Capítulo 2.
Seguridad en UNIX y Herramientas de Seguridad.

2.1. La Seguridad en UNIX.

En la década de los ochenta para mucha gente el concepto de *seguridad* era algo inimaginable en el entorno Unix: la facilidad con que un experto podía acceder a un sistema, burlar todos sus mecanismos de protección y conseguir el máximo nivel de privilegio era algo de sobra conocido por todos, por lo que nadie podía pensar en un sistema Unix *seguro*.

Afortunadamente, los tiempos han cambiado mucho desde entonces. Aunque en un principio y según uno de sus creadores, Unix no se diseñó para ser seguro **[Ritchie, 1986]**, a finales de los 80 se convirtió en el primer sistema operativo en alcanzar niveles de seguridad casi militares **[Hecht, 1988] [Serlin, 1991]**. En la actualidad se puede considerar el sistema operativo de propósito general más fiable del mercado; cualquier entorno Unix puede ofrecer los mecanismos de seguridad suficientes para satisfacer las necesidades de la mayoría de instituciones. El problema es que en muchas ocasiones se pone a trabajar a Unix tal y como se instala por defecto, lo que convierte a cualquier sistema operativo, Unix o no, en un auténtico agujero en cuanto a seguridad se refiere: cuentas sin *passwords* o con *passwords* por defecto, servicios abiertos, sistemas de ficheros susceptibles de ser compartidos.

A la vista de lo comentado en este punto, parece claro que Unix ha dejado de ser ese sistema arcaico e inseguro de sus primeros tiempos para convertirse en el entorno de trabajo más fiable dentro de la gama de sistemas operativos de propósito general; sin embargo, por alguna extraña razón, mucha gente tiende a considerar todavía a los equipos Unix como amenazas en la red, especialmente a los clones gratuitos como Linux o FreeBSD que habitualmente se ejecutan en PCs; el hecho de que sean gratuitos no implica en ningún momento que sean inestables, y mucho menos, inseguros: empresas tan importantes como *Yahoo!* (`www.yahoo.com`) o *Citroën* (`www.citroen.com`), o el propio servicio postal de Estados Unidos utilizan estos entornos como servidores *Web* o como *cortafuegos* en sus redes.

2.2. Seguridad: Sistema de Archivos.

Dentro del sistema Unix todo son archivos: desde la memoria física del equipo hasta el ratón, pasando por módems, teclado, impresoras o terminales. Esta filosofía de diseño es uno de los factores que más éxito y potencia proporciona a Unix **[Kernighan, 1984]**, pero también uno de los que más peligros entraña: un simple error en un permiso puede permitir a un usuario modificar todo un disco duro o leer los datos tecleados desde una terminal. Por esto, una correcta utilización de los permisos, atributos y otros controles sobre los ficheros es vital.

En un sistema Unix típico existen tres tipos básicos de archivos: archivos planos, directorios, y archivos especiales (dispositivos); generalmente, al hablar de *archivos* nos solemos referir a todos ellos si no se especifica lo contrario. Los **archivos planos** son secuencias de *bytes* que *a priori* no poseen ni estructura interna ni contenido significante para el sistema: su significado depende de las aplicaciones que interpretan su contenido. Los **directorios** son archivos cuyo contenido son otros archivos de cualquier tipo (planos, más directorios, o archivos especiales), y los **archivos especiales** son archivos que representan dispositivos del sistema; este último tipo se divide en dos grupos: los dispositivos orientados a carácter y los orientados a bloque. La principal diferencia entre ambos es la forma de realizar operaciones de entrada/salida: mientras que los dispositivos orientados a carácter las realizan *byte* a *byte* (esto es, carácter a carácter), los orientados a bloque las realizan en bloques de caracteres.

El **sistema de archivos** es la parte del núcleo más visible por los usuarios; se encarga de abstraer propiedades físicas de diferentes dispositivos para proporcionar una interfaz única de almacenamiento: el archivo. Cada sistema Unix tiene su sistema de archivos nativo (por ejemplo, *ext2* en Linux, UFS en Solaris o EFS en IRIX), por lo que para acceder a todos ellos de la misma forma el núcleo de Unix incorpora una capa superior denominada VFS (*Virtual File System*) encargada de proporcionar un acceso uniforme a diferentes tipos de sistema de archivos.

