea insertar algún dato en cierto campo por ejemplo:

```
45 8 * * 6 /etc/ADM/sandra/bin/cacha_rata
```

Esto significa que el comando /etc/ADM/sandra/bin/cacha_rata se ejecutará todos los Sábados a las 8:45 de la mañana.

2.1.6 Cookies.

Las cookies constituyen una potente herramienta empleada por los servidores web para almacenar y recuperar información acerca de sus visitantes: conservan información entre páginas sucesivas que visita el usuario, extendiendo significativamente las capacidades de las aplicaciones cliente/servidor basadas en Web [VIL00]. Mediante el uso de cookies se permite al servidor recordar algunos datos concernientes al usuario, como sus preferencias para la visualización de las páginas de ese servidor, nombre y contraseña, productos que más le interesan o, simplemente, un identificador único.

Una cookie no es más que un archivo de texto simple que algunos servidores piden a un navegador que escriba en el disco duro. El contenido de la cookie lo dicta el servidor y normalmente consistirá en un número para identificar unívocamente al visitante. Este número se utiliza como índice en una gran base de datos, en la que se va almacenando la mayor cantidad posible de información acerca de lo que el visitante ha estado haciendo por sus páginas: qué enlaces sigue, qué páginas lee, qué fotos mira, qué documentos o programas descarga, etc. De esta forma, si el usuario apaga la computadora y se conecta de nuevo al día siguiente, la cookie permitirá identificarle, reconociéndole como el mismo usuario del día anterior, con lo que se puede continuar recabando información acerca de él.

En sí, esta técnica no parece muy preocupante. Desgraciadamente, a menudo se rellenan formularios con el nombre, apellidos, dirección, teléfono y a veces incluso datos aún más privados y sensibles

Afortunadamente, existen numerosas herramientas para restringir el uso de las cookies y el rastreo de los navegantes y para ocultar la identidad.

Lo que las cookies no pueden hacer

A pesar de que circulan muchas versiones sobre falsos virus por Internet, una creencia muy extendida entre los usuarios inexpertos es que las cookies pueden contagiar un virus. Para ello, es requisito indispensable que la cookie contenga código ejecutable, por un lado, y además que luego se le mande ejecutar. Hasta la fecha no se conoce una cookie ejecutable; en segundo lugar, aunque la hubiera, además debería existir una aplicación que la invocase para que el código destructivo se ejecutara, lo cual exigiría una ardua labor de programación. Una cookie de http tampoco puede ser usada para extraer datos de un disco duro En definitiva, una cookie no almacena más información confidencial que la que le se quiera dar al servidor que nos la envía.

Lo que las cookies sí pueden hacer

Las cookies son simplemente texto, que se puede editar perfectamente con cualquier editor ASCII, y como tales, son elementos pasivos que no pueden emprender ninguna acción. Sólo pueden ser leídos o escritos, pero no pueden ejecutarse ni mandar ejecutar ningún programa, por lo tanto no representan ninguna amenaza para la computadora ni pueden infectarlo con ningún virus.

Es importante comprender que por diseño las cookies poseen las siguientes limitaciones [VIL00]:

Trescientas cookies en total en el archivo de cookies. Si llega la número 301, se borra la más antigua. 4 Kbytes por cookie, para la suma del nombre y valor de la cookie. Veinte cookies por servidor o dominio (los hosts y dominios completamente especificados se tratan como entidades separadas y tienen una limitación de 20 cookies cada uno, no juntos). Ninguna máquina que no encaje en el dominio de la cookie puede leerla. Es decir, la información almacenada en una cookie no puede ser leída por una máquina distinta de la que la envió. Ahora bien, en Internet Explorer, esto no es del todo cierto debido a un agujero de seguridad que permite a cualquier sitio web visualizar la información almacenada en las cookies.

2.2 Seguridad en red

Cuando se conecta una máquina a una red, al mismo tiempo que se abre el sistema a innumerables fuentes de información y servicios[TOM94], queda también expuesta a todos los posibles ataques desde el exterior. El control de la red es uno de los aspectos fundamentales de seguridad, puesto que la mayoría de los intentos de ataque se llevan a cabo a partir de posibles agujeros existentes en los sistemas y servicios de red. A continuación se presentan los puntos más críticos de seguridad de un sistema abierto.

Clonación de máquinas

Un sistema puede pasar por otro asumiendo su dirección y el nombre del host. La dirección de red no es única para cada máquina. Es cierto que está almacenada en un chip de la tarjeta de red pero se puede cambiar usando el
comando:

```
% ifconfig le0 ether x:x:x:x:x:x
```

es posible, por tanto, adquirir la identidad de de otro través de otra máquina del mismo tipo.

Escuchando la red

En el caso de Ethernet cualquier sistema conectado a la red puede escuchar las conversaciones entre los otros sistemas. Si es un acceso remoto, entonces la contraseña puede ser visible. A esto se le conoce como ``snooping" (fisgonear). Los controladores ethernet, normalmente, leen todos los paquetes que le llegan pero los que tienen otro destino son descartados. Se puede hacer, sin embargo, que la tarjeta actué en modo promiscuo. En este caso interpretará todos los paquetes que le lleguen. En Solaris 2, existe el programa */usr/sbin/snoop* [SUN88], que se desarrolló para poder detectar y corregir problemas en la red y que lee los paquetes que llegan a nuestro sistema. Este programa es un riesgo de seguridad.

2.2.1 Autentificación de red

Debido a la posibilidad de escuchar los paquetes que circulan por la red, es importante que no se transmitan passwords sin encriptar por ella. Pero el hecho de enviar los passwords encriptados plantea el problema de la comunicación de las claves. Uno de los servicios que usan autentificación de red es el de RPC. Algunas aplicaciones que usan este servicio son Admintool y NIS+[SUN90b]. Las modalidades de autentificación que se pueden usar en Solaris 2 son:

AUTH_NONE	Sin autentificaci'on
AUTH_SYS	Identificaci'on basada en el NetID
AUTH_DES	Usa autentificaci'on DES
AUTH_KERB	Autentificaci'on *Kerberos*

Las características de seguridad que incluyen cada una de ellas son la siguientes:

a) Autentificación DES. Usa el estándar de encriptación DES y criptografía de llave pública para llevar a cabo la autentificación de usuarios y sistemas. En el sistema de criptografía de llave pública el cliente y el servidor obtienen una clave común denominada de conversación. El servidor deduce la clave conversación combinando la clave pública del cliente con su clave privada y viceversa. El cliente establece sus claves

públicas y privadas usando el comando keylogin, como root. Un usuario, normalmente, lo lleva a cabo de forma transparente cada vez que accede al sistema.

b) Autentificación Kerberos. Este sistema de autentificación fue desarrollado por el MIT dentro del proyecto Athena [ROS92] . Usa autentificación DES para encriptar unas etiquetas que se asignan cuando se accede al sistema. Duran un cierto periodo de tiempo, por defecto 8 horas. La clave debe mostrarse cada vez que el usuario accede a archivos.

2.2.2 Hosts compartidos

La existencia de lo que se denominan hosts compartidos permite la posibilidad de acceder, desde uno a otro, sin necesidad de introducir el password. Por esta razón son claramente inseguros. El archivo ***/etc/host.equiv*** [SUN90b] se usa para indicar los hosts compartidos. Si un usuario quiere hacer un rlogin o un rsh y tiene una cuenta en el sistema local con el mismo nombre, se le permite el acceso sin que se le requiera de nuevo el password. Hay que llegar a un compromiso entre funcionalidad y seguridad a la hora de la configuración de estos archivos. Nunca se deben compartir máquinas que son locales ni tampoco aquellas situadas en lugares públicos.

Un símbolo + en el mismo, indica que cualquier host conocido, será considerado como un host compartido. El archivo ***.rhosts*** , en el directorio local de cada usuario, permite hosts compartidos para una determinada combinación de máquina y usuario. Se usa normalmente para acceder a las cuentas en diferentes máquinas pertenecientes a distintas organizaciones, que no se incluyen en el archivo /etc/hosts.equiv. Este archivo presenta más problemas de seguridad que el otro ya que no lo controla el administrador.

2.2.3 Terminales seguras

En la versión de UNIX Solaris 2, aparece el concepto de terminal ``segura". Sólo se permitirá el acceso de root desde un terminal, cuando junto a la entrada correspondiente del mismo en el archivo ***/etc/ttytab***, aparezca la palabra secure. Por defecto en Sun todos los terminales se consideran seguras. [SUN90b]

La configuración más segura es eliminar la palabra secure de todas las entradas incluso de la consola. Con esto se consigue que el *root* deba entrar primero con su nombre de usuario y usar después el comando su. Así se puede llevar un control de quiénes han sido los que han entrado como root en el sistema.

2.2.4 Seguridad del sistema NFS

El NFS [SUN90b] se diseñó para permitir a los usuarios compartir (exportar en Solaris 1) los archivos a través de la red. Uno de los usos más comunes de NFS es para las estaciones sin disco. Hay que tener en cuenta varios aspectos para lograr que este sistema sea seguro. El archivo ***/etc/exports*** (en Solaris 1) es el archivo más importante en la configuración de NFS, en él se indica qué archivos se exportan. Un ejemplo de una entrada de este archivo puede ser la siguiente:

/export/swap/cliente1 -access = cliente1, root = cliente1

La entrada root indica los hosts a cuyos archivos se permite acceso por parte del super-usuario. Normalmente el NFS traslada el **id** del super-usuario a otro especial denominado ***nobody*** para prevenir que el root tenga derechos sobre los archivos de la máquina remota. Esto es bueno para la seguridad pero puede plantear un problema a la hora de la administración del sistema. No se debe dar nunca acceso de root a los archivos de NFS el *host* no compartido. La entrada access da una lista de hosts a los que se permite montar el sistema de archivos. Si no se indica, cualquier máquina de la red puede montar cualquier archivo vía NFS.

