10
1 ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS ARAGON

SISTEMAS DE SEGURIDAD EN UNIX

T E S I S

QUE PARA OBTENER EL TITULO DE:

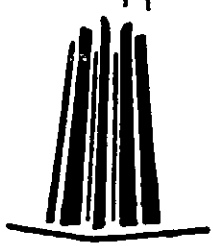
INGENIERO EN COMPUTACION

P R E S E N T A N:

SAMUEL CONDE VILLAGRANA

FELIPE JAVIER FLORES GOMEZ

ASESOR: ING. JUAN GASTALDI PEREZ



SAN JUAN DE ARAGON, ESTADO DE MEXICO **FEBRERO 1998**

Dedico esta tesis
a mis Padres y Hermanos.

AGRADECIMIENTOS

Quiero agradecer a mis padres y hermanos que siempre han estado cerca de mí y por su inestimable apoyo en todo momento.

También agradezco a familiares y amigos que de alguna u otra forma me han ayudado incondicionalmente a cumplir con esta meta.

Finalmente, un especial reconocimiento y agradecimiento a todos los profesores que con su tiempo y dedicación nos ayudaron a finalizar con este trabajo.

INDICE

CAPITULO 8

CAPITULO 9

CAPITULO 10

OBJETIVOS:

Dar a conocer las diferentes técnicas y herramientas para incrementar de manera considerable la seguridad en los sistemas basados en Unix.

Crear en el administrador del sistema conciencia de sus tareas e informar de los problemas a los que puede llegar a enfrentarse y sus posibles soluciones.

INTRODUCCION

Unix es un sistema operativo que fue diseñado en y para un ambiente de trabajo compartido, donde la protección de los datos no era lo más importante. Además, al ser un sistema operativo tan complejo, es difícil para los programadores preveer todas las posibles interacciones entre sus componentes, muchas de las cuales pueden resultar en problemas de seguridad.

Sin embargo, con el paso de los años, se le ha ido dando cada vez mayor importancia a los aspectos de seguridad en Unix, y actualmente es posible obtener niveles satisfactorios de seguridad en este sistema operativo.

Diferentes autores reconocen distintos tipos de seguridad en cómputo, de acuerdo a los distintos ámbitos en los que se manejan l ainformación y los sistemas de cómputo. Sin embargo, los más generales se pueden reducir a tres:

-Confidencialidad y control de acceso: Un sistema de cómputo no debe permitir que la información contenida en él sea accesible a nadie que no tenga la autorización adecuada.

-Integridad y autenticidad: Un sistema de cómputo seguro no debe permitir modificaciones no autorizadas a los datos contenidos en él. Esto comprende cualquier tipo de modificaciones: por errores de hardware o de software, causados por alguna persona de forma intencional, causados accidentalmente o por siniestros.

-Disponibilidad: La información puede estar sana y salva en el sistema, pero de poco sirve si los usuarios no tienen acceso a ella. La disponibilidad significa que los recursos del sistema se mantendrán funcionando de forma eficiente y que los usuarioslos podrán utilizar en el momento en que necesiten. Tambiém significa que el sistema sea capaz de recuperarse rápidamente en caso de ocurrir un problema de cualquier especie.

Como con todas las tecnologías, a lo largo de los años ha surgido la intención de legislar acerca de la seguridad en cómputo. El esfuerzo más reconocido es el del Departamento de Defensa de los Estados Unidos, que en 1985 emitió el Department of Defense Trusted Computer System Evaluation Criteria (TCSEC), mejor conocido como el "Libro Naranja". Este documento define criterios objetivos y muy específicos de evaluación que permiten clasificar a los sistemas de cómputo de acuerdo a sus características de seguridad.

La seguridad es un tema extremadamente subjetivo. Por lo tanto, no todos los sitios tienen la necesidad o la posibilidad de implementar todos los mecanismos que se van a describir. Es importante tomar una actitud realista ante la seguridad y no implementar todo lo nuevo solamente por hacerlo, sin realizar algún tipo de análisis previo.

Los pasos que se tienen que seguir se pueden resumir en tres preguntas:

-¿Qué se quiere proteger?

-¿Contra qué se quiere proteger?

-¿Cuánto tiempo, dinero y esfuerzo se puede invertir para lograrlo?

Los niveles de seguridad se pueden incrementar hasta niveles muy altos, si se cuenta con los recursos apropiados, pero se considera imposible eliminar por completo los riesgos.

Es un hecho que es posible obtener niveles razonables de seguridad, dedicándole tiempo y esfuerzo como administradores de sistemas.

CAPITULO 1

REDES

1.1 Objetivo de las redes.
1.2 Aplicación de las redes.
1.3 Estructura de la red.
1.4 Topologías de redes.

1.1 Objetivo de las redes

xisten muchas y diversas organizaciones que ya cuentan con un número considerable de computadoras en operación, y con frecuencia aislados unos de otros; por ejemplo: una compañía con varias fábricas puede tener una computadora en cada una de ellas para mantener un seguimiento de inventarios, observar la productividad, llevar la nómina local, etc.

Inicialmente cada uno de estas computadoras pudo haber estado trabajando de forma aislada de las demás, pero en algún momento la administración puede decidir interconectarlas para tener así la capacidad de extraer y correlacionar información referente a toda la compañía. El propósito consiste en compartir recursos y el objetivo es hacer que todos los programas, datos y equipos estén disponibles para cualquiera de la red que así lo solicite, sin importar la localización física del recurso y el usuario. En otras palabras, no debe evitar que éste los pueda utilizar como si fueran originados localmente. Otro aspecto de compartir recursos es el relacionado con la compartición de carga; este objetivo se puede resumir diciendo que es un intento por acabar con la imposición de la geografía.