2.2.1. Permisos de un archivo.

Los permisos de cada archivo son la protección más básica de estos objetos del sistema operativo; definen quién puede acceder a cada uno de ellos, y de qué forma puede hacerlo. Cuando hacemos un listado largo de ciertos archivos podemos ver sus permisos junto al tipo de fichero correspondiente, en la primera columna de cada línea:

```
anita: ~# ls -l /sbin/rc0
-rwxr--r--   3 root     sys         2689 Dec  1  1998 /sbin/rc0
anita:~#
```

En este caso vemos que el archivo listado es un archivos plano (el primer carácter es un `-') y sus permisos son `rwxr-r-'. ¿Cómo interpretar estos caracteres? Los permisos se dividen en tres ternas en función de a qué usuarios afectan; cada una de ellas indica la existencia o la ausencia de permiso para leer, escribir o ejecutar el archivo: una `r` indica un permiso de lectura, una `w` de escritura, una `x` de ejecución y un `-' indica que el permiso correspondiente no está activado. Así, si en una de las ternas tenemos los caracteres `rwx`, el usuario o usuarios afectados por esa terna tiene o tienen permisos para realizar cualquier operación sobre el archivo. ¿De qué usuarios se trata en cada caso? La primera terna afecta al propietario del archivo, la segunda al grupo del propietario cuando lo creó (recordemos un mismo usuario puede pertenecer a varios grupos) y la tercera al resto de usuarios.
Una forma sencilla para controlar esto es utilizar el comando umask, por ejemplo:

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

Cuando un usuario intenta acceder en algún modo a un archivo, el sistema comprueba qué terna de permisos es la aplicable y se basa únicamente en ella para conceder o denegar el acceso; así, si un usuario es el propietario del archivo sólo se comprueban permisos de la primera terna; si no, se pasa a la segunda y se aplica en caso de que los grupos coincidan, y de no ser así se aplican los permisos de la última terna.

De esta forma es posible tener situaciones tan curiosas como la de un usuario que no tenga ningún permiso sobre uno de sus archivos, y en cambio que el resto de usuarios del sistema pueda leerlo, ejecutarlo o incluso borrarlo; obviamente, esto no es lo habitual, y de suceder el propietario siempre podrá restaurar los permisos a un valor adecuado.

Después de ver el procedimiento de modificación de los permisos de un archivo, este puede parecer demasiado complicado y arcaico para un sistema operativo moderno; a fin de cuentas, mucha gente prefiere gestores gráficos de permisos, igual que prefiere gestores gráficos para otras tareas de administración , programas que dan todo hecho y no obligan al administrador a "complicarse" llenos de menús desplegables y diálogos que una y otra vez preguntan si realmente deseamos modificar cierto permiso (*>Está usted seguro? >Realmente seguro? >Es mayor de edad? >Me lo jura?*). Incluso esas personas aseguran que el procedimiento gráfico es mucho más claro y más potente que el que Unix ofrece en modo texto. Nada más lejos de la realidad, hemos de pensar que este modelo de protección está vigente desde hace casi treinta años y no ha cambiado **absolutamente nada**. Si en todo este tiempo no se ha modificado el mecanismo, obviamente es porque siempre ha funcionado y lo sigue haciendo bien.

2.2.2. Lista de control de acceso: ACLs.

Las Listas de Control de Accesos (ACL por Access Control List) son un mecanismo proporcionado por el UNIX de Hewlett-Packard (HP-UX) para poner en práctica un sistema de control de accesos discrecional. Su utilización permite aumentar enormemente la flexibilidad con que se manejan los permisos de acceso de los archivos y directorios en UNIX.

El mecanismo clásico de permisos de acceso proporcionado por el UNIX tan solo define tres clases de usuarios a los que se puede asociar dichos permisos: el propietario (u), el grupo (g) y otros (o). Sin embargo utilizando listas de control de accesos es posible manipular los permisos de acceso para cada uno de los usuarios y los grupos del sistema, lo que da lugar a un nivel de selectividad mucho más elevado.

Supongamos por ejemplo que queremos permitir que el usuario *pablo* de nuestro grupo pueda escribir en un archivo dado, pero no queremos conceder este permiso a

ninguno de los otros usuarios de nuestro grupo. Este tipo de control de permisos no es posible utilizando el mecanismo tradicional de bits de permiso (rwxrwxrwx). Supongamos por otro lado que pertenecemos al grupo *inf* y queremos conceder el permiso de lectura de un archivo dado a todos los usuarios del grupo *temp*. Utilizando el método tradicional la única forma de hacerlo es conceder este mismo permiso a todos los usuarios de grupos distintos de *inf*. Si embargo , utilizando las listas de control de accesos podemos lograr los dos objetivos anteriores, dado que con las ACLs podemos manejar los permisos a nivel de usuarios individuales, a nivel de grupos individuales, e incluso a nivel de usuarios concretos en grupos concretos.