Otro archivo importante dentro de NFS es el /etc/netgroup [SUN90b] se utiliza para la definición de grupos de red. Este archivo se controla a partir de las ``páginas amarillas'' y se debe volver a construir en las mismas siempre que sea modificado. Como ejemplos de entradas en este archivo podemos tener:

```
Grupo_A   (servidora,,) (clientea1,,) (clientea2,,)
Grupo_B   (servidorb,,) (clienteb1,,) (clienteb2,,)
Todos     Grupo_A Grupo_B
```

Donde *Todos* es un supergrupo de los dos anteriores. Cada miembro de un grupo se define como una tripleta: (host, usuario, dominio). Si el campo de host o de usuario no aparece, entonces cualquier host o cualquier usuario pertenecen al grupo. Normalmente la mejor forma de configurar el archivo netgroup es crear un grupo de red para cada servidor y su cliente, y construir super grupos a partir de ellos. Esto da la flexibilidad de especificar el más pequeño grupo posible de hosts para cada sistema de archivos en ***/etc/exports***. (/etc/dfs/dfstab en Solaris 2). La forma de indicar, desde la línea de comandos, que un sistema de archivos se comparte es:

share -F nfs -o ro /export/applications share -F nfs -o rw /var/mail

donde los campos rw indican derecho de escritura y lectura y ro sólo lectura. Si el sistema va a compartirse tanto de escritura como lectura es buena idea compartir en nivel más bajo del directorio posible:

share -F nfs -o rw = support /export/support share -F nfs -o rw = sales /export/sales

mejor que:

share -F nfs -o rw /expot
Para el caso del cliente hay que evitar que la seguridad se vea comprometida, si se montan sistemas de archivos desde servidores no cualificados, puesto que se tiene poco control de los archivos que contiene. Un riesgo importante es cuando existen archivos ***setuid***.

El comando mount ofrece la posibilidad de controlarlos:

mount -o nosuid server: /export/export

si se usa esta opción todos los EID/GID de los archivos se cambian a UID/GID antes de que ser ejecutados.

Una aplicación sobre los RPCs seguros, es el NFS seguro. Para el caso normal, el NFS valida una petición de un archivo autentificando la máquina pero no el usuario. Cualquiera que tenga privilegios de root en el cliente NFS puede tomar cualquier identidad a través del comando su, si esta coincide con la de algún usuario del sistema montado adquiere sus privilegios. Con los NFS seguros los accesos se validad a partir de un identificador de red. Si un usuario no ha sido autentificado, entonces se le da la identidad del usuario nobody, con los permisos correspondientes.

2.2.5 Seguridad de NIS

NIS (Network Information System) [SUN88] fue desarrollado por Sun Microsystems para reducir el esfuerzo necesario para la puesta a punto y mantenimiento de las estaciones en UNIX. Se basa en la centralización de los archivos necesarios para la configuración de nuestra red, en un única máquina que sería el servidor NIS. Sólo será necesario actualizar los archivos de configuración en esta máquina para que los cambios tengan efecto en todas las que forman parte de nuestro dominio.

NIS [SUN90b] se basa en el protocolo RPC(Remote Procedure Calls) que usa el estándar XDR. Mediante RPC se consigue que un proceso pueda llevar a cabo llamadas a

funciones en máquinas remotas. El propósito de NIS es el establecimiento de una base de datos distribuida. El servidor NIS se construye de tal forma que pueda leer la base de datos pero no pueda actualizarla. Al protocolo NIS se le considera, un servicio público, seguro y que usa RPCs sin autentificación.

NIS se considera seguro puesto que no modifica los mapas directamente y los archivos pueden ser leídos por todo el mundo [HSP91]. Esto desde el punto de vista del servidor, pero cambiando hacia el cliente, la seguridad es bastante pobre. Puesto que NIS no utiliza autentificación a nivel de RPC, cualquier máquina de la red puede enviar una respuesta falsa, y pasar por el servidor de NIS. Para que el ataque tenga éxito, el paquete con la respuesta falsa debe llegar al cliente antes de que responda el servidor. También debe ser capaz de observar la petición y generar la respuesta. Para esto es preciso que la máquina que lleva a cabo el ataque esté situada en el mismo camino de comunicación del servidor y el cliente. Los problemas de seguridad que resultan se puede crear fácilmente un mapa NIS para sustituir al del servidor. Un intruso podrá acceder al cliente NIS a través de los comandos de acceso remoto, cambiando en el mapa hosts.byaddr la dirección de la máquina que lleva a cabo la intrusión a otra máquina que se encuentre el archivo /etc/hosts.equiv o en uno de los archivos .rhosts de los usuarios.

La solución de estos problemas de seguridad es emplear autentificación para el protocolo RPC [TOM94]. La más segura de las autentificaciones es la DES. Con Solaris 2 llega el servicio NIS+ que elimina la mayor parte de problemas de seguridad que NIS planteaba. NIS+ utiliza RPCs seguros si se configura de forma adecuada. En ocasiones por rapidez de instalación se configura en el nivel 0. La forma de indicar el nivel es:

rpc.nisd -S <nivel>

El daemon se lanza desde /etc/rc2.d/S71lrpc. La gestión de los mapas de NIS+, se lleva a cabo mediante Admintool. Es conveniente inspeccionar el archivo nsswitch.conf, dónde se indica si programas del sistema, como login, usarán los archivos locales de /etc, o bien, los mapas de NIS+, o ambos y en qué orden.

2.2.6 Seguridad del DNS

El servicio DNS(Domain Name System), es una amplio conjunto de bases de datos distribuida [SCH93] usado a lo largo de la internet, proporcionando correspondencia entre los nombres de host y las direcciones de IP de los mismos. Existe la posibilidad de abusar de este servicio para poder entrar en un sistema.

Suposiciones durante la fase de autentificación, pueden conllevar serias grietas de seguridad importantes. El problema de seguridad es similar al que existe en el NIS. La autentificación se lleva a cabo en muchos casos a partir del nombre o la dirección. Las aplicaciones de alto nivel, usan en la mayoría de los casos los nombres para la autentificación, puesto que las tablas de direcciones son mucho más difíciles de crear, entender y mantener. Por ejemplo, si alguien quiere suplantar una máquina por otra no tiene más que cambiar una de las entradas de la tabla que relaciona su nombre con su dirección. Este es el problema fundamental de DNS. Para conseguir esto una máquina debe obtener primero el número ID de la petición DNS, para ello debe construir el paquete de respuesta y usar la opción de enrutamiento de fuente, para hacerlo llegar al que llevó a cabo la petición [SMA88] [SUN88].

2.2.6 FTP

El protocolo de transferencia de archivos (FTP), se implementa mediante los programas ftp y ftpd[SUN88]. Las viejas versiones de los mismos presentaban ciertos fallos de seguridad. Para el ftpd, si se dispone de una versión de antes de diciembre de 1988, es preciso cambiarla. Una de las posibilidades que ofrece este servicio es el FTP anónimo, para la distribución de software público. Existía por ejemplo, un antiguo bug del ftp mediante el cual conseguíamos accesos de root [FV86]:

```
% ftp -n
ftp> open www.udlap.mx
Connected to www.udlap.mx
220 www.udlap.mx FTP server ready
ftp> quote user ftp
331 guest login ok, send ident as password.
ftp> quote cwd ~root
530 Please login with USER and PASS
ftp> quote pass ftp
230 Guest login ok, access restrictions apply.
```

si esto funciona habremos entrado como root en el sistema.

2.2.7 El servicio de correo electrónico

El correo electrónico es uno de los servicio más usados. El programa sendmail [SUN88] se usa para recibir y mandar los mensajes de correo electrónico. Este programa tiene gran cantidad de bugs ya que no hay ningún programador que logre comprenderlo

del todo. Antiguas versiones tienen agujeros bien conocidos. Son necesarias, también, ciertas medidas de seguridad a la hora de su instalación [CUR90]:

a) Eliminar el alias de ``decode'' para los archivos /etc/aliases y /usr/lib/aliases.
b) Si se crea un alias para que los mensajes sean enviados a programas, hay que estar completamente seguro de que es imposible obtener un shell o enviar un mensaje a un shell de esos programas.
c) Asegurarse que el password ``withard'' se elimina del archivo de configuración, sendmail.cf.
d) Asegurarse que el sendmail no tenga el comando ``Debug''.

2.2.8 El servidor inetd y el daemon in.routed

Es el servidor de red y tiene toda una lista de servicios a la espera de recibir respuesta de sus peticiones. Algunos servicios pueden deshabilitarse para mejorar la seguridad. Los servicios que se quieren dar se incluyen en el archivo */etc/inetd.conf*. La mayor parte de ellos corren como root y por tanto si alguien consigue hacerse con su control el resultado puede ser serio. Algunos de los más importantes son:

in.fingerd: Permite a los usuarios remotos listar todos aquellos conectados al sistema. Los hackers lo utilizan para ver si el administrador del sistema está en él.

rcp.rexd: Permite la los usuarios ejecutar comandos en máquinas remotas con muy poca autentificación. Es mejor usar en su lugar /usr/bin/rsh que es mucho más seguro.

in.tfpd: Una versión reducida de ftp, con poca autentificación.

admina: Usado por admintool. Corre con setuid = root. Lo mejor es deshabilitarlo si no se administran máquinas de forma remota, o al menos hacerlo con la opción -S.

El daemon **in.routed** [TOM94] gestiona automáticamente las tablas de enrutamiento. Se intercambia con otra máquinas de la red intercambiando información de enrutamiento, usando el protocolo RIP (Routing Information Protocol). Lo que hace es indicar a las máquinas que lo solicitan, la dirección de aquellas que conoce. Si la máquina tiene una única red conectada no es necesario que esté corriendo. Es mucho mejor decir, en cuanto a mejora del rendimiento, a la máquina explícitamente dónde enviar los paquetes no locales. Esto se consigue estableciendo un router por defecto:

/sbin/route -f add default <comms_server_name> 1

Desde el punto de vista de seguridad esta no es la forma idónea. Si se usa más de un gateway se puede mantener manualmente la tabla de enrutamiento con:

/sbin/route add net <red> <gateway> 1

Donde <red> es uno de los nombres de /etc/networks , y <gateway> es el nombre de la máquina local que conecta la red.

2.3 Supervisión en Unix. *Auditing*

Un requisito importante de seguridad es la posibilidad de visualizar qué es lo que está pasando y qué ha pasado en un sistema dado [TOM94]. Para ello hay que controlar los archivos a los que se acceda de forma no autorizada, así como la supervisión del propio sistema para detectar cualquier agujero de seguridad.