Un segundo objetivo consiste en proporcionar una alta fiabilidad al contar con fuentes alternativas de suministro; por ejemplo: todos los archivos podrían duplicarse en dos o tres máquinas de tal manera que si una de ellas falla o no se encuentra disponible (como consecuencia de una falla de hardware) podría utilizarse alguna de las otras copias. Además la presencia de múltiples CPU's significa que si una de ellas deja de funcionar, las otras pueden ser capaces de encargarse de su trabajo aunque se tenga un rendimiento global menor. Para aplicaciones militares, bancarias, de control de tráfico aéreo y muchas otras, es muy importante la capacidad de carga de los sistemas para continuar funcionando a pesar de existir problemas de hardware.

Otro objetivo es el ahorro económico, las computadoras pequeñas tienen una mejor relación costo/rendimiento, comparada con la ofrecida por las máquinas grandes. Estas son a grandes rasgos diez veces más rápidas que el más rápido de los microprocesadores, pero su costo, es miles de veces mayor. Este desequilibrio ha ocasionado que muchos de los diseñadores de sistemas constituidos por poderosas computadoras personales, una por usuario, tengan los datos guardados en una o varias

máquinas que funcionan como servidor de equipo compartido. Este objetivo conduce al concepto de redes con varias computadoras localizadas en el mismo edificio. A este tipo de red se le conoce como LAN (Red de Area Local); en contraste con lo amplio de una WAN (Red de Area Amplia) a la que también se le conoce como red de gran alcance.

Un punto muy relacionado, es la capacidad para aumentar el rendimiento del sistema en forma gradual a medida que crece la carga, esto se hace simplemente añadiendo más computadoras.

Otro objetivo del establecimiento de una red de computadoras, es proporcionar un poderoso medio de comunicación entre personas que se encuentran muy alejadas entre sí. Con el empleo de una red es relativamente fácil para dos o más personas que viven en lugares separados tener la capacidad de escribir un informe juntos.

A la larga el uso de las redes para aumentar la comunicación entre los seres humanos puede ser más importante que los mismos objetivos técnicos, por ejemplo: la mejora de la confiabilidad. En la Figura 1.1 se muestra la clasificación de los sistemas de multiprocesadores distribuidos de acuerdo a su tamaño físico; y en la parte superior se encuentran las máquinas de procesamiento de datos (flujo de datos) que son ordenadores con un alto nivel de paralelismo y muchas unidades funcionales de trabajo en el mismo programa.

Después vienen los multiprocesadores que son sistemas que se comunican a través de memoria compartida, a continuación de los multiprocesadores encontramos las verdaderas redes, que son ordenadores que se comunican por medio de intercambio de mensajes.

1.2 Aplicación de las redes

El reemplazo de una máquina grande por estaciones de trabajo sobre una LAN no ofrece la posibilidad de introducir muchas aplicaciones nuevas, aunque podrían mejorarse la confiabilidad y el rendimiento. Sin embargo la disponibilidad de una WAN (pública) si genera nuevas aplicaciones viables.

y algunas de ellas pueden ocasionar importantes efectos en la totalidad de la sociedad. Para darnos idea sobre algunos de los usos importantes de redes de computadoras, se mencionará el acceso a bases de datos remotas y facilidades de comunicación a valor añadido; en un futuro próximo no será difícil

Distribución entre procesadores	Procesadores ubicados en:	Ejemplo
0.1 m.	Tarjeta del circuito	Máquina de flujo de datos
1.0 m.	El sistema	Multiprocesador
10.0 m.	El cuarto	Red de Area Local
100.0 m.	El edificio	Red de Area Local
1.0 Km.	La universidad	Red de Area Local
10.0 Km.	La ciudad	Red de Gran Alcance
100.0 Km.	El país	Red de Gran Alcance
1000.0 Km.	El continente	Interconección de redes de gran Alcance
10,000 Km	El planeta	Interconección de redes de gran Alcance

observar a cualquier persona hacer desde su casa reservaciones de hotel, avión, etc. para cualquier parte del mundo, obteniendo además la confirmación de manera inmediata. Dentro de esta clase también se encuentran las operaciones bancarias que se llevan a cabo desde el domicilio particular.

Todas estas aplicaciones operan sobre redes por razones económicas, es decir, si se quiere llamar a una computadora remota mediante una red, resulta más económico que hacerlo directamente. La posibilidad de obtener un costo más bajo es que el enlace telefónico normal utiliza un circuito caro y en exclusiva durante todo el tiempo que dura la llamada, mientras en el acceso a través de una red sólo se utilizan los enlaces de larga distancia cuando no están transmitiendo datos.

Una tercera forma que nos muestra el amplio potencial de uso de redes, es su empleo como medio de comunicación, los investigadores en informática ya toman como hecho poder enviar correo electrónico a cualquier parte del mundo desde sus terminales. En el futuro será posible para cualquier persona enviar y recibir correo electrónico y no sólo aquellas personas que se encuentran en el mundo de las computadoras, además este tipo de correo es capaz de transmitir voz digitalizada, así como fotografías e imágenes de televisión y vídeo, aunque se sigue tratando de mejorar la resolución.

1.3 Estructura de la red.

En toda red existe una colección de máquinas para correr programas de usuarios (aplicaciones), estas máquinas son denominadas servidores (en algunas ocasiones también se utiliza el término de equipo terminal de datos o sistema final), los servidores están conectados mediante una subred de comunicación. El trabajo de la subred consiste en enviar mensajes entre servidores, de la misma manera como el sistema telefónico envía palabras entre la persona que habla y la que escucha. El diseño de la red se simplifica notablemente cuando se separan los aspectos puros de la comunicación de la red (la subred) de los aspectos de aplicación, una subred en la mayoría de las redes de área extendida consiste en dos componentes diferentes: las líneas de transmisión y los componentes de conmutación.

Las líneas de transmisión se encargan de mover los bits (información) entre las máquinas; los elementos de conmutación son ordenadores especializados que se utilizan para conectar dos o más líneas de transmisión cuando los datos llegan por una línea de entrada, el elemento de conmutación deberá seleccionar una línea de salida para reexpedirlos.