Cada archivo y directorio del sistema tiene asociada una lista de control de accesos (ACL), siendo cada elemento o entrada de la lista de la forma (usuario.grupo, modo). Cada una de las entradas de la lista de un archivo especifica los permisos de acceso a ese archivo para un usuario en un grupo.

2.3. Auditoría del Sistema.

Casi todas las actividades realizadas en un sistema Unix son susceptibles de ser, en mayor o menor medida, monitorizadas: desde las horas de acceso de cada usuario al sistema hasta las páginas *Web* más frecuentemente visitadas, pasando por los intentos fallidos de conexión, los programas ejecutados o incluso el tiempo de CPU que cada usuario consume. Obviamente esta facilidad de Unix para recoger información tiene unas ventajas inmediatas para la seguridad: es posible detectar un intento de ataque nada más producirse el mismo, así como también detectar usos indebidos de los recursos o actividades “sospechosas”; sin embargo, existen también desventajas, ya que la gran cantidad de información que potencialmente se registra puede ser aprovechada para crear negaciones de servicio o, más habitualmente, esa cantidad de información puede hacer difícil detectar problemas por el volumen de datos a analizar **[Schneier, 1998]**, por esta razón hablamos muy poco este tema.

2.4. Sistemas de detección de intrusos.

A pesar de que un enfoque clásico de la seguridad de un sistema informático siempre define como principal defensa del mismo sus controles de acceso (desde una política implantada en un cortafuegos hasta unas listas de control de acceso en un *router* o en el propio sistema de ficheros de una máquina), esta visión es extremadamente simplista si no tenemos en cuenta que en muchos casos esos controles no pueden protegernos ante un ataque **[Lunt, 1990].**

Por poner un ejemplo sencillo, pensemos en un *cortafuegos* donde hemos implantado una política que deje acceder al puerto 80 de nuestros servidores *Web* desde cualquier máquina de Internet; ese cortafuegos sólo comprobará si el puerto destino de una trama es el que hemos decidido para el servicio HTTP, pero seguramente no tendrá en cuenta si ese tráfico representa o no un ataque o una violación de nuestra política de seguridad: por ejemplo, no detendrá a un pirata que trate de acceder al archivo de contraseñas de una máquina aprovechando un *bug* del servidor *Web*. Desde un pirata informático externo a nuestra organización a un usuario autorizado que intenta obtener privilegios que no le corresponden en un sistema, nuestro entorno de trabajo no va a estar nunca a salvo de intrusiones.

Llamaremos **intrusión** a un conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso **[Heady, 1990]**; analizando esta definición, podemos darnos cuenta de que una intrusión no tiene por qué consistir en un acceso no autorizado a una máquina: también puede ser una negación de servicio. A los sistemas utilizados para detectar las intrusiones o los intentos de intrusión se les denomina **sistemas de detección de intrusiones** (*Intrusion Detection Systems*, IDS) o, más habitualmente y aunque no sea la traducción literal **sistemas de detección de intrusos**; cualquier mecanismo de seguridad con este propósito puede ser considerado un IDS, pero generalmente sólo se aplica esta denominación a los sistemas automáticos (*software* o *hardware*): es decir, aunque un guardia de seguridad que vigila en la puerta de la sala de operaciones pueda considerarse en principio como un sistema de detección de intrusos, como veremos a continuación lo habitual (y lógico) es que a la hora de hablar de IDSs no se contemplen estos casos.

a) Clasificación de los IDSs.

Generalmente existen dos grandes enfoques a la hora de clasificar a los sistemas de detección de intrusos: o bien en función de **qué** sistemas vigilan, o bien en función de **cómo** lo hacen. Si elegimos la primera de estas aproximaciones tenemos dos grupos de sistemas de detección de intrusos: los que analizan actividades de una única máquina en busca de posibles ataques, y los que lo hacen de una subred (generalmente, de un mismo dominio de colisión) aunque se emplacen en uno sólo de los *hosts* de la misma. Esta última puntualización es importante: un IDS que detecta actividades sospechosas en una red no tiene porqué (y de hecho en la mayor parte de casos no suele ser así) ubicarse en todas las máquinas de esa red.

- **IDSs basados en red**. Un IDS basado en red monitoriza los paquetes que circulan por nuestra red en busca de elementos que denoten un ataque contra alguno de los sistemas ubicados en ella; el IDS puede situarse en cualquiera de los *hosts* o en un elemento que analice todo el tráfico (como un HUB o un enrutador). Esté donde esté, monitorizará diversas máquinas y no una sola: esta es la principal diferencia con los sistemas de detección de intrusos basados en *host*.