2.3.1 Supervisión del sistema de cuentas.

Se debe llevar un seguimiento del sistema de cuentas para comprobar dos sucesos: usuarios que han accedido al sistema cuando no deberían estar conectados (por la noche, durante las vacaciones...) y programas que los usuarios normalmente no deberían estar usando. A continuación se muestran los archivos que ofrece el sistema para llevar a cabo este control. [VIL00]

El archivo lastlog

El archivo /usr/adm/lastlog [SUN88] almacena la hora del más reciente acceso del usuario al sistema y desde dónde se ha llevado a cabo el acceso. Este archivo es el que usa la orden finger. Un usuario podría conocer si alguien ha entrado en su cuenta, recordando cuándo fue la última vez que accedió al sistema.

```
# finger sandra
Login name: sandra                      In real life: MURILLO CANO SANDRA R
Directory: /archivos/vol07/sandra       Shell: /bin/csh
Last login Sun Mar 11 21:34 on pts/0     from acadaplic.pue.udlap.mx
No unread mail
No Plan.
```

El archivo sulog

Este es un archivo de texto (/var/adm/sulog) donde se registran las ejecuciones de la orden su indicando fecha, hora,usuario que lanza el proceso y la identidad que alcanza, terminal asociada y éxito (+) o fracaso (-)

```
#more /var/adm/sulog
SU 01/13 08:57 + pts/9 root-osvaldo
SU 01/13 09:34 - pts/9 root-sandra
SU 01/13 09:53 + pts/13 enrique-operador
SU 01/13 10:10 + pts/10 root-is107375
SU 01/13 11:03 + pts/9 root-agonza
SU 01/13 11:41 - pts/11 rcastro-root
SU 01/13 11:45 + pts/11 rcastro-root
```

Los archivos utmp y wtmp

El archivo */etc/utmp* [SUN88a] se utiliza para la visualización de aquellos conectados al sistema en cada instante. La orden ***who*** muestra por pantalla este archivo. Aunque habitualmente este archivo está situado en */var/adm/* , junto con otros archivos de *log*, es posible encontrarlo en /etc en algunas versiones. [VIL00]. Para cada usuario se da el nombre, la terminal usada, el momento de acceso al sistema, y el host remoto si se conecta vía red. El archivo */usr/adm/wtmp* [SUN88], almacena cada acceso al sistema y cada salida del mismo. Se puede mostrar con who:

```
  % who /usr/adm/wtmp

is109655   pts/22        Apr  2 22:01     (sun19sala4.pue.udlap.mx)
is109655   pts/28        Apr  2 22:04     (acadaplic.pue.udlap.mx)
is113456   pts/20        Apr  2 22:09     (tiranoray)
udlasc     pts/22        Apr  2 22:10     (tiranoray)
is113345   pts/21        Apr  2 22:11     (acadaplic.pue.udlap.mx)
co100166   pts/22        Apr  2 22:11     (tiranoray)
```

En el caso de que varios usuarios entren al sistema al mismo tiempo, es posible, que los tiempos de acceso y abandono se mezclen dando una salida confusa. Este archivo también se puede visualizar utilizando la orden last. En las nuevas versiones del sistema operativo se incluyen también los ``pseudoaccesos'' como el ftp.

```
acadaplic#  last -10
fa109374         pts/27   sun02sala4.pue.udlap.mx      Tue Apr  3 17:21   still logged in
em108536         pts/26   sun21sala4.pue.udlap.mx      Tue Apr  3 17:18 - 17:23  (00:04)
adminfo          pts/15   ssray2                       Tue Apr  3 17:15   still logged in
root             pts/25   dns-sec.pue.udlap.mx         Tue Apr  3 17:13   still logged in
```

```
ua012839     pts/25   ssray2                Tue Apr  3 17:13 - 17:13  (00:00)
adminfo      pts/22   ssray2                Tue Apr  3 17:05   still logged in
ii107424     pts/4    mailweb               Tue Apr  3 17:02   still logged in
```

Estos archivos ya son obsoletos en Solaris 8 y en su lugar existen los archivos **utmpx** y **wtmpx**.

El archivo acct

El archivo */etc/adm/acct* almacena cada ejecución de un proceso en el sistema, quién lo ejecutó, cuándo y cuanto tiempo tardó en ejecutarse. La información se registra cuando se completa el comando pero siempre que se haya compilado el kernel con la opción SYSACCT, que por defecto no está activada. Este archivo puede visualizarse mediante el programa ***lastcomm*** [SUN88]. Sin argumentos se muestra toda la información. Se puede limitar a un determinado programa, a un usuario, o a una determinado terminal.

2.3.2 Supervisión de la red

La tarea es bastante difícil, porque existen muchas maneras de romper su seguridad. Existen varios programas a disposición del administrador para ayudarle en esta labor.

syslog

El syslog [SUN88], es un mecanismo mediante el cual se manda cualquier mensaje de error a la consola del sistema. Normalmente los mensajes se almacenan en /usr/adm/messages, con la fecha, el tiempo que aparecieron, nombre del programa que envió el mensaje, y el identificador del proceso.

Son de particular interés los mensajes del los programas login y su. Se debe suprimir la posibilidad de que alguien entre directamente como root, porque no se conoce quién es el que está usando la cuenta. También se almacena cualquier caso en el que una persona intente repetidas veces entrar en una determinada cuenta sin éxito. Si se encuentran mensajes de estos, es posible que haya un cracker intentando descubrir los passwords de los usuarios. También se almacena los momentos en los que se usa el comando su, para root o para otro nombre de usuario. Con ello se puede observar si hay usuarios que comparten los passwords, o para ver si existe un hacker que ha entrado en el sistema y pretende acceder a otras cuentas.

El comando showmount

Este comando se puede usar en un servidor de NFS, para ver todos los hosts que tienen algún sistema de archivos montado sobre el servidor. Con la opción -a se muestra tanto el host, como los directorios [SUN90a].

```
webserver.pue.udlap.mx:/home/mx/vol25
xnic.pue.udlap.mx:/home/mx/acadesw
xnic.pue.udlap.mx:/home/mx/vol22
zeus.pue.udlap.mx:/home/mx/acadesw
```

La salida de showmount puede usarse para dos asuntos, la primera es que sólo máquinas locales al sistema deben estar presentes. Sólo se deben encontrar los directorios habituales. Si existen directorios que no lo son se debe encontrar quién los está montando y por qué, aunque es una manera inocente de entrar en el sistema, puede significar un intento de rotura de la seguridad.

El comando npasswd

El comando npasswd [CUR90], desarrollado por Clyde Hoover, pretende sustituir al comando passwd estándar en UNIX. Este programa hace que los passwords sean más seguros evitando que los usuarios elijan unos ``malos''. Alguna de las opciones que presenta este programa son las siguientes:

- Longitud mínima del password.
- Permite que se pueda obligar a los usuarios a mezclar mayúsculas y minúsculas.
- No aceptar palabras simples, como una letra repetida.
- Rechazar passwords que contengan el nombre de la máquina u otra información relacionada con ella.
- Comprueba si la palabra elegida contiene el nombre de cuenta, o el nombre de pila o apellidos.
- Comprueba si la palabra está incluida en algunos de los diccionarios, incluido el del sistema.
- Características de seguridad del sistema C2 de SUN: [DEP85] [SUN90a]:

- Almacenamiento de todos los instantes en los que un usuario entra y abandona el sistema, permite la ejecución de algunos comandos de administración y la ejecución de ciertas operaciones privilegiadas.
- Un mecanismo con el que se consigue proteger los passwords encriptados, guardándolas en el archivo /etc/shadow, sin permisos de lectura.
- Posibilidad de usar un sistema de encriptación DES.

2.3.3 Supervisión del sistema de archivos

Las tareas mas sobresalientes que se deben llevar a cabo en el control del sistema de archivos son:

- Buscar archivos que pueden modificarse por usuarios no autorizados.
- Detectar cambios no autorizados.
- Poder recuperar el sistema, dentro de lo posible, después de que se hayan dado cambios no autorizados.

A continuación se presentan alguanas herramientas para llevar a cabo dicha tarea:

El comando find

El comando find es un comando de búsqueda de propósito general. Usando varios argumentos y la comparación de patrones basándose en el nombre de archivo, tipo, modo, propietario, modificación, fecha, y otras características, se puede conseguir obtener información importante en cuanto a la seguridad del sistema de archivos. Algunas de las características que podemos buscar son:

a) Encontrar archiovos Setuid y Setgid. Estos programas confieren derechos especiales a los usuarios que los ejecutan, es necesario comprobar que no hay programas inseguros instalados. Se deben seguir especialmente los programas que tengan como dueño efectivo al root. Un truco que usan los crackers es una vez que han entrado en el sistema, dejar uno de estos programas pudiendo así volver a entrar en el sistema cuando lo deseen. La forma de llevar a cabo la búsqueda es la siguiente:

```
# find / -type f -a \(-perm -4000 -o -perm -2000 \) -print
```

algunas de estas opciones son:

/.

Indica el directorio desde dónde va a dar comienzo la búsqueda.

-type f.

Sólo examina los archivos normales, para directorios es la opción ``d", para ligas simbólicos ls ``l", la ``c" para dispositivos de caracteres especiales, y la ``b" para dispositivos de bloques especiales.

-a.

Quiere decir ``y". Esto es que se busque los archivos del tipo indicado y con la condición de permisos que se incluye a continuación.

\(-perm -4000 -o -perm -2000 \).

Los paréntesis se utilizan para agrupar. El perm -4000 indica que se busquen aquellos archivos con el bit ``4000" activo (en hexadecimal). Es el bit des ``se-user-id". El bit ``2000" es análogo al anterior sólo que para los grupos. La -o indica ``o".

-print.