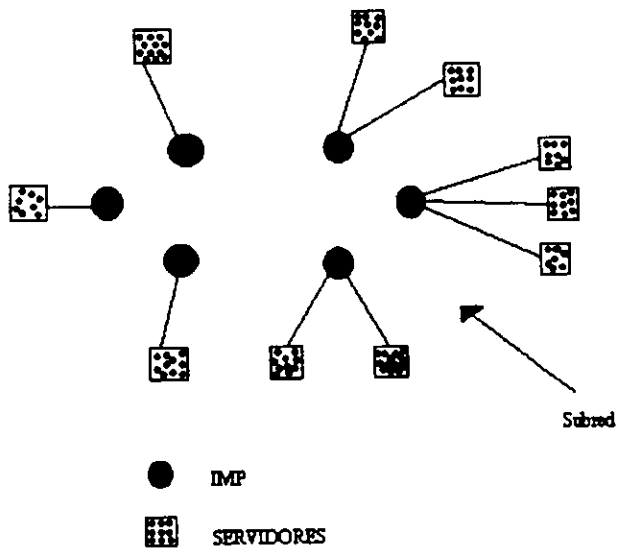


Figura 1.2 Relación entre los servidores y la subred

A los elementos de conmutación los llamaremos procesadores de intercambio de mensajes (IMP), aunque los términos nodo de conmutación de paquetes, sistema intermedio y central de conmutación de datos, también son empleados con frecuencia. En la ilustración que se muestra en la Fig.1.2 cada uno de los servidores está conectado a un procesador de intercambio de mensajes aunque en ocasiones lo haga con más de uno, por consecuencia todo el tráfico que va o viene del servidor pasa a través de un procesador de intercambio de mensaje.

En términos generales puede decirse que existen dos tipos de diseños para la subred de comunicación que son: 1. Canales punto a punto y 2. Canales de difusión. En el primero de ellos, la red contiene varios canales o líneas telefónicas alquiladas, conectando a cada una de ellas un par de procesadores de intercambio de mensajes, si dos procesadores de intercambio desean comunicarse y no comparten un cable común, deberán hacerlo de manera indirecta a través de otros procesadores de

intercambio de mensaje. Cuando un mensaje (que en contexto de subred normalmente se le denomina paquete) se envía de un procesador de intercambio de mensajes a otro, a través de uno o más procesadores de intercambio de mensaje intermedio, se almacenará ahí y no continuará su camino hasta que las líneas de transmisión den una salida para reexpedirlo, es decir, se encuentre libre. La subred que utiliza este principio se denomina punto a punto, de almacenamiento y envío o de conmutación de paquetes. Casi todas las redes de área extendida tienen redes del tipo de almacenamiento y reenvío.

La difusión se utiliza como un segundo tipo de arquitectura de comunicación y la utilizan la mayoría de las redes de área local y un número reducido de redes de área extendida. En una red de área local el procesador de intercambio de mensaje se reduce a un solo chip, el cuál se incluye en el interior del servidor, de tal manera que siempre habrá un servidor por cada procesador de intercambio de mensaje.

Los sistemas de difusión tienen un solo canal de comunicación que a su vez es compartido por todas las máquinas que constituyen la red. Los paquetes que una máquina cualquiera envía son recibidos por todas las demás, el campo de dirección localizado en el interior de un paquete especifica a quien va dirigido. En el momento en que se recibe un paquete se verifica el campo de dirección, y si el paquete está destinado a otra máquina este simplemente se ignora. Por lo normal los sistemas de difusión también admiten la posibilidad de dirigir un paquete a todos los destinos mediante el empleo de un código especial incluido en el campo de dirección, cuando se transmite un paquete con dicho código, éste es recibido y procesado por todas las máquinas de la red. Algunos sistemas de difusión también soportan la transmisión a un subconjunto de máquinas, lo cuál se conoce como difusión restringida.

1.4 Topologías de redes.

La topología de redes se refiere a la forma en que se conectan los dispositivos físicamente a la red, es decir, es un parámetro primario que condiciona fuertemente las prestaciones que de la red pueden obtenerse. El acierto en la elección de una u otra estructura dependerá de su adaptación en cada caso al tipo de tráfico que debe cursar y de una valoración de la importancia relativa de las prestaciones

que de la red se pretende obtener. Pueden seleccionarse sin embargo algunos criterios básicos que permitan efectuar comparaciones generales entre las topologías. Así convendrá analizar:

- Costo-modularidad. Esto es en cuanto al costo en medios de comunicación y a la sencillez de instalación y mantenimiento.

- Flexibilidad-complejidad. Por la dificultad que supone incrementar o reducir el número de estaciones.

- Fiabilidad-adaptabilidad. Por los efectos que una falla en una estación o en el medio de comunicación puedan provocar en la red, así como las necesidades de reconfiguración como procedimiento de mantener el servicio mediante encaminamientos (enrutamientos) alternativos.

- Dispersión-concentración. Por su adecuación a instalaciones con poca o mucha dispersión geográfica.

- Retardo-caudal. Por el retardo mínimo introducido por la red o su facilidad para manejar grandes flujos de información sin que se produzcan bloqueos o congestiones.

Una fuerte exigencia de alguna de estas características puede obligar a reducir a la instalación de una determinada red por el tipo de topología que se utiliza. Así para cubrir servicios donde la fiabilidad de la comunicación es de gran importancia, no debería utilizarse una red con topología de estrella, ya que una avería en el nodo central bloquea toda la red.

1.4.1 Topologías de una red de área amplia (WAN).

En la fig. se muestra una red de tipo malla completamente conectada, como se puede observar , esta red es una en que cada estación (J) tiene un enlace directo para cada una de las otras estaciones; la cuál tiene un enrutamiento definido, en donde un enlace único es usado para alcanzar cada destino.