- **IDSs basados en máquina.** Mientras que los sistemas de detección de intrusos basados en red operan bajo todo un dominio de colisión, los basados en máquina realizan su función protegiendo un único sistema; de una forma similar guardando las distancias, por supuesto a como actúa un escudo antivirus residente en MS-DOS, el IDS es un proceso que trabaja en *background* (o que despierta periódicamente) buscando patrones que puedan denotar un intento de intrusión y alertando o tomando las medidas oportunas en caso de que uno de estos intentos sea detectado.

La segunda gran clasificación de los IDSs se realiza en función de cómo actúan estos sistemas; actualmente existen dos grandes técnicas de detección de intrusos **[Serlin, 1991]**: las basadas en la detección de anomalías (*anomaly detection*) y las basadas en la detección de usos indebidos del sistema (*misuse detection*). Aunque más tarde hablaremos con mayor profundidad de cada uno de estos modelos, la idea básica de los mismos es la siguiente:

- **Detección de anomalías**. La base del funcionamiento de estos sistemas es suponer que una intrusión se puede ver como una anomalía de nuestro sistema, por lo que si fuéramos capaces de establecer un perfil del comportamiento habitual de los sistemas seríamos capaces de detectar las intrusiones por pura estadística: probablemente una intrusión sería una desviación excesiva de la media de nuestro perfil de comportamiento.
- **Detección de usos indebidos**. El funcionamiento de los IDSs basados en la detección de usos indebidos presupone que podemos establecer patrones para los diferentes ataques conocidos y algunas de sus variaciones; mientras que la detección de anomalías conoce lo normal (en ocasiones se dice que tienen un"conocimiento positivo", *positive knowledge*) y detecta lo que no lo es, este esquema se limita a conocer lo anormal para poderlo detectar (conocimiento negativo, *negative knowledge*).

2.5. Firewall.

Un *firewall* es un sistema o grupo de sistemas que hace cumplir una política de control de acceso entre dos redes. De una forma más clara, podemos definir que un firewall como cualquier sistema (desde un simple *router* hasta varias redes en serie) utilizado para separar, en cuanto a seguridad se refiere, una máquina o subred del resto, protegiéndola así de servicios y protocolos que desde el exterior puedan suponer una amenaza a la seguridad. El espacio protegido, denominado **perímetro de seguridad**, suele ser propiedad de la misma organización, y la protección se realiza contra una red externa, no confiable, llamada **zona de riesgo [Ranum, 1995].** Evidentemente la forma de aislamiento más efectiva para cualquier política de seguridad consiste en el aislamiento físico, es decir, no tener conectada la máquina o la subred a otros equipos o a Internet (figura 1.4 (A)). Sin embargo, en la mayoría de organizaciones, los usuarios necesitan compartir información con otras personas situadas en muchas ocasiones a miles de kilómetros de distancia, con lo que no es posible un aislamiento total. El punto opuesto consistiría en una conectividad completa con la red (figura 1.5 (B)), lo que desde el punto de vista de la seguridad es muy problemático: cualquiera, desde cualquier parte del mundo, puede potencialmente tener acceso a nuestros recursos.

Un término medio entre ambas aproximaciones consiste en implementar cierta separación lógica mediante un cortafuegos (figura 2.1 (C)).