Indica que se impriman todos los archivos que coincidan con las características anteriores.

b) Encontrar archivos modificables por cualquiera. Estos archivos pueden ser un agujero de seguridad si un ``cracker" consigue entrar en el sistema y los modifica. Para llevar a cabo la búsqueda a partir del comando find:

```
# find / -perm -2 -print
```

Dónde el **-2** indica el bit de permisos de escritura de un archivos. Esta búsqueda es larga e incluye ciertos archivos que deben poderse escribir por todo el mundo, por ejemplo, los del directorio /dev, al igual que los permisos sobre las ligas simbólicas que no tienen significado.

c) Encontrar archivos sin dueño. El encontrar archivos con propietario desconocido, puede ser una indicación que un cracker ha conseguido entrar en nuestro sistema. La forma de llevar a cabo la búsqueda a partir del comando find es la siguiente:

```
# find / -nouser -print
```

La opción **-nouser** comprueba que el propietario del archivo se encuentre en el archivo de passwords del sistema. También se puede hacer lo mismo para el caso de los grupos con la opción -nogroup.

d) Búsqueda del archivo .rhosts. Como se ha visto, puede ser conveniente no permitir que los usuarios tengan un archivo .rhosts en sus directorios. Mediante el comando find:

```
# find /home -name .rhosts -print
```

Checklist

El empleo de estas listas puede ser una herramienta muy útil a la hora de descubrir cambios no autorizados del sistema de directorios. Un cheklist son cada uno de los nombres de los archivos que forman parte de un grupo de directorios [CUR90], su tamaño, propietarios, fechas de modificación, y otros datos que pueden ser de interés a la hora de comprobar si se ha modificado de forma no autorizada el sistema de archivos. Para llevar un control de este tipo basta con usar los comandos estándares de UNIX ls y diff. La forma de proceder es la siguiente, primero se usa el comando ls para generar una lista ``maestra''. El mejor momento de hacerlo es justo después de la instalación del sistema operativo. Una forma sencilla de llevarlo a cabo es la siguiente [MAR93]:

```
# ls -aslgR /bin /etc /usr > ListaMaestra
```

Es conveniente borrar algunas líneas correspondientes a algunos archivos que se usan a menudo, como el /etc/utmp y el /usr/adm/acct. Para crear una lista con la situación actual del sistema de archivos basta ejecutar la línea anterior con otro nombre. Una vez hecho esto podemos pasar a compararlos con el comando diff:

```
# diff ListaMaestra ListaActual
```

Las líneas que están sólo en el primer archivo se imprimirán con un ``<'' delante y las que están únicamente en el segundo archivo se imprimen precedidas de ``>''. Con una construcción adecuada de las listas periódicamente, es fácil detectar posibles cambios no autorizados.

2.4 Detección de intrusos

2.4.1 Tipos de intrusiones

Es cuestión importante detectar la intrusión de personas indeseables o no autorizadas en el sistema lo antes posible para que cause el menor daño posible. Según [HLMS90] una intrusión se define como:

Cualquier conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de una información o un recurso.

Normalmente una intrusión intenta[KS94]:

- Acceder a una determinada información.

3 Manipular cierta información.

4 Hacer que el sistema se no funcione de forma segura o inutilizarlo.

Una intrusión es la violación de la política de seguridad del sistema. Los intrusos pueden utilizar fallos en la arquitectura de los sistemas y el conocimiento interno del sistema operativo para superar el proceso normal de autentificación. Existen diferentes tipos de intrusiones que pueden afectar a un determinado sistema[SMA88]:

Intentos de entrada: Una persona ajena a nuestro sistema intenta acceder de forma no autorizada al mismo. Se detectan normalmente por modelos de comportamiento atípicos, o violaciones de las restricciones dadas por la política de seguridad.

Ataque enmascarado: A partir de un usuario del sistema se intenta un ataque al mismo, es detectado a partir de modelos de comportamiento atípico o violaciones de constraints de seguridad.

Penetraciones en el sistema de control: Son normalmente detectadas a partir de la observación de modelos especiales de actividad.

Fuga: Se utilizan de manera excesiva los recursos de un sistema. Se detectan normalmente por usos anormales de los recurso de E/S.

Rechazo de servicio: Detectados por uso atípico de los recursos del sistema.

Uso malicioso: Detectado normalmente por modelos de comportamiento atípico, violaciones de las restricciones de seguridad, o uso de privilegios especiales.

Esta clasificación se basa en el efecto de las intrusiones y la forma de llevarlas a cabo.

2.4.2 Detección de intrusos a partir de comportamiento anómalo.

Se detectan a partir de la caracterización anómala del comportamiento y del uso que hacen de los recursos del sistema. Este tipo de detección pretende cuantificar el comportamiento normal de un usuario. Hay que tener en cuenta las tres distintas posibilidades que existen en un ataque atendiendo a quién es el que lo lleva a cabo [FRA94]:

- Penetración externa. Que se define como la intrusión que se lleva a cabo a partir un usuario o un sistema de computadores no autorizado.
- Penetraciones internas. Son aquellas que llevan a cabo por usuarios autorizados de sistemas de computadoras que no están autorizados al acceso a los datos que se están siendo comprometidos.
- Abuso de recursos. Se define como el abuso que un usuario lleva a cabo sobre unos datos o recursos de un sistema al que está autorizado su acceso.

La idea central de este tipo de detección es el hecho de que la actividad intrusiva es un subconjunto de las actividades anómalas [KS94]. Esto puede parecer razonable por el hecho de que si alguien consigue entrar de forma ilegal en el sistema, no actuará como un usuario comprometido, seguramente el comportamiento se alejará del de un usuario normal [FV86].

Sin embargo en la mayoría de las ocasiones una actividad intrusiva resulta del agregado de otras actividades individuales que por sí solas no constituyen un comportamiento intrusivo de ningún tipo. Idealmente el conjunto de actividades anómalas es el mismo del conjunto de actividades intrusivas, de todas formas esto no siempre es así:

a) Intrusivas pero no anómalas. Se les denomina falsos negativos o errores de tipo I. En este caso la actividad es intrusiva pero como no es anómala no conseguimos detectarla. Se denominan falsos negativos porque el sistema erróneamente indica ausencia de intrusión.
b) No intrusivas pero anómalas. Se denominan falsos positivos o errores de tipo II. En este caso la actividad es no intrusiva, pero como es anómala el sistema decide que es intrusiva. Se denominan falsos positivos, porque el sistema erróneamente indica la existencia de intrusión.
c) Ni intrusiva ni anómala. Son negativos verdaderos, la actividad es no intrusiva y se indica como tal.
d) Intrusiva y anómala. Se denominan positivos verdaderos, la actividad es intrusiva y es detectada.

Los primeros no son deseables, porque dan una falsa sensación de seguridad del sistema, el intruso en este caso puede operar libremente en el sistema. Los falsos positivos se deben de minimizar, en caso contrario lo que puede pasar es que se ignoren los avisos del sistema de seguridad, incluso cuando sean acertados. Los detectores de intrusiones anómalas requieren mucho gasto computacional, porque se siguen normalmente varias métricas para determinar cuánto se aleja el usuario de lo que se considera comportamiento normal [HLMS90].

2.5 Criptología.

Criptografía. Definición clásica.
(kriptos - oculto, graphos - escribir)
Arte de escribir mensajes en clave secreta o enigmáticamente. [CAB96]

2.5.1 Breve historia.

El primer caso documentado de escritura secreta data del siglo V a.C., durante la guerra entre Atenas y Esparta. El cifrado se basaba sólo en la alteración del mensaje mediante la inclusión de símbolos innecesarios. Otro caso proviene de la época de los romanos (siglo I a.C.). El cifrado consistía en una sustitución de determinados símbolos por otros según una regla fija (método César)

Para el siglo XIV, Cicco Simoneta en su libro "Liber Zifrorum" estudia diversos sistemas basados en simples sustituciones de letras. Un siglo después, Alberti se destaca en el criptoanálisis y por ello se le nombra el padre de la criptología. En 1586 Blaise de Vigenere publica "Traicté des Chiffres" donde recoge diferentes métos utilizados en la época. En el siglo XIX se utiliza un método basado en la reordenación de los símbolos del mensaje, llamado trasposición, que junto con la sustitución constituye la base de los cifrados clásicos.

En 1949 la criptografía empezó a ser considerada como una ciencia aplicada debido a su relación con otras ciencias, como la estadística, la teoría de números, la teoría de información y la teoría de la complejidad computacional. [CAB96]

2.5.2 Criptografía.

La criptografía corresponde solo a una parte de la comunicación secreta. Si se requiere secreto en la comunicación es porque existe desconfianza de intercepción por un "enemigo". Este enemigo utilizará todos los medios para descifrar esos mensajes

mediante un conjunto de técnicas y métodos que constituyen una ciencia conocida como criptoanálisis.

La criptología se denomina al conjunto de **criptografía** y **criptoanálisis**. En toda comunicación se presenta una lucha entre criptógrados y criptoanalistas. Un éxito de unos representa siempre un fracaso de los otros. [CAB96]

2.5.3 Criptografía Clásica.

2.5.3.1 Método César

Éste fue un método utilizado por los romanos en la edad antigua, y el cual se le atribuye supuestamente a César, de aquí su nombre. [VIL00]

El sistema es el más conocido y sobre el primero que se empezó a pensar detenidamente como utilizar un sistema para esconder información adesconocidos.

El sistema se basa en un simple desplazamiento de las letras en el abecedario:

A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9

Pues sea el mensaje HOLA y la clave K=-3

El mensaje criptado sería EMIX, esto sería HOLA encriptado.

Es un sistema bastante simple y sencillo, además es fácilmente detectable para el criptoanalista y fácil de descifrar.

Solo debemos tener el abecedario , el mensaje y la clave, tanto para encriptarlo como para desencriptarlo. Al igual que la clave pudiera ser - , tambien puede ser +, por lo que el desplazamiento de las letras sería hacia delante.

Si la clave es -, para encriptar sería retroceder en el abecedario, pero para desencriptar, sería avanzar en el abecedario, y si la clave fuera +, seria para encriptar desplazar hacia delante en el abecedario y - para desencriptar.

2.5.3.2 Cifrado de Gronsfeld

El cifrado de Gronsfeld usa más de un alfabeto cifrado para poner en clave el mensaje y que se cambia de uno a otro según se pasa de una letra del texto en claro a otra. [MIT94]

Es decir que deben tenerse un conjunto de alfabetos cifrados y una forma de hacer corresponder cada letra del texto original con uno de ellos. Para dejar esto más claro veamos una de las tablas que se usaban antes de la era de los ordenadores para hacer un cifrado de este tipo. (Figura 2.1)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0:	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
1:	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
2:	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
3:	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
4:	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
5:	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
6:	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
7:	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
8:	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
9:	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

Figura 2.1 Tabla ejemplo para Cifrado de Gronsfeld

Ahora se toma una serie de dígitos como clave. Suponga 1203456987. Se utiliza la tabla para el cifrado del mensaje:

EMBARCAMOS AL ANOCHECER

Se escribir la clave debajo del texto original las veces que sea necesario

EMBARCAMOS AL ANOCHECER
1203456987 12 034569871

Se sustituye cada letra por la correspondiente del alfabeto que indica el número debajo suyo.