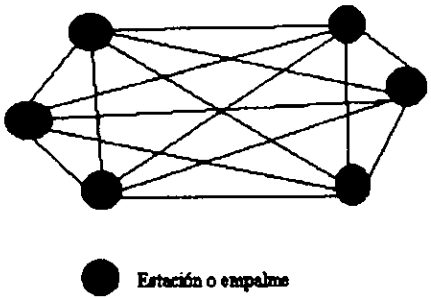


Fig.1.3 Malla completamente conectada

Una red de malla más realista es la llamada "malla irregular", en donde la estación es gobernada por los requerimientos del usuario. En el cuál los nodos de interconexión dependen de la disponibilidad de la ruta enlace, líneas de banda ancha y costo. Además la selección es mucho más sofisticada que en una malla completamente conectada. Este tipo de malla se muestra en la siguiente figura.

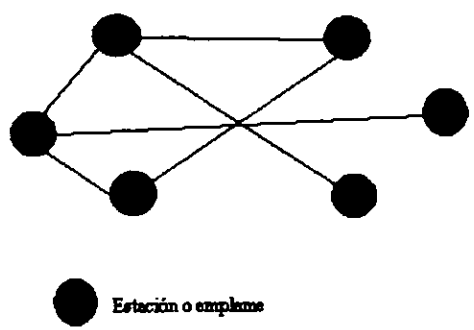


Fig.1.4 Malla irregular

1.4.2 Topologías de redes de área local (LAN).

Las redes de área local (LAN) fueron creadas para compartir recursos de información en el medio ambiente de una empresa u organización a nivel local. La expresión red de área local describe un método mediante el cuál los microordenadores pueden compartir información y recursos dentro de un área local limitada, generalmente inferior a una milla.

Una LAN exige que las estaciones de trabajo individuales estén unidas físicamente por algún medio de transmisión (usualmente cable coaxial o par trenzado) y que haya algún software de red residente en disco duro, que permita compartir periféricos, datos y programas de aplicaciones. Además las redes de área local son generalmente simétricas.

1.4.2.1 Topologías de estrella.

Todas las estaciones están unidas mediante medios bidireccionales a un módulo o nodo central, que efectúa nodos de conmutación como se ilustra en la Fig.1.5. Un ejemplo frecuente en redes locales es la adaptación de una central telefónica privada con conmutación de circuitos (PABX) a la interconexión de sistemas o recursos informáticos situados en plantas o edificios contiguos.

El nodo central asume además las labores de control y dispone de gran parte de los recursos informáticos comunes (memorias masivas, impresoras rápidas, etc.). El nodo aísla una estación de otra resultando una red fiable frente a averías en las estaciones. Sin embargo, una avería en el nodo principal deja totalmente bloqueada a la red y sin posibilidad de reconfiguración. La "flexibilidad-complejidad" es buena permitiendo incrementar o disminuir con sencillez el número de estaciones, ya que las modificaciones son sencillas y están todas localizadas en el nodo central. Puede resultar costosa por la longitud del medio de transmisión a instalar. No permite cursar grandes flujos de tráfico por congestionarse el nodo.

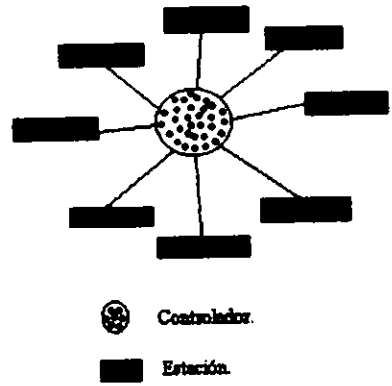


Figura. 1.5 Topología de estrella

1.4.2.2 Topología de árbol.

Es una extensión de la arquitectura de estrella; permite establecer una jerarquía clasificando a las estaciones en grupos y niveles según el nodo a que están conectadas y su distancia jerárquica al nodo central.

De características similares a la red de estrella, reduce la longitud de los medios de comunicación incrementando el número de nodos. Se puede adaptar a redes con grandes distancias geográficas y predominancia de tráfico local, características más propias de una red pública de datos que de una red privada local. Este tipo de topología se muestra en la figura 1.6.

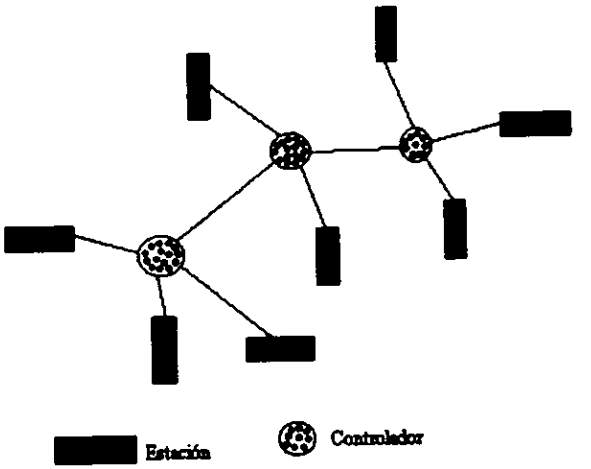


Figura.1.6 Topología de árbol.

1.4.2.3 Topologías de anillos simples y múltiples.

Los módulos de comunicaciones de las estaciones están interconectados formando un anillo, de modo que todas la información pasa por todos los módulos que únicamente envían a la estación los paquetes a ella destinados, como se muestra en la Figura 1.7. Aunque mediante la multiplexación de canales en frecuencia o transformadores híbridos, el anillo puede estar formado por un único medio de comunicación bidireccional; suele recurrirse a dos líneas separadas: una de transmisión y otra de conexión.