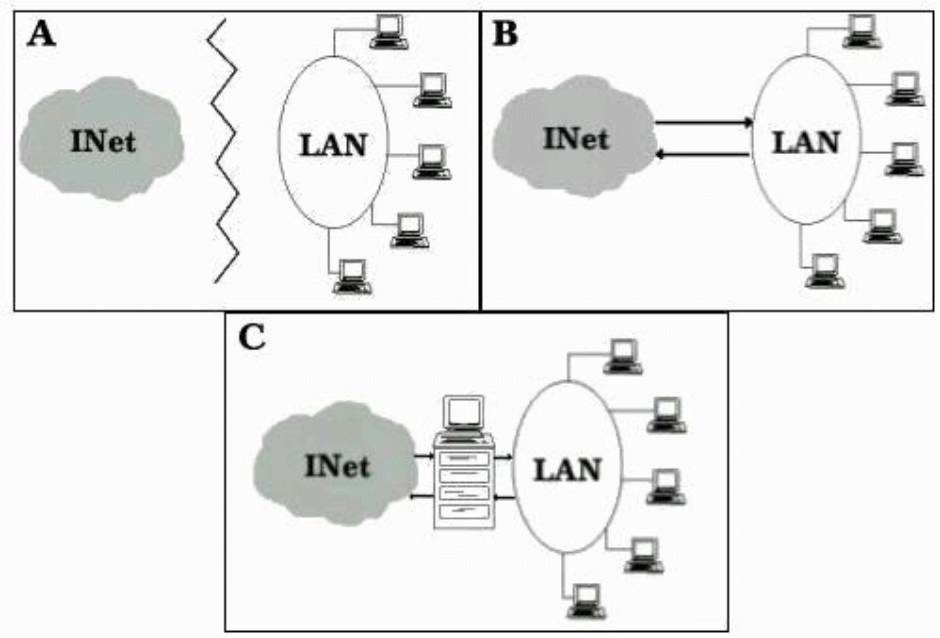

Figura 2.1: (A) Aislamiento, (B) Conexión total, (C) Cortafuegos entre la zona de riesgo y el perímetro de seguridad **[Lizárraga, 2001].**

Antes de hablar de los firewall es casi obligatorio dar una serie de definiciones de partes o características de funcionamiento de un ***firewall***; por máquina o *host* **bastión** (también se denominan **gates**) se conoce a un sistema especialmente asegurado, pero en principio vulnerable a todo tipo de ataques por estar abierto a Internet, que tiene como función ser el punto de contacto de los usuarios de la red interna de una organización con otro tipo de redes. El *host* bastión filtra tráfico de entrada y salida, y también esconde la configuración de la red hacia fuera. Por **filtrado de paquetes** entendemos la acción de denegar o permitir el flujo de tramas entre dos redes (por ejemplo la interna, protegida con el *cortafuegos*, y el resto de Internet) de acuerdo a unas normas predefinidas; aunque el filtro más elemental puede ser un simple *router*, trabajando en el nivel de red del protocolo OSI, esta actividad puede realizarse además en un puente o en una máquina individual.

El filtrado también se conoce como *screening*, y a los dispositivos que lo implementan se les denomina **chokes**; el *choke* puede ser la máquina bastión o un elemento diferente. Un **proxy** es un programa (trabajando en el nivel de aplicación de OSI) que permite o niega el acceso a una aplicación determinada entre dos redes.

Los *firewall* son cada vez más necesarios en nuestras redes, pero todos los expertos recomiendan que no se usen **en lugar de** otras herramientas, sino **junto a** ellas; cualquier firewall, desde el más simple al más avanzado, presenta dos gravísimos problemas de seguridad: por un lado, centralizan todas las medidas en un único sistema, de forma que si éste se ve comprometido y el resto de nuestra red no está lo suficientemente protegido el atacante consigue amenazar a toda la subred simplemente poniendo en jaque a una máquina. El segundo problema, relacionado con éste, es la falsa sensación de seguridad que un firewall proporciona: generalmente un administrador que no disponga de un firewall va a preocuparse de la integridad de todas y cada una de sus máquinas, pero en el momento en que instala el firewall y lo configura asume que toda su red es segura, por lo que se suele descuidar enormemente la seguridad de los equipos de la red interna. Esto, como acabamos de comentar, es un grave error, ya que en el momento que un pirata acceda a nuestro firewall, recordemos que es un sistema muy expuesto a ataques externos, automáticamente va a tener la posibilidad de controlar toda nuestra red. Además, esto ya no es un problema de los *firewall* sino algo de sentido común, un firewall evidentemente no protege contra ataques que no pasan por él: esto incluye todo tipo de ataques internos dentro del perímetro de seguridad, pero también otros factores que *a priori* no deberían suponer un problema.

2.5.1. Componentes de un Firewall.

a). Proxy de aplicación. Es habitual que los firewall utilicen aplicaciones *software* para reenviar o bloquear conexiones a servicios como *finger*, *telnet* o FTP; a tales aplicaciones se les denomina servicios **proxy**, mientras que a la máquina donde se ejecutan se le llama **pasarela de aplicación**. Los servicios *proxy* poseen una serie de ventajas de cara a incrementar nuestra seguridad **[Wack, 1994]**; en primer lugar, permiten únicamente la utilización de servicios para los que existe un *proxy*, por lo que si en nuestra organización la pasarela de aplicación contiene únicamente *proxies* para *telnet*, HTTP y FTP, el resto de servicios no estarán disponibles para nadie.