HRDHCPROLL DQ CUZPYGZXU

El cifrado de Gronsfeld altera la frecuencia de las letras del texto, pues por ejemplo la letra más corriente en español, la E, se cifra de forma diferente según su posición en el texto original.

2.5.3.3 Cifrado Bífido

El método Bífido es un cifrado fraccionario. Es decir que cada letra viene representada por una o más letras o símbolos, y donde se trabaja con estos símbolos más que con las letras mismas. [GAR91]

El método comienza con la utilización de un alfabeto desordenado en un cuadrado 5x5. Observa un ejemplo donde la clave para el alfabeto desordenado es DIPLOMA: (Figura 2.2)

*	1	2	3	4	5
1	D	IJ	P	L	O
2	M	A	B	C	E
3	F	G	H	K	N
4	Q	R	S	T	U
5	V	W	X	Y	Z

Figura 2.2 Tabla para ejemplo de Cifrado Bífido.

Al ser un cuadro sólo de 5x5 obliga a cifrar de la misma forma la I y la J. El contexto permitirá distinguir cual de las dos letras se pretendía cifrar.

Para cifrar el texto en claro se escriben los equivalentes numéricos de cada letra, utilizando sus "coordenadas". Si por ejemplo el texto en claro es:

VEN A LAS TRES

El equivalente numérico es:

51 25 35 22 14 22 43 44 42 25 43

Que dividido en dos partes queda:

5 1 2 5 3 5 2 2 1 4 2

2 4 3 4 4 4 2 2 5 4 3

Si ahora leemos los números como columnas en lugar de por filas resulta:

52 14 23 54 34 54 22 22 15 44 23

Que volviendo a consultar la tabla resulta en el mensaje cifrado:

WLBYKYAAOTB

Este método altera la frecuencia de los caracteres a diferencia de lo que ocurre por ejemplo con los cifrados monoalfabéticos. Admite algunas variaciones como por ejemplo dividir la lista en 3,4,...n partes.

2.5.4 Criptografía de Clave Secreta.

2.5.4.1 DES

DES es el Estándar para el Cifrado de Datos, un algoritmo originalmente desarrollado por IBM y posteriormente modificado y respaldado por el gobierno de los Estados Unidos (estándar al menos hasta 1988). Es el más estudiado y el mas utilizado de todos los algoritmos de llave única. [CAB96] [DIF92]

DES cifra bloques de 64 bits a la vez, utilizando una llave de 56 bits. (Originalmente era de 128 bits, pero IBM nunca hizo público el por qué de la reducción)

Seguridad DES

DES puede ser atacado utilizando todas las llaves posibles (2^56). O(2^55) en promedio. Sin embargo, se dice que NSA modificó el algoritmo para hacerlo más débil que el original. A pesar de eso, aún no ha sido posible romperlo.

Una computadora especializada para romper DES, con un costo estimado en U$$1 M. tomaría 3.5 hrs. en romper un mensaje. [CAB96] Ha habido intentos que han disminuido la complejidad de romper DES hasta 2^43, pero han resultado poco prácticos.

2.5.4.2 Triple DES

Utilizando Triple-DES, el texto es cifrado tres veces. Hay varias formas en que esto se hace [CAB96] [LOU91]:

DES-EEE3. Tres veces se cifra, con tres llaves diferentes.
DES-EDE3. Se cifra, descifra y cifra, con tres llaves diferentes.

DES-EEE2 y DES-EDES2. Similar a los anteriores, pero se utilizan para el primero y último la misma llave.
Se estima que la forma más segura de DES es con tres llaves diferentes. En el caso del algoritmo de fuerza bruta, se requieren O(2^112) pasos por romperlo [HUN91].

2.5.4.3 IDEA

International Data Encryption Algorithm, desarrollado por Lai y Massey [CAB96] [AND93] [HSP91]. Es un algoritmo iterativo que trabaja sobre bloques de 64 bits con llaves de 128 bits.

Es fácil implementar hardware y software, aunque es similar en velocidad a DES. Se cree que IDEA es inmune a análisis diferencial, lo que lo hace más seguro que DES. Daemon reportó que hay 2 ^ 51 llaves débiles en IDEA.

2.5.4.4 Rijndael (Un nuevo estándar de cifrado)

En 1996, el Instituto Nacional de Estándares y Tecnología (National Institute of Standards and Technology, NIST) dió los primeros pasos para la consolidación de un Estándar de Cifrado Avanzado (Advanced Encryption Standard, AES). [VIL00] Su objetivo fue desarrollar una especificación para encontrar un algoritmo de cifrado que sustituya al agonizante DES (56 bits de clave, por lo que resulta inseguro actualmente, y diseño orientado a hardware, por lo que resulta relativamente ineficiente en software), de manera que el nuevo algoritmo sea capaz de proteger la información sensible de los ciudadanos y del gobierno hasta bien entrado el siglo XXI. Se espera que el algoritmo seleccionado sea utilizado por el Gobierno de Estados Unidos y voluntariamente por el sector privado, en sustitución de DES. Por extensión, presumiblemente sería adoptado por el resto de países del mundo [AND93].

En septiembre de 1997, se realizó una convocatoria para la presentación de algoritmos, en la que se especificaron los siguientes criterios de evaluación y requisitos mínimos de aceptación: [VIL00]

1. Ser público.
2. Ser un algoritmo de cifrado en bloque simétrico.
3. Estar diseñado de manera que se pueda aumentar la longitud de clave según las necesidades.
4. Ser implementable tanto en hardware como en software.
5. Estar
a) disponible gratuitamente, o

b) disponible bajo términos consistentes con la política de patentes del Instituto Nacional Americano de Estándares (American National Standards Institute, ANSI).
6. Los algoritmos que cumplan con los requisitos arriba citados serían juzgados de acuerdo con los siguientes factores:

a) seguridad (e.d., el esfuerzo necesario para criptoanalizarlos),
b) eficiencia computacional,
c) requisitos de memoria,
d) adecuación hardware y software,
e) simplicidad de diseño,
f) flexibilidad y
g) requisitos de licencia.

Los algoritmos propuestos tenían además que soportar obligatoriamente una longitud de bloque de 128 bits y una longitud de clave de 128, 192 y 256, al margen de cualesquiera otras longitudes posibles. La razón por la que se ha exigido la posibilidad de utilizar claves de 192 y 256 es que a pesar de que una clave de 128 bits es totalmente "imposible" de reventar probando todas las posibles combinaciones con los recursos computacionales actuales, no puede asegurarse que en un futuro próximo el estado del arte de la computación no vaya a permitir reventar claves de esa longitud. Dado que el estándar se desea que pueda utilizarse hasta bien entrado el siglo XXI, por lo menos hasta el año 2060, se debe realizar una elección conservadora.

Los últimos avances teóricos en computación cuántica hacen temer que en un plazo breve, si se construyesen estos ingenios, los problemas numéricos hoy considerados irresolubles en un tiempo razonable, como la factorización o los logaritmos discretos, se podrán resolver mucho más rápido (en un tiempo igual a la raíz cuadrada del empleado por una computadora digital tradicional), tirando por tierra la seguridad de los algoritmos criptográficos con longitudes de claves actuales. La raíz cuadrada del número de todas las claves posibles de 256 bits (esto es, 2^256) es igual al número de todas claves posibles de 128 bits (esto es, sqrt(2^256)=2^(256/2)=2^128). En otras palabras, asumiendo las velocidades previstas, una máquina cuántica tardaría el mismo tiempo en realizar una búsqueda exhaustiva en un espacio de claves de 256 bits que una computadora digital actual en un espacio de claves de 128 bits, y por lo tanto se trataría de una tarea irrealizable.

El 2 de octubre del 2000, el NIST anunció el algoritmo ganador: Rijndael, propuesto por los belgas Vincent Rijmen y Joan Daemen (de ahí su nombre). Rijndael es un cifrador de bloque que opera con bloques y claves de longitudes variables, que pueden ser especificadas independientemente a 128, 192 ó 256 bits. El resultado intermedio del

cifrado se denomina Estado, que puede representarse como un array rectangular de bytes de cuatro filas. La transformación que tiene lugar en cada vuelta de cifrado está compuesta a su vez de cuatro transformaciones diferentes:

a) Sustitución de bytes no lineal, operando independientemente sobre cada uno de los bytes del Estado.
b) Desplazamiento de las filas del Estado cíclicamente con offsets diferentes.
c) Mezcla de columnas, que se realiza multiplicando las columnas del Estado módulo x^4+1, consideradas como polinomios en GF(2^8), por un polinomio fijo c(x).
d) Adición de la clave de vuelta, en la que se aplica al Estado por medio de un simple XOR. La clave de cada vuelta se deriva de la clave de cifrado mediante el esquema de clave.

El esquema de clave consiste en dos operaciones: expansión de clave y selección de clave de vuelta de cifrado. El proceso de cifrado consta de tres pasos: una adición inicial de la clave de vuelta, n vueltas de cifrado y una vuelta final.

2.5.5 Criptografía de Clave Pública.

2.5.5.1 Sistema RSA

RSA es un algoritmo que se fundamenta en el hecho de que la factorización de números primos es un problema difícil (en el sentido computacional) [CAB96]:

El algoritmo es extremadamente simple:
Primero es necesario calcular los valores de las llaves:

1.- Encontrar dos números primos **grandes** (100 dígitos o más), **p** y **q**
2.- Definir n como n=pq
3.- Encontrar n número primo aleatorio **e** tal que sea primo relativo del entero (p-1)(q-1)
4.- Calcular el entero único **d** en rango 1<= **e** <= (p-1)(q-1)
tal que
ed = 1 (mod (p-1)(q-1)) d existe y es único.
5.- La llave pública es P = (e,n)
6.- La llave secreta es S = (d,n)

Para cifrar un M, utilizando P : C=P(M) = M^e (mod n)
Y para descifrarlo, utilizando S: M = S© = C^d (mod n)

Utilizando RSA es posible distinguir sin ambigüedad al remitente de un menasje. Es muy útil para firmas electrónicas y para certificados (documentos digitales que atestiguan que una llave pública corresponde a un individuo determinado).