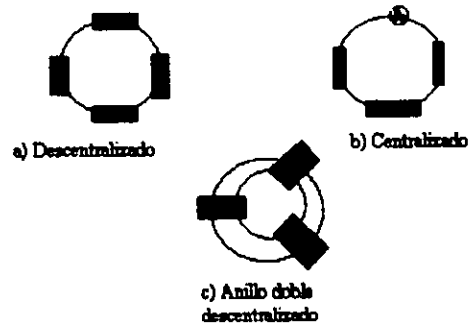


Figura.1.7 Anillo simple

La velocidad de transmisión puede ser así mayor y el transceptor es mucho más sencillo. En redes centralizadas el anillo incluye un controlador, lo que no es frecuente en redes locales, donde se prefieren los procedimientos distribuidos por ser más flexibles. El flujo que puedan cursar viene limitado por el ancho de banda del medio de transmisión, si el número de estaciones es elevado, el retardo total puede resultar excesivamente grande para determinadas aplicaciones en tiempo real, debido al retardo introducido por cada estación. Suele utilizarse para conectar sistemas informáticos de capacidad media y alta, especialmente si están bastante separados geográficamente (decenas de kilómetros).

La relación costo-modularidad es buena, así como la flexibilidad para incrementar el número de estaciones. La aparición de una falla en el medio de comunicación bloquea totalmente a la red sin posibilidad de reconfiguración. Para aminorar este problema se han estudiado y construido redes locales con dos o más anillos. La red DDLCN (Double Distributed Loop computer Network) desarrollada por M.Liu en la Universidad de Ohio es un ejemplo de topología de doble anillo.

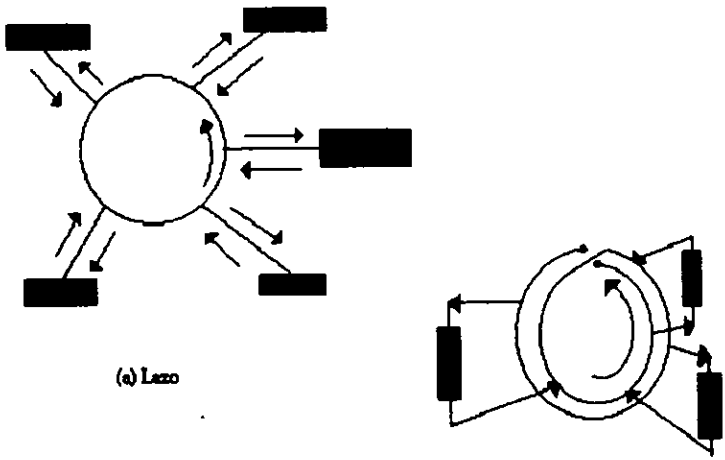
1.4.2.4 Topología de Bus y Multipunto.

Los módulos de comunicaciones están conectados de un único medio de comunicación (bus) que recorre todas las estaciones, al igual que en la estructura de anillo, no es necesario efectuar enrutamientos. Mientras que en el anterior los mensajes recorrían sucesivamente todas las estaciones siguiendo el orden de conexión, aquí la topología es de difusión y todas las estaciones reciben simultáneamente la información.

En aplicaciones a redes locales el control de acceso al medio suele ser distribuido. Sin embargo aunque forma parte de una red más compleja, la conexión suele efectuarse a través de un controlador que gestiona también el bus, a esta estructura se le denomina multipunto. Dentro de la topología de un bus distinguiremos entre bidireccional y unidireccional. El bus bidireccional se transmite en ambas direcciones por el mismo medio o medios conductores (bus paralelo), la transmisión suele efectuarse por división espectral, asignación secuencial en el tiempo o menos frecuentemente mediante transformadores híbridos o duplexores.

l bus unidireccional mediante amplificadores sencillos permite alcanzar distancias mayores (decenas de kilómetros). A cambio se requiere aumentar la longitud de cable utilizado. Son tres las formas o tipos de conexión más utilizadas: 1. Lazo, 2. Horquilla y 3. Espiral. El lazo es un bus que se inicia y termina en un controlador que centraliza la gestión, a diferencia de la topología en anillo con controlador, los módulos de comunicación no están incluidos en el bucle sino que cuelgan de el, como se muestra en la Figura 1.8.

En la espiral el tiempo que una estación tarda en recibir su propio mensaje es constante e igual para todas las estaciones, uniformando los detectores de bus ocupado que ya no dependen en su actuación del lugar que la estación ocupa en la red, obsérvese Figura 1.8b. Las topologías de un bus son en general las más sencillas de instalar, adaptándose con facilidad a la distribución geográfica de estaciones y con un costo reducido, especialmente los buses bidireccionales son para distancias no superiores a 1.5 Km.



(a) Lazo

(b) Espiral

Figura 1.8 Bus unidireccional.

Su gran modularidad de flexibilidad para variar el número de estaciones es una de sus principales ventajas. La conexión al bus debe de efectuarse mediante adaptadores pasivos y aislados, de modo que una avería en alguna de las estaciones no impida el correcto funcionamiento del resto de la red. Una avería sin embargo, en el medio de comunicación inhabilita el funcionamiento de toda la red o la separa en dos redes independientes, no existiendo posibilidad de reconfiguración.

Mientras el retardo de propagación es más reducido que en otras topologías (como el anillo), presenta mayores dificultades para una utilización eficiente de la capacidad del recurso, dando lugar a complejos algoritmos de control de acceso. Las estructuras unidireccionales son más costosas que las bidireccionales y sólo suelen justificarse cuando la longitud de la red obligue a utilizar amplificadores, o cuando por utilizar un medio de poca capacidad para la velocidad de transmisión, o para aumentar el número de servicios (voz, vídeo, datos, etc.) que se quiere incluir, resulte conveniente duplicar el medio de comunicación.

Las redes de banda ancha (broad band) responden a este último caso y suelen utilizar buses bidireccionales frente a los buses unidireccionales más frecuentes en redes de banda base (base band).