Una segunda ventaja es que en la pasarela es posible filtrar protocolos basándose en algo más que la cabecera de las tramas, lo que hace posible por ejemplo tener habilitado un servicio como FTP pero con órdenes restringidas (podríamos bloquear todos los comandos put para que nadie pueda subir ficheros a un servidor

b) Monitorización de la actividad. Monitorizar la actividad de nuestro firewall es algo indispensable para la seguridad de todo el perímetro protegido; la monitorización nos facilitará información sobre los intentos de ataque que estemos sufriendo (origen, franjas horarias, tipos de acceso...), así como la existencia de tramas que aunque no supongan un ataque *a priori* sí que son al menos sospechosas **[Bellovin, 1993].**

c) Filtrado de paquetes. Un *firewall* sencillo puede consistir en un dispositivo capaz de filtrar paquetes, un *choke*: se trata del modelo de firewall más antiguo **[Schimmel, 1997]**, basado simplemente en aprovechar la capacidad de algunos *routers*, denominados *screening routers*, para hacer un enrutado selectivo, es decir, para bloquear o permitir el tránsito de paquetes. En un firewall de filtrado de paquetes los accesos desde la red interna al exterior que no están bloqueados son directos (no hay necesidad de utilizar *proxies*, como sucede en los firewall basados en una máquina con dos tarjetas de red), por lo que esta arquitectura es la más simple de implementar (en muchos casos sobre *hardware* ya ubicado en la red) y la más utilizada en organizaciones que no precisan grandes niveles de seguridad, como las que vemos aquí. No obstante, elegir un firewall tan sencillo puede no ser recomendable en ciertas situaciones, o para organizaciones que requieren una mayor seguridad para su subred, ya que los simples *chokes* presentan más desventajas que beneficios para la red protegida. El principal problema es que no disponen de un sistema de monitorización sofisticado, por lo que muchas veces el administrador no puede determinar si el *router* está siendo atacado o si su seguridad ha sido comprometida. Además las reglas de filtrado pueden llegar a ser complejas de establecer, y por tanto es difícil comprobar su corrección: habitualmente sólo se comprueba a través de pruebas directas, con los problemas de seguridad que esto puede implicar.

2.6. Criptografía.

En el *mundo real*, si una universidad quiere proteger los expedientes de sus alumnos los guardará en un armario ignífugo, bajo llave y vigilado por guardias, para que sólo las personas autorizadas puedan acceder a ellos para leerlos o modificarlos; si queremos proteger nuestra correspondencia de curiosos, simplemente usamos un sobre; si no queremos que nos roben dinero, lo guardamos en una caja fuerte. Lamentablemente, en una red no disponemos de todas estas medidas que nos parecen habituales: la principal (podríamos decir **única**) forma de protección va a venir de la mano de la criptografía. El cifrado de los datos nos va a permitir desde proteger nuestro correo personal para que ningún curioso lo pueda leer, hasta controlar el acceso a nuestros archivos de forma que sólo personas autorizadas puedan examinar (o lo que quizás es más importante, modificar) su contenido, pasando por proteger nuestras claves cuando conectamos a un sistema remoto o nuestros datos bancarios cuando realizamos una compra a través de Internet. Hemos presentado con anterioridad algunas aplicaciones que utilizan de una u otra forma la criptografía para proteger nuestra información **[Schneier, 1998]**. La **criptología** (del griego *krypto* y *logos*, estudio de lo oculto, lo escondido) es la ciencia que trata los problemas teóricos relacionados con la seguridad en el intercambio de mensajes en clave entre un emisor y un receptor a través de un canal de comunicaciones (en términos informáticos, ese canal suele ser una red de computadoras). Esta ciencia está dividida en dos grandes ramas: la **criptografía**, ocupada del cifrado de mensajes en clave y del diseño de criptosistemas (hablaremos de éstos más adelante), y el **criptoanálisis**, que trata de descifrar los mensajes en clave, rompiendo así el criptosistema. En lo sucesivo nos centraremos más en la criptografía y los criptosistemas que en el criptoanálisis, ya que nos interesa, más que romper sistemas de cifrado (lo cual es bastante complicado cuando trabajamos con criptosistemas serios), el saber cómo funcionan éstos y conocer el diseño elemental de algunos sistemas seguros. La criptografía es una de las ciencias consideradas como más antiguas, ya que sus orígenes se remontan al nacimiento de nuestra civilización.