Las llaves pública y privada tienen características matemáticas, su generación es siempre en parejas, y están relacionadas de tal forma que si dos llaves públicas son diferentes, entonces, las correspondientes llaves privadas son diferentes y viceversa. En otras palabras, si dos sujetos tienen llaves públicas diferentes, entonces sus llaves privadas son diferentes.

La idea es la de que cada individuo genere un par de llaves: pública y privada. El individuo debe de mantener en secreto su llave privada, mientras que la llave pública la puede dar a conocer a los demás individuos.

El procedimiento de firmado consiste en que mediante un programa de cómputo, un sujeto alimenta un documento a firmar y su llave privada (que solo el conoce). El programa produce como resultado un mensaje digital denominado firma digital. Juntos, el documento y la firma constituyen el documento firmado. [VIL00]

2.5.5.2 Sistema de ElGamal

El problema del logaritmo discreto consiste en calcular x a partir de la expresión : y=a ^ (mod p), siendo p un número primo. Los problemas de la factorización y el cálculo de logaritmos discretos parecen tener idéntica dificultad. [CAB96]:
En 1985 ElGamal presenteó un sistema basado en la aparente intratabilidad del problema del logaritmo discreto. Este sistema fue desarrollado a partir del conocido algoritmo de distribución de claves de Diffie y Hellman [CAB96]:

Supóngase que A y B quieren ponerse de acuerdo en la clave secreta común que van a usar ambos para comunicarse entre sí. Para ello A escoge un entero aleatorio Ka entre 1 y p-1 (secreto) y calcula a^Ka Ap (público). B hace lo mismo con Kb y. La clave secreta común que usarán ambos es a^(KaKb). Es secreta porque sólo ellos pueden calcularla. En este cifrado se distinguen las claves secreta, pública, de cifrado y de descifrado.

Este sitema tiene un funcionamiento distinto al del resto de los sistemas de clave pública, ya que el cifrado se realiza utilizando, demá de la clave pública del receptor, la clave secreta de emisor.

2.6 Gestión de claves (*passwords*)

Abarca la generación, distribución, almacenamiento, tiempo de vida, destrucción y aplicación de las claves de acuerdo con una política de seguridad [GAS89] [GAR91].

Generación de claves.

La seguridad de un algoritmo descansa en la clave. Un criptosistema que haga uso de claves criptográficamente débiles será él mismo débil. Algunos aspectos a considerar que se presentan a la hora de la elección de las claves son:

Espacio de claves reducido.

Cuando existen restricciones en el número de bits de la clave, o bien en la clase de bytes permitidos (caracteres ASCII, caracteres alfanuméricos, imprimibles, etc.), los ataques de fuerza bruta con hardware especializado o proceso en paralelo pueden desbaratar en un tiempo razonable estos sistemas.

Elección pobre de la clave

Cuando los usuarios eligen sus claves, la elección suele ser muy pobre en general (por ejemplo, el propio nombre o el de la mujer), haciéndolas muy débiles para un ataque de fuerza bruta que primero pruebe las claves más obvias (ataque de diccionario).

Claves aleatorias

Claves buenas son las cadenas de bits aleatorios generadas por medio de algún proceso automático (como una fuente aleatoria fiable o un generador pseudo-aleatorio criptográficamente seguro), de forma que si la clave consta de 64 bits, las 264 claves posibles sean igualmente probables. En el caso de los criptosistemas de clave pública, el proceso se complica, ya que a menudo las claves deben verificar ciertas propiedades matemáticas (ser primos dos veces seguros, residuos cuadráticos, etc.).

Frases de paso

Esta solución al problema de la generación de contraseñas seguras (y fáciles de recordar) por parte del usuario consiste en utilizar una frase suficientemente larga que posteriormente es convertida en una clave aleatoria por medio de un algoritmo (key-crunching).

Distribución de claves

Sin duda alguna, el problema central de todo sistema de gestión de claves lo constituyen los procedimientos de distribución de éstas. Esta distribución debe efectuarse previamente a la comunicación. Los requisitos específicos en cuanto a seguridad de esta distribución dependerán de para qué y cómo van a ser utilizadas las claves. Así pues, será necesario garantizar la identidad de su origen, su integridad y, en el caso de claves secretas, su confidencialidad.

Las consideraciones más importantes para un sistema de gestión de claves son el tipo de ataques que lo amenazan y la arquitectura del sistema. Normalmente, es necesario que la distribución de claves se lleve a cabo sobre la misma red de comunicación donde se está transmitiendo la información a proteger. Esta distribución es automática y la transferencia suele iniciarse con la petición de clave por parte de una entidad a un Centro de Distribución de Claves (intercambio centralizado) o a la otra entidad involucrada en la comunicación (intercambio directo). La alternativa es una distribución manual (mediante el empleo de correos seguros, por ejemplo), independiente del canal de comunicación. Esta última alternativa implica un alto coste económico y un tiempo relativamente largo para llevarse a cabo, lo que la hace descartable en la mayoría de las situaciones. La distribución segura de claves sobre canal inseguro requiere protección criptográfica y, por tanto, la presencia de otras claves, conformando una jerarquía de claves. En ciertopunto se requerirá protección no criptográfica de algunas claves (llamadas maestras), usadas para intercambiar con los usuarios de forma segura las claves que usarán en su(s) futura(s) comunicación(es). Entre las técnicas y ejemplos no criptográficos podemos citar seguridad física y confianza.

La distribución de claves se lleva siempre a cabo mediante protocolos, es decir, secuencias de pasos de comunicación (transferencia de mensajes) y pasos de computación. Muchas de las propiedades de estos protocolos dependen de la estructura de los mensajes intercambiados y no de los algoritmos criptográficos subyacentes. Por ello, las debilidades de estos protocolos provienen normalmente de errores cometidos en los niveles más altos del diseño.

Las claves criptográficas temporales usadas durante la comunicación, llamadas claves de sesión, deben ser generadas de forma aleatoria. Para protegerlas será necesaria seguridad física o cifrado mediante claves maestras, mientras que para evitar que sean modificadas deberá utilizarse seguridad física o autenticación. La autenticación hace uso de parámetros como time-stamps y contadores para protegerse también contra la reactuación con antiguas claves.

Almacenamiento de claves.

En sistemas con un solo usuario, la solución más sencilla pasa por ser su retención en la memoria del usuario. Una solución más sofisticada y que desde luego funcionará mejor para claves largas, consiste en almacenarlas en una tarjeta de banda magnética, en una llave de plástico con un chip ROM (ROM key) o en una tarjeta inteligente, de manera que el usuario no tenga más que insertar el dispositivo empleado en alguna ranura a tal efecto para introducir su clave [NAT94].

Otra manera de almacenar claves difíciles de recordar es en forma encriptada mediante una clave fácil de recordar, como por ejemplo almacenar en disco la clave privada RSA cifrada mediante una clave DES.

Tiempo de vida de claves.

Una clave nunca debería usarse por tiempo indefinido. Debe tener una fecha de caducidad, por las siguientes razones [McC98]:

- Cuanto más tiempo se usa una clave, aumenta la probabilidad de que se comprometa (la pérdida de una clave por medios no criptoanalíticos se denomina compromiso).
- Cuanto más tiempo se usa una clave, mayor será el daño si la clave se compromete, ya que toda la información protegida con esa clave queda al descubierto.
- Cuanto más tiempo se usa una clave, mayor será la tentación de alguien para intentar desbaratarla.
- En general es más fácil realizar criptoanálisis con mucho texto cifrado con la misma clave.

Para protocolos orientados a conexión, una elección obvia es usar la misma clave de sesión durante la duración de la comunicación, siendo descartada al finalizar la comunicación y nunca reutilizada. Si la conexión lógica posee una vida muy larga, sería prudente en este caso cambiar la clave de sesión periódicamente, por ejemplo cada vez que el número de secuencia de la PDU completa un ciclo.

Para protocolos no orientados a conexión, no existe un inicio o fin de sesión explícitos. Por lo tanto, no resulta tan obvio con qué frecuencia debería cambiarse la clave. Con el fin de no recargar la información de control ni retrasar la transacción, una estrategia válida sería usar una clave de sesión durante un cierto período o para un cierto número de transacciones.

Las claves maestras no necesitan ser reemplazadas tan frecuentemente, ya que se usan ocasionalmente para el intercambio de claves. En cualquier caso, no hay que olvidar

que si una clave maestra se compromete, la pérdida potencial es enorme, de hecho, todas las comunicaciones cifradas con claves intercambiadas con esa clave maestra.

En el caso del cifrado de grandes archivos de datos, una solución económica y segura, mejor que andar descifrando y volviendo a cifrar los archivos con una nueva clave todos los días, sería cifrar cada archivo con una única clave y después cifrar todas las claves con una clave maestra, que deberá ser almacenada en un lugar de alta seguridad, ya que su pérdida o compromiso echaría a perder la confidencialidad de todos los archivos [DEP85].

Destrucción de claves.

Las claves caducadas deben ser destruidas con la mayor seguridad, de modo que no caigan en manos de un adversario, puesto que con ellas podría leer los mensajes antiguos. En el caso de haber sido escritas en papel, éste deberá ser debidamente destruido; si habían sido grabadas en una EEPROM, deberá sobreescribirse múltiples veces, y si se encontraba en EPROM, PROM o tarjeta de banda magnética, deberán ser hechas añicos (muy pequeñitos, a poder ser). En función del dispositivo empleado, deberá buscarse la forma de que se vuelvan irrecuperables [FIA97].