1.4.2.5 Topología múltiple.

Cuando las estaciones pueden agruparse en conjuntos de forma que el tráfico hacia otro conjunto en mucho menor que el interior, puede resultar preferible distribuirlas en varias redes en lugar de una, conectadas a través de un puerto o puente tal como se presenta en la Figura 1.9., sin que naturalmente sea necesario que todas las redes tengan la misma topología. algunas veces la división de una red en dos puede venir forzado por las propias restricciones de la topología o el método de acceso al cable. Así, por ejemplo, en una red Ethernet (bus bidireccional) las estaciones no pueden estar separadas más de 2.5 Km.

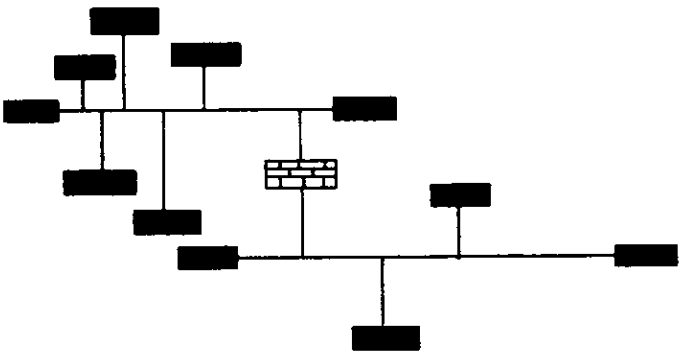


Figura 2.9 Conexión de dos buses mediante un puente

CAPITULO 2

SEGURIDAD EN REDES

2.1 Análisis de los niveles de seguridad.
2.2 Análisis de los asuntos de seguridad local.
2.3 Caducidad y control de la contraseña.
2.4 Vándalos y contraseñas.
2.5 Seguridad C2 y la base computacional confiable.
2.6 Como entender la equivalencia de red.
2.7 Como definir usuarios y grupos.
2.8 Como entender los permisos.
2.9 Como explorar los métodos de encriptación de datos.

2.1 Análisis de los niveles de seguridad.

De acuerdo con los estándares de seguridad en computadoras desarrollado por el Departamento de Defensa de Estados Unidos , el criterio estándar para la evaluación de computadoras confiables, se usan varios niveles de seguridad para proteger de un ataque al hardware, al software y a la información guardada. Los siguientes niveles describen diferentes tipos de seguridad física, autenticidad del usuario, confiabilidad del software tanto del sistema operativo como de las aplicaciones del usuario. Esos estándares también imponen límites en los diferentes tipos de sistemas que podrán ser conectados a su sistema.

2.1.1 Nivel D1

El nivel D1 es la forma más elemental de seguridad disponible. Este estándar parte de la base que asegura que todo el sistema no es confiable. No hay protección disponible para el hardware ; el sistema operativo se compromete con facilidad, y no hay autenticidad con respecto a los usuarios y sus derechos para tener acceso a la información que se encuentra en la computadora. Este nivel de seguridad se refiere por lo general a los sistemas operativos como MS-DOS , MS-Windows y System 7.x de Apple Macintosh.

Estos sistemas operativos no distinguen entre usuarios y carecen de un sistema definido para determinar quién trabaja en el teclado. Tampoco tiene control sobre la información que puede introducirse en los discos duros.

2.1.2 Nivel C1

El nivel C tiene dos subniveles de seguridad: C1 y C2 . El nivel C1 , o sistema de protección de seguridad discrecional, describe la seguridad disponible en un sistema típico UNIX. Existe algún nivel de protección para el hardware puesto que no puede comprometerse tan fácil, aunque todavía es posible. Los usuarios deberán identificarse a sí mismos con el sistema por medio de un nombre de registro del usuario y una contraseña. Esta combinación se utiliza para determinar que derechos de acceso a los programas e información tiene cada usuario.

Estos derechos de acceso son permisos para archivos y directorios. Estos controles de acceso discrecional habilitan al dueño del archivo o directorio, o al administrador del sistema, a evitar que algunas personas tengan acceso a los programas e información de otras personas. Sin embargo, la cuenta de la administración del sistema no está restringida a realizar cualquier actividad. En consecuencia, un administrador de sistemas sin escrúpulos puede comprometer con facilidad la seguridad del sistema sin que nadie se entere.

Además, varias de las tareas cotidianas de administración del sistema sólo puede realizarse al registrarse el usuario conocido como raíz. Con la centralización de los sistemas de computadoras de hoy día, no es raro entrar a una organización y encontrar a dos o tres personas que saben la contraseña raíz. Esto es un problema por sí mismo, puesto que no existe forma de distinguir los errores que hizo uno u otro usuario en el sistema el día de ayer.

2.1.3 Nivel C2

El segundo subnivel, C2 , fue diseñado para ayudar a solucionar tales hechos. Junto con las características de C1, el nivel C2 incluye características de seguridad adicional que crean un medio de acceso controlado. Este medio tiene la capacidad de reforzar las restricciones de los usuarios en su ejecución de algunos comandos o el acceso a algunos archivos basados no sólo en permisos , sino en

niveles de autorización. Además , este nivel de seguridad requiere auditorías del sistema. Esto incluye la creación de un registro de auditoría para cada evento que ocurre en el sistema.

La auditoría se utiliza para mantener los registros de todos los eventos relacionados con la seguridad , como aquellas actividades practicadas por el administrador del sistema. La auditoría requiere de autenticidad adicional porque si no, ¿Cómo puede estar seguro de que la persona ejecuta el comando es realmente quién dice ser? La desventaja de la auditoría es que requiere un procesador adicional y recursos de dloco del subsistema.

Con el uso de autorizaciones adicionales, es posible que los usuarios de un sistema C2 tengan la autoridad para realizar tareas de manejo de sistemas sin necesidad de una contraseña raíz. Esto mejora el rastreo de las tareas relativas a la administración, puesto que cada usuario realiza el trabajo en lugar del administrador del sistema.