Su uso original era el proteger la confidencialidad de informaciones militares y políticas, pero en la actualidad es una ciencia interesante no sólo en esos círculos cerrados, sino para cualquiera que esté interesado en la confidencialidad de unos determinados datos: actualmente existe multitud de software y hardware destinado a

analizar y monitorizar el tráfico de datos en redes de computadoras; si bien estas herramientas constituyen un avance en técnicas de seguridad y protección, su uso indebido es al mismo tiempo un grave problema y una enorme fuente de ataques a la intimidad de los usuarios y a la integridad de los propios sistemas. Aunque el objetivo original de la criptografía era mantener en secreto un mensaje, en la actualidad no se persigue únicamente la privacidad o **confidencialidad** de los datos, sino que se busca además garantizar la **autenticidad** de los mismos (el emisor del mensaje es quien dice ser, y no otro), su **integridad** (el mensaje que leemos es el mismo que nos enviaron) y su **no repudio** (el emisor no puede negar el haber enviado el mensaje).

.2.6.1. Criptosistemas.

a) Criptosistemas de clave secreta. Denominamos criptosistema de clave secreta (de clave privada, de clave única o simétrico) a aquel criptosistema en el que la clave de cifrado, puede ser calculada a partir de la de descifrado y viceversa. En la mayoría de estos sistemas, ambas claves coinciden, y por supuesto han de mantenerse como un secreto entre emisor y receptor: si un atacante descubre la clave utilizada en la comunicación, ha roto el criptosistema. Hasta la década de los setenta, la invulnerabilidad de todos los sistemas dependía de este mantenimiento en secreto de la clave de cifrado. Este hecho presentaba una gran desventaja: había que enviar, aparte del criptograma, la clave de cifrado del emisor al receptor, para que éste fuera capaz de descifrar el mensaje. Por tanto, se incurría en los mismos peligros al enviar la clave, por un sistema que había de ser supuestamente seguro, que al enviar el texto plano.

b) Criptosistemas de clave pública. En 1976, Whitfield Diffie y Martin Hellman, de la Universidad de Stanford, demostraron la posibilidad de construir criptosistemas que no precisaran de la transferencia de una clave secreta en su trabajo **[Diffie, 1976]**. Esto motivó multitud de investigaciones y discusiones sobre la criptografía de clave pública y su impacto, hasta el punto que la NSA (*National Security Agency*) estadounidense trató de controlar el desarrollo de la criptografía, ya que la consideraban una amenaza peligrosa para la seguridad nacional. Veamos ahora en que se basan los criptosistemas de clave pública. En éstos, la clave de cifrado se hace de conocimiento general (se le llama **clave pública**). Sin embargo, no ocurre lo mismo con la clave de descifrado (**clave privada**), que se ha de mantener en secreto. Ambas claves no son

independientes, pero del conocimiento de la pública no es posible deducir la privada sin ningún otro dato (recordemos que en los sistemas de clave privada sucedía lo contrario). Tenemos pues un par clave pública-clave privada; la existencia de ambas claves diferentes, para cifrar o descifrar, hace que también se conozca a estos criptosistemas como **asimétricos**. Cuando un receptor desea recibir una información cifrada, ha de hacer llegar a todos los potenciales emisores su clave pública, para que estos cifren los mensajes con dicha clave. De este modo, el único que podrá descifrar el mensaje será el legítimo receptor, mediante su clave privada. Matemáticamente, si es el algoritmo cifrador y el descifrador, se ha de cumplir que representando un mensaje, y siendo y las claves de descifrado y cifrado, respectivamente.

2.6.2. Criptoanálisis.

El criptoanálisis es la ciencia opuesta a la criptografía (quizás no es muy afortunado hablar de ciencias *opuestas*, sino más bien de ciencias *complementarias*), ya que si ésta trata principalmente de crear y analizar criptosistemas seguros, la primera intenta romper esos sistemas, demostrando su vulnerabilidad: dicho de otra forma, trata de descifrar los criptogramas.

El término *descifrar* siempre va acompañado de discusiones de carácter técnico, aunque asumiremos que descifrar es conseguir el texto en claro a partir de un criptograma, sin entrar en polémicas de reversibilidad y solidez de criptosistemas. En el análisis para establecer las posibles debilidades de un sistema de cifrado, se han de asumir las denominadas condiciones del peor caso: (1) el criptoanálista tiene acceso completo al algoritmo de encriptación, (2) el criptoanálista tiene una cantidad considerable de texto cifrado, y (3) el criptoanálista conoce el texto en claro de parte de ese texto cifrado.