2.7 Qué es un firewall para Internet.

Los firewalls son una forma sumamente efectiva de seguridad para redes. A continuación se describe lo que los firewalls para Internet pueden hacer por la seguridad general de un sitio. [RAN94] [CUR90]

En la construcción de edificios, un muro contra incendios (*firewall*) está diseñado para evitar que se propague el fuego de una parte a otra. En teoría, un firewall para Internet cumple un propósito similar: evita que los peligros de Internet se extiendan a su red interna. En la práctica, un firewall para Internet se asemeja más bien a la fosa de un castillo medieval que al muro contra incendios de un edificio moderno. Satisface varios propósitos [RAN94]:

- Restringe el acceso a un punto cuidadosamente controlado.
- Evita que los atacantes se acerquen más a sus demás defensas.
- Restringe a las personas para que salgan en un punto cuidadosamente controlado.

Un firewall para Internet se instala con mayor frecuencia en el punto donde la red interna protegida se conecta con Internet, como muestra la siguiente figura: (Figura 2.3)

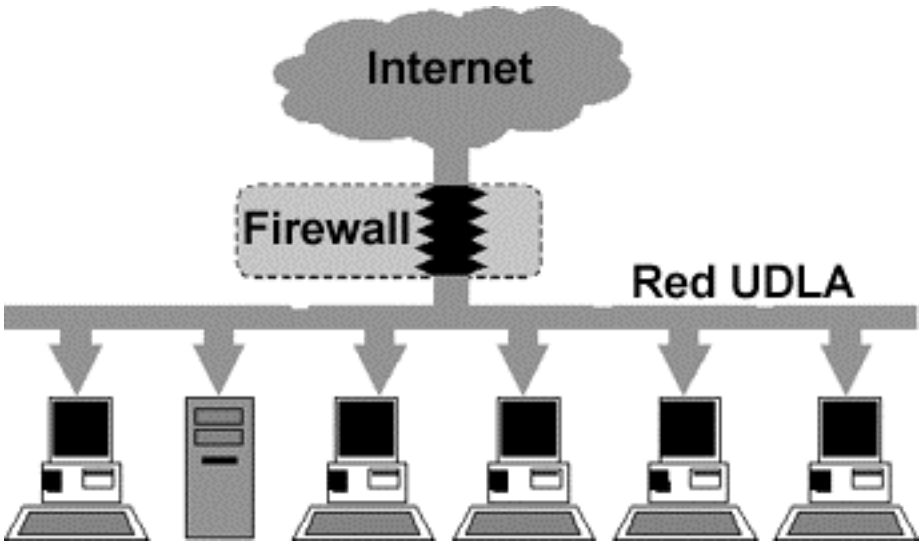

Figura 2.3. Vista general de un Firewall para Internet

Un firewall, por lo general, separa una red interna de Internet

Todo el tráfico que viene de Internet o que sale de la red interna pasa por el firewall. Debido a ello el firewall tiene la oportunidad de asegurarse de que el tráfico sea aceptable.

Esto significa que lo que pasa (correo electrónico, transferencia de archivos, inicio de sesiones remotas o cualquier tipo de interacción específica entre sistemas específicos) cumple con la política de seguridad del sitio [DIF92].

Su implementación física varía de un sitio a otro. Con mayor frecuencia, un firewall es un conjunto de componentes de hardware (un enrutador, una computadora anfitrión, o cierta combinación de enrutadores, computadoras y redes con software apropiado). Existen varias formas de configurar este equipo, dependiendo de la política de seguridad, del presupuesto y de las operaciones generales de un sitio específico.

Un firewall rara vez es un solo objeto físico, aunque algunos de los productos comerciales más nuevos intentan poner todo en la misma caja. Es común que un firewall tenga varias partes y algunas de éstas quizá cumplan otras tareas además de funcionar como parte del firewall. La conexión a Internet es casi siempre parte de un firewall.

Se ha comparado un firewall con la fosa de un castillo medieval, y como tal, no es invulnerable. No protege contra la gente que ya está dentro; funciona mejor en conjunción con defensas internas y, aunque la llene de cocodrilos, algunas personas lograrán nadar al otro lado. Un firewall también tiene sus desventajas; construir uno requiere de gastos y esfuerzos significativos, y las restricciones que le impone a las personas de adentro pueden ser una molestia considerable.

Los firewalls pueden utilizarse como los muros contra incendio de un edificio, para dividir partes de un sitio de las otras cuando éstas tienen necesidades de seguridad específicas.

Los firewall se clasifican en router apantallado, bastión, gateway dual y gateway de host apantallado.

Router Apantallado.

Puede ser un router comercial, o bien, un host configurado a tal efecto que tenga capacidad de filtrar paquetes. Tienen capacidad de bloquear el tráfico procedente de determinados hosts.

Host Bastión.

Es el punto de la red crítico en cuanto a seguridad, debido a que si su seguridad es violada la red a proteger se puede ver comprometida. Generalmente tienen software de seguridad y tienen un control exhaustivo de los audits.

Gateway dual.

Algunos firewalls se construyen sin el primero de los elementos, simplemente conectando un host sin enrutamiento IP entre la red e internet. Se puede comunicar desde la red con el gateway, o desde fuera con el mismo, pero no se permite tráfico directo entre ambas redes.

Gateway de host apantallado.

Es posiblemente la estructura más común para firewalls. Se construye a partir de un router apantallado y un host bastión. Este último se sitúa en la red local, y el primero se configura de tal forma que sea el host bastión el único de la red al que se puede llegar desde internet. Además en el router se bloquea el tráfico con destino a ciertos

puertos del host bastión, permitiendo sólo la utilización de un reducido número de servicios.

Existen más configuraciones de firewalls, pero estas vistas son las más comunes. Es importante cuando se tiene un firewall controlar ciertos aspectos como:

a) Control de daños. Si el firewall cae, hay que tener claros los pasos que se deben seguir.
b) Zonas de riesgo. Ver lo grande que es la zona de riesgo durante la operación normal del firewall. Una medida de esto puede ser el número de hosts y router que pueden ser probados desde fuera.
c) Modo de fallos. Si se traspasa el firewall, hay que comprobar si va a ser fácil de detectar, para ello hay que ver la información que se guarda para poder diagnosticar el ataque.
d) Facilidad de uso. Medir los inconvenientes que plantea el firewall.
e) Postura. Es la idea básica de diseño de un firewall: "Aquello que no está expresamente prohibido, está permitido" o viceversa.

2.7.1 Qué puede hacer un firewall.

Piense en un firewall como un cuello de botella. Todo el tráfico que entra y sale debe pasar por este estrecho punto de inspección. Un firewall da un grado de eficiencia enorme a la seguridad de redes porque le permite concentrar sus medidas de seguridad en este punto de inspección: el punto donde su red se conecta con Internet. [McC98]

Resulta más eficiente centrar la seguridad de esta forma que extender las decisiones de seguridad y tecnologías por todas partes, intentando cubrir todas las bases de manera individual. Aunque la implantación de firewalls puede costar miles de dólares, la mayoría de los sitios encuentran que concentrar el hardware y software de seguridad más efectivo en el firewall es menos caro y más eficaz que otras medidas (y, con toda certeza, menos caro que tener una seguridad ineficaz).

Un firewall puede reforzar las políticas de seguridad

Muchos de los servicios que las personas demandan de Internet son, por su propia naturaleza, inseguros. Un firewall es como el agente de tránsito para estos servicios. Refuerza las políticas de seguridad del sitio, permitiendo que pasen sólo los servicios "aprobados" y aquellos que cumplen con las reglas establecidas para ello.

Por ejemplo, la administración de un sitio puede decidir que ciertos servicios, como el Sistema de Archivos de Red de Sun (NFS) es simplemente muy riesgoso para utilizarlo a través de un firewall. No importa qué sistema intente ejecutarlos o que usuario los quiera: el firewall los mantendrá potencialmente peligrosos dentro de él. (Aún ahí pueden utilizarse para que los usuarios internos se ataquen entre sí, pero eso queda fuera del control del firewall.) Otro sitio tal vez decida que sólo un sistema interno puede comunicarse con el mundo exterior. Otro más quizá decida permitir el acceso de todos los sistemas de cierto tipo, o que pertenecen a un grupo determinado; las variaciones en las políticas de seguridad de un sitio son interminables.

Puede llamarse un firewall para ayudar a poner en vigor políticas más complicadas. Por ejemplo, quizá a sólo ciertos sistemas dentro del firewall se les permite transferir archivos hacia y desde Internet; empleando otros mecanismos para controlar qué usuario tienen acceso a esos sistemas, puede controlar qué usuarios tienen estas capacidades. Dependiendo de las tecnologías que se elijan para implementar el firewall, éste puede tener una habilidad mayor o menor de poner en vigor tales políticas.

Un firewall puede almacenar su relación con Internet de manera eficiente

Debido a que todo transita por el firewall, éste proporciona un buen lugar para reunir información sobre el uso de sistemas y redes (o el mal uso). Como un punto de único acceso, un firewall puede registrar lo que ocurre entre la red protegida y la red externa.

Un firewall limita la exposición

Aunque este punto es más relevante para el uso de firewalls internos vale la pena mencionarlo. A veces se utiliza un firewall para mantener separada una sección de la red de otra. Al hacer esto, evita que los problemas que impactan una sección se extiendan a través de toda la red. En algunos casos, hará esto porque una sección de la red puede ser más confiable que otra; en otros casos, debido a que una sección es más sensible que otra. Cualquiera que sea el motivo, la existencia de un firewall limita el daño que un problema de seguridad en la red puede causar a la red en general.

2.7.2 Qué no puede hacer un firewall.

Los firewalls ofrecen excelente protección contra las amenazas a la red, pero no son una solución de seguridad total. [McC98] Ciertas amenazas están fuera del control del firewall. Debe encontrar otras formas de protegerse contra ellas incorporando

seguridad física, seguridad para anfitrión y educación para el usuario en su plan general de seguridad. A continuación se analizan algunas de las debilidades de los firewalls.

Un firewall no puede proteger contra personas maliciosas internas.

Un firewall puede evitar que un usuario del sistema envíe información del propietario fuera de la organización a través de su conexión de red; también lo evitaría no teniendo una conexión de red. Pero ese mismo usuario podría copiar los datos en disco, cinta o papel y sacarlos del edificio en su portafolio.

Si el atacante ya está dentro del firewall , éste no puede hacer casi nada por resistir un ataque. Los usuarios internos pueden robar datos, dañar el hardware y el software y modificar los programas de manera sutil sin acercarse al firewall. Las amenazas desde adentro requieren de medidas de seguridad internas, como seguridad para anfitrión y educación para el usuario.