Estas autorizaciones adicionales no deben confundirse con los permisos SGID y SUID que se pueden aplicar a un programa. En cambio, éstas son autorizaciones específicas que permiten al usuario ejecutar comandos específicos o tener acceso a las tablas de acceso restringido. Por ejemplo, los usuarios que no tengan la autoridad necesaria para analizar la tabla de procesos, verán sólo los procesos al ejecutar el comando ps.

2.1.4 Nivel B1

El nivel B de seguridad tiene tres niveles. El nivel B1, o protección de seguridad etiquetada, es el primer nivel que soporta seguridad de multinivel como la secreta y la ultrasecreta. Este nivel parte del principio de que un objeto bajo control de acceso obligatorio no puede aceptar cambios en los permisos hechos por el dueño del archivo.

2.1.5 Nivel B2

El nivel B2, conocido como protección estructurada, requiere que se etiquete cada objeto. Los dispositivos como discos duros, cintas o terminales podrán tener asignado un nivel sencillo o múltiple de seguridad. Este es el primer nivel que empieza a referirse al problema de un objeto a un nivel más elevado de seguridad en comunicación con otro objeto a un nivel inferior.

2.1.6 Nivel B3

El nivel B3, o nivel de dominios de seguridad, refuerza a los dominios con la instalación de hardware. Por ejemplo, el hardware y administración de memoria se usa para proteger el dominio de seguridad de un acceso no autorizado o la modificación de objetos en diferentes dominios de seguridad. Este nivel requiere que la terminal del usuario se conecte al sistema por medio de una ruta de acceso segura.

2.1.7 Nivel A

El nivel A, o nivel de diseño verificado es hasta el momento el nivel más elevado de seguridad validado por el libro naranja. Incluye un proceso exhaustivo de diseño, control y verificación. Para lograr este nivel de seguridad, todos los componentes de los niveles inferiores deben incluirse; el diseño requiere ser verificado en forma matemática; además, es necesario realizar un análisis de los canales encubiertos y de la distribución confiable. Distribución confiable significa que el hardware y el software han estado protegidos durante su expedición para evitar violaciones a los sistemas de seguridad.

2.2 Análisis de lo asuntos de seguridad local.

Además de los productos de seguridad o las regulaciones desarrolladas fuera de su organización, deberá trabajar para resolver los asuntos de seguridad que podrán ser locales o restringidos a su organización o a un subgrupo de ésta. Tales asuntos de seguridad local incluyen políticas de seguridad y controles de contraseña.

2.2.1 Políticas de Seguridad.

Se pueden establecer instancias principales en el desarrollo de políticas que reflejen la seguridad en su máquina. Estas instancias principales forman la base de todas las demás políticas de seguridad y regulan los procedimientos acomodados para implantarlas.

Aquello que no se permiten forma expresa, está prohibido, es el primer paso a la seguridad. Esto significa que su organización ofrece un grupo de servicios preciso y documentado, y que todos los demás está prohibido. Por ejemplo, si decide permitir transferencias ftp anónimas hacia y desde una máquina particular, pero no acepta servicios telnet, entonces el soporte documental para ftp ilustra este planteamiento, y no el de telnet.

La alternativa del planteamiento es aquello que no esté prohibido de manera expresa .Esto significa que a menos que usted indique en forma expresa que un servicio no está disponible, entonces todos los servicios estarán disponibles. Por ejemplo, si no dice con claridad que las sesiones de telnet a un anfitrión dado están prohibidas, entonces quiere decir que están permitidas.(Sin embargo, todavía puede evitar el servicio al no permitir una conexión con el puerto TCP/IP).

Sin importar que decisión tome, la razón para definir una política de seguridad, es determinar qué ación deberá tomarse en caso de que la seguridad de una organización se vea comprometida. La política también intenta describir qué acciones serán toleradas y cuáles no.

2.2.2 El archivo Password.

La primera línea de defensa encontra del acceso no autorizado al sistema es el archivo /etc/password. Por desgracia, también es el punto más debil. este archivo consiste de líneas o registros en las cuales cada línea se divide en siete campos con dos puntos(:), como lo ilustra el siguiente ejemplo:

chare:u7mHuh5R4UmVo:105:100:Chris Hare:/u/chare:/bin/ksh
username:encrypted password: UID:GID: comment: home directory:login shell

La tabla 2.1 explica el contenido y los valores típicos para cada archivo.

La tabla 2.1 proporciona información adicional que vale la pena mencionar. No es necesario que la contraseña encriptada real se coloque en el campo de contraseña encriptado. Este campo puede contener otros valores. Por ejemplo, para evitar que alguien registre con una cuenta específica, esa contraseña puede cambiarse a un valor no igualable, como NOLOGIN . Este campo, sin embargo, contiene por lo general una x o un asterisco (*), que indica que la contraseña se guarda en otro lado.

En esas situaciones, la contraseña se guarda ya sea en la base computacional confiable (TCB) , la cuál se explicará más adelante en este capítulo, o en el archivo shadow password.

Los permisos en el archivo password son de sólo lectura, lo que significa que nadie puede editar el archivo. De manera similar, el archivo shadow password y los archivos en el TCB son, en general, de sólo lectura también.