2.7. Herramientas de Seguridad.

¿Por qué utilizar herramientas de seguridad en los sistemas Unix? Ningún sistema operativo se puede considerar “seguro” tal y como se instala por defecto; normalmente, cualquier distribución de un sistema se instala pensando en proporcionar los mínimos problemas a un administrador que desee poner la máquina a trabajar inmediatamente, sin tener que preocuparse de la seguridad. Es una cuestión de puro *marketing*: imaginemos un sistema Unix que por defecto se instalara en su modo más restrictivo en cuanto a seguridad; cuando el administrador desee ponerlo en funcionamiento conectándolo a una red, ofreciendo ciertos servicios, gestionando usuarios y periféricos, deberá conocer muy bien al sistema, ya que ha de dar explícitamente los permisos necesarios para realizar cada tarea, con la consiguiente pérdida de tiempo. Es mucho más productivo para cualquier empresa desarrolladora de sistemas proporcionarlos completamente abiertos, de forma que el administrador no tenga que preocuparse mucho de cómo funciona cada parte del sistema que acaba de instalar: simplemente inserta el CDROM original, el *software* se instala, y todo funciona a la primera, aparentemente sin problemas. Esta política, que lamentablemente siguen casi todas las empresas desarrolladoras, convierte a un sistema Unix que no se haya configurado minímamente en un fácil objetivo para cualquier atacante.

Es más, la complejidad de Unix hace que un administrador que para aumentar la seguridad de su sistema se limite a cerrar ciertos servicios de red o detener algunos demonios obtenga una sensación de falsa seguridad, esta persona va a pensar que su sistema es seguro simplemente por realizar un par de modificaciones en él, cosa que es completamente falsa.

a) Nessus. Es la herramienta de evaluación de seguridad "Open Source" de mayor renombre. Nessus es un escáner de seguridad remoto para Linux, BSD, Solaris y Otros Unix. Está basado en plug-in(s), tiene una interfaz basada en GTK, y realiza más de 1200 pruebas de seguridad remotas. Permite generar reportes en HTML, XML, LaTeX, y texto ASCII; también sugiere soluciones para los problemas de seguridad.

b) Ethereal. Oliendo el pegamento que mantiene a Internet unida. Ethereal es un analizador de protocolos de red para Unix y Windows, y es libre {free}. Nos permite examinar datos de una red viva o de un archivo de captura en algún disco. Se puede examinar interactivamente la información capturada, viendo información de detalles y sumarios por cada paquete. Ethereal tiene varias características poderosas, incluyendo un completo lenguaje para filtrar lo que querremos ver y la habilidad de mostrar el flujo reconstruido de una sesión de TCP. Incluye una versión basada en texto llamada tethereal.

c) Tripwire. El abuelo de las herramientas de comprobación de integridad de archivos. Un comprobador de integridad de archivos y directorios. Tripwire es una herramienta que ayuda a administradores y usuarios de sistemas monitoreando alguna posible modificación en algún set de archivos. Si se usa regularmente en los archivos de sistema (por Ej. diariamente), Tripwire puede notificar a los administradores del sistema, si algún archivo fue modificado o reemplazado, para que se puedan tomar medidas de control de daños a tiempo.

d) Titan. Este software de auditoría informática, detecta problemas de seguridad en la máquina local. El mismo se limita a informarnos de los posibles problemas de seguridad que podemos tener.

e) TCP Wrappers. TCP Wrappers se encarga de definir una serie de redes o máquinas autorizadas a conectar con nosotros. Cualquier administrador que desee un mínimo de seguridad ha de instalar TCP Wrappers en sus equipos; incluso algunos Unixes como Linux o BSDI lo ofrecen por defecto al instalar el operativo.

f) SSH (Secure Shell). Es un software cuya principal función es permitir la conexión remota segura a sistemas a través de canales inseguros, aunque también se utiliza para la ejecución de órdenes en ese sistema remoto o transferir archivos desde o hacia él de manera fiable.

g) Crack. Crack, desarrollado por el experto en seguridad Alec Muffet, es el “adivinador” de contraseñas más utilizado en entornos Unix; actualmente se encuentra en su versión 516.5, que funciona correctamente en la mayoría de clones del sistema operativo (Linux, Solaris, OSF...). Ejecutar periódicamente Crack sobre el archivo de

contraseñas de sus sistemas es algo muy recomendable para cualquier administrador minímamente preocupado por la seguridad, sin importar que se utilicen mecanismos para obligar a los usuarios a elegir passwords aceptables.

En este capítulo hemos tocado lo temas más importante de la seguridad en UNIX, de acuerdo a la investigación que se realizo en este trabajo de tesis los temas que exponemos son importantes para alcanzar un buen nivel de seguridad.