Un firewall no puede proteger contra conexiones que no pasan por él

Un firewall puede controlar el tránsito que pasa por él de manera eficaz; sin embargo, no hay nada que pueda hacer con el tránsito que no pasa por él. Por ejemplo, ¿qué sucede si el sitio permite acceso a los sistemas internos que están atrás del firewall por medio de acceso telefónico conmutado? El firewall no tiene absolutamente ninguna forma de evitar que un intruso entre así a través de un módem.

A veces, los usuarios que son expertos técnicos o los administradores del sistema instalan sus propias "puertas traseras" (back door) a la red (digamos una conexión de módem), ya sea en forma temporal o permanente, porque se oponen a las restricciones que los firewall les imponen a ellos y a sus sistemas. Los firewalls no pueden hacer nada con esto. Se trata en realidad de un problema de manejo de personas, no un problema técnico.

Un firewall no puede proteger contra amenazas antes desconocidas

Un firewall está diseñado para protegerlo contra amenazas conocidas. Uno bien diseñado puede proteger también contra nuevas amenazas (por ejemplo, al negar todos menos los servicios confiables, un firewall evita que las personas instalen servicios nuevos e inseguros). Sin embargo, ningún firewall puede defenderse de manera automática contra cada amenaza nueva que surge. Periódicamente, las personas descubren nuevas formas de actuar, utilizando servicios que antes eran confiables, o

usando ataques que simplemente no se le habían ocurrido antes a nadie. No puede instalar un firewall una sola vez y esperar que lo proteja para siempre.

Un firewall no puede proteger contra virus

Los firewalls no pueden mantener los virus de PC y Macintosh fuera de la red. Aunque muchos firewalls revisan todo el tránsito que entra para determinar si puede pasar a la red interna, la exploración ocurre en su mayoría a nivel direcciones fuente y destino y número de puerto, no en los detalles de los datos. Aun cuando se tenga filtrado de paquetes o software proxy complejo, la protección contra virus en un firewall no es muy práctico. Hay demasiados tipos de virus e infinidad de formas en que uno de ellos puede ocultarse dentro de los datos.

Detectar un virus al azar en un paquete de datos que pasa a través de un firewall es muy difícil; requiere de:

2 Reconocer que el paquete es parte de un programa.
3 Determinar cómo debe verse el programa.
4 Determinar que el cambio se debe a un virus.

Incluso el primer punto es un reto. La mayor parte de los firewalls son máquinas protectoras de varios tipos con diferentes formatos de archivos ejecutables. Un programa puede ser un archivo ejecutable compilado o un script , y muchas máquinas soportan múltiples archivos ejecutables compilados. Además, la mayoría de los programas están empaquetados para su transporte, y con frecuencia también están comprimidos. Los paquetes transferidos por medio de correo electrónico o noticias de Usenet también están codificados a ASCII de diferentes maneras.

En el supuesto de que pudiera hacer un trabajo perfecto de bloquear un virus en el firewall, aun así no ha solucionado el problema de los virus. No ha hecho nada sobre las fuentes más comunes de virus: software descargado de sistemas de tableros de foros de discusión (BBS) de acceso telefónico conmutado, software traído en disquetes desde casa u otros sitios e incluso software preinfectado de fábrica son más comunes que el software infectado con virus desde Interntet. Cualquier cosa que haga para solucionar esas amenazas también resolverá el problema de software transferido a través de un firewall.

La forma más práctica de solucionar el problema de los virus es a través de programas antivirus basados en anfitrión, y la educación del usuario en relación con los peligros de los virus y las preocupaciones que se deben tomar contra ellos.

2.7.3 Enrutamiento en Unix

Un firewall consiste en una máquina en la que se ha desactivado el enrutamiento de IP. Mediante este servicio una máquina encamina paquetes de aquellas que se encuentran en su tabla de enrutamiento. En Solaris 2 existe la posibilidad de hacerlo con el comando ndd [CHE94]:

```
# ~ndd /dev/ip ip_forwarding 0
```

Si se añade esta línea al archivo /etc/rc2.d/S69inet se desactiva definitivamente el encaminamiento de paquetes.

2.8 Herramientas existentes.

A continuación se presentan algunas herramientas de seguridad existentes, algunas de dominio público mundial y otras (la mayoría) no es posible que salgan de Estados Unidos.

Portmapper

Portmapper es una herramienta que permite monitorear y filtrar peticiones a diferentes servicios atendidos con portmapper (por ejemplo, NFS o Yellow Pages). Su uso es mucho menos común ya que ha quedado desplazado por los mapeadores de puertos estándar que vienen con muchos sistemas Unix, y que incorporan en la actualidad más medidas de seguridad que este paquete.

lsof (List Open Files)

lsof es un programa que examina descriptores de archivo abiertos por procesos; de esta forma, se puede localizar sockets a la escucha, procesos que escriben en un archivo, archivos abiertos por un proceso (importante a la hora de detectar sniffers), etc.

TCFS (Transparent Cryptographic File System)

TCFS es un sistema de archivos que incorpora cifrado de datos. Siguiendo un mecanismo parecido a NFS, ofrece la posibilidad de mantener archivos encriptados en una unidad, y también incrementa la seguridad en la comunicación por red entre cliente y servidor de discos: todos los datos viajan encriptados, lo que hace a TCFS bastante recomendable para sistemas distribuidos. TCFS opera a nivel de kernel, por lo que a priori es más rápido y seguro que CFS, de Matt Blaze.

CFS (Cryptographic File System)

CFS es un sistema de archivos cifrado que opera a nivel de usuario. Actúa como interfaz de cualquier sistema de archivos estándar de Unix, incluyendo NFS (los datos no viajan o se mantienen en el disco en texto claro en ningún momento).

Tripwire

Tripwire es un verificador de integridad para archivos y directorios, muy útil para evitar la inserción de troyanos entre los ejecutables de nuestro sistema. Utiliza para ello un algoritmo de firma digital (generalmente con función hash MD5 y Snefru)

Strobe

Este programa es un supervisor de puertos que funciona sobre multitud de sistemas Unix. Puede resultar útil para comprobar que servicios de red se tienen aceptando conexiones en el sistema, y así poder reducir su número al mínimo necesario.

COPS (Computer Oracle and Password System)

COPS es una colección de herramientas de seguridad para Unix (quizás algo obsoletas, pero útiles igualmente) que sirve para automatizar tareas que normalmente el administrador realiza de forma manual (verificación de passwords aceptables, restricciones de sistemas de archivos NFS, "+" en /etc/host.equiv...). Las posibles debilidades del sistema son guardadas en un archivo resultado o enviadas por correo, pero NUNCA corregidas.

TCPdump

TCPdump es una herramienta para analizar el tráfico de la red. Imprime las cabeceras de paquetes que circulan por un interfaz de red y cumplen unas determinadas características (un protocolo determinado, paquetes dirigidos a una determinada dirección, a un cierto puerto...).

TIGER

Esta es una herramienta formada por diversos programas que nos permite auditar la seguridad de nuestro sistema (principalmente aquellos fallos que pueden comprometer la seguridad del administrador, como problemas en daemons). También utiliza sistemas de firma digital para detectar la modificación no autorizada de binarios.

Sniffit & TOD

Sniffit es una herramienta de monitoreo y sniffer que trabaja sobre diferentes plataformas Unix. Proporciona a un administrador información detallada sobre todo el tráfico que circula por su sistema, y también le ofrece la posibilidad de neutralizarlo si se utiliza TOD (Touch of Dead), es decir, cerrar las conexiones que pasan a través de su máquina

ARPWatch

ARPWatch es una herramienta utilizada para comprobar la correspondencia entre pares IP-Dirección HW. En caso de que un cambio en un par se produzca (esto es, se escuche en la interfaz de red del sistema), ARPWatch envía un mail al administrador, notificando el suceso. También sirve para notificar la existencia de estaciones nuevas o la retransmisión de estaciones que llevaban mucho tiempo apagadas. De esta forma, es útil para notificar posibles ataques, como IP Spoofing.

Argus

Argus es un analizador de red especialmente diseñado para redes con un elevado tráfico. Su potente histórico (información guardada) permite detectar problemas en la red, detectar servicios nuevos, tráfico bloqueado en un router, escaneos por parte de un posible intruso ... de una forma rápida y efectiva.

IP Filter

IP Filter es una herramienta para filtrar paquetes en estaciones Unix que están actuando como routers (una especie de firewall). Permite filtrar tanto paquetes de entrada como de salida de la red, en función del protocolo utilizado, de la ip origen o destino, etc.

Npasswd

Npasswd es una herramienta que sustituye al comando passwd de cualquier sistema Unix habitual. Es una recomendable herramienta para cualqiuer sistema, ya que obliga a los usuarios a escoger passwords aceptables, rechazando palabras de diccionario, *joes* (mismo login y password), claves demasiado cortas, etc...Funciona con NIS o seguridad C2 (Shadow Password).

S/KEY

S/Key es un mecanismo que implementa el One Time Password (es el caso más extremo del *Aging Password* o envejecimiento de contraseñas, en el que una clave sólamente sirve para una sesión), evitando así los potenciales peligros derivados de la captura de un password por parte de un cracker.

DTK

Deception ToolKit es una herramienta que escucha peticiones al sistema de Unix y responde con información falsa, haciendo que el potencial intruso piense que nuestro sistema está lleno de bugs. De esta forma, y gracias al monitoreo y sistema de logs de DTK, se puede obtener mucha información sobre los atacantes, así como ganar tiempo si interesa seguir el ataque.

Titan

Titan está formado por una serie de shellscripts que solucionan pequeños problemas de seguridad genéricos, como la existencia por defecto de accesos lp con un shell válido; los autores insisten en el hecho de que no reemplaza a ningún software de seguridad, sino que se limita a solucionar estos pequeños problemas y facilitar la instalación y configuración a los administradores.

chrootuid

Este daemon, diseñado por Wietse Venema, ejecuta un servicio de red (http, gopher...) con el mínimo privilegio posible y con acceso restringido al sistema de archivos (mediante chroot()).

SWatch

Este programa es útil para monitorear los archivos de log de un sistema y emprender acciones (como enviar un correo al administrador) cuando se detectan situaciones sospechosas.