Tabla 2.1

El archivo /etc/passwd

Nombre del campo	Descripción	Valores tipicos
username	Este campo identifica al usuario con el sistema. Es en primer término para beneficio humano. Deben ser únicos en una máquina dada y de manera ideal, dentro de una organización.	chare rceh brb markd
encrypted password	Este campo contiene, o puede contener, la contraseña encriptada. La manera de encriptar contraseñas se explica más adelante en este capítulo.	u7mHuh5R4UmVo x * NOLOGIN
UID	Esta es la representación numérica del usuario en el sistema.	0-60,000
GID	Esta es la representación numérica del grupo login al cuál pertenece el usuario.	0-60,000
comment	Este contiene la información respecto al usuario. Con frecuencia lista un nombre completo de usuario, número de teléfono, u otra información.	Chris Hare Ops Manager, x273
home directory	Este es el directorio donde se ubica al usuario con base en login.	/u/chare /usr/lib/ppp/ppp-users
login shell	Este es el shell de registro que inicia para que los usuarios queden habilitados para interactuar con el sistema. Recuerde, no todos los shells de registro son interactivos.	/bin/sh /bin/csh /bin/ksh /bin/tcsh /usr/lib/uucp/uucico

La información de contraseña en esos archivos se actualiza mediante el comando password. Los permisos en el comando password incluyen el bit SUID (para establecer la ID del usuario), que hace que el usuario que ejecuta el programa parezca el dueño del mismo, o en este caso, raíz. Gracias a la aplicación del bit SUID , el usuario puede cambiar la contraseña aunque no tenga la autoridad para editar el archivo.

2.2.3 El archivo shadow password

Las versiones de UNIX que no incluyen las opciones de seguridad avanzada de Secure Ware pueden soportar al archivo shadow password. Cuando aparece el caracter x en el campo de contraseña, entonces la contraseña real del usuario se guarda en el archivo shadow password, /etc/shadow. A diferencia del archivo /etc/passwd, aquél deberá ser legible sólo para la raíz , como se ilustra en el siguiente ejemplo:

```
# ls -l /etc/shadow
-r——— 1 root        auth 861 oct 24 11: 46 /etc/ shadow
#
```

El formato del archivo /etc/shadow es idéntico al formato del archivo /etc/passwd , ya que ambos tienen 7 campos delimitados por dos puntos (:). Sin embargo, el primero sólo contiene el nombre del usuario, la contraseña encriptada y la contraseña de información caduca, como se ilustra en el siguiente ejemplo:

```
# cat /etc/shadow
root: DYQC9rXCioAuo:8887:0:0::
daemon:*::0:0::
mmdf:MZ74AeYMs4sl6:8842:0:0::
ftp:b80lug/921MeY:8363::::
anyone::8418::::
chare:7xqmj9fj3bVw2:9009::::
pppdemo:4QYrWsJEHc8lA:9062::::
#
```

El archivo /etc/shadow se crea mediante el comando pwconv en los dos sistemas que lo soportan. Sólo el superusuario puede crear el archivo shadow password. En los sistemas SCO, la información en /etc/passwd y la base de datos protegida con contraseña se utilizan para crear los archivos shadow password. En los sistemas SVR4, se usa la información de /etc/passwd.

La principal ventaja de utilizar el archivo shadow password es que coloca la contraseña encriptada en un archivo al que no tiene acceso los usuarios normales, así se reducen las posibilidades de que un intruso sea capaz de robar la información de la contraseña encriptada.

2.2.4 El archivo dialup password

La capacidad de instalar contraseñas adicionales en lineas de puertos en serie no está presente, por desgracia, en todas las versiones de UNIX. Estas se encuentran con más frecuencia ahora en los sistemas basados en SCO . La protección con contraseñas de marcación telefónica para estas lineas en serie se controla por medio de dos archivos: /etc/dialups y /etc/d_passwd, el primero contiene una lista de los puertos en serie protegidos por una contraseña de marcación telefónica. Ese archivo se ilustra en el siguiente ejemplo:

```
# cat dialups
/dev/tty2A
#
```

El archivo /etc/d_passwd se utiliza para identificar el shell de registro que corresponde a una contraseña específica. A diferencia de la contraseña normal, que emplea el nombre del registro del usuario, la contraseña de marcación telefónica va unida al shell de registro que un usuario determinado tiene en uso. Esto significa que cada usuario que utiliza el shell Bourne tiene la misma contraseña de marcación telefónica. El siguiente texto ilustra la contraseña típica de marcación telefónica.

```
# cat /etc/d_passwd
```

```
/bin/sh::
/usr/lib/uucp/uucico::
#
```

Cada línea o registro del archivo consiste de campos delimitados por la repetición de dos puntos (::). El primer campo identifica el shell que es para la contraseña especificada, el segundo campo lista la contraseña. En el ejemplo anterior, ningún shell tiene una contraseña. Esto significa que no se indicará al usuario que introduzca una contraseña cuando se registre.

La contraseña de marcación telefónica se agrega mediante el uso de la opción -m en el comando passwd. Esta opción le informa a passwd que la contraseña que busca es para un shell específico en el archivo de contraseña de marcación telefónica. La sintaxis para esta forma de comando es la siguiente:

```
passwd      -m shell_name
```

La ejecución de este comando se ilustra en el ejemplo siguiente:

```
# passwd -m /bin/s:
Setting modem password for login shell: /bin/sh
Please enter new password:
Modem password: testing
Re-enter password: testing
#
```

En el ejemplo anterior, el administrador del sistema agrega una contraseña de marcación telefónica al shell /bin/sh. Esto significa que a cualquier usuario que se registre en el sistema y tenga el shell Bourne como shell de registro se le indicará que presente la contraseña de marcación telefónica. En el ejemplo anterior se muestra la contraseña que debe introducirse como lo habría tecleado el administrador del sistema. Igual que en el comando passwd normal, la contraseña en cuestión no se despliega al momento de escribirse. Aquí se muestra sólo con el propósito de ilustrarlo. Observe que sólo el administrador del sistema puede cambiar la contraseña de marcación telefónica para un shell.

El comando passwd modifica el contenido del archivo d_passwd para incluir la nueva contraseña, como lo ilustra el ejemplo siguiente:

```
# cat d_passwd
/bin/sh:ORa691.Na1jjQ:
/usr/lib/uucp/uucico::
#
```

Como se observa en el ejemplo anterior, los usuarios del shell Bourne tienen ahora que proporcionar la contraseña adicional al registrarse en los puertos terminales especificados en el archivo /etc/dialups.

2.2.5 El archivo group.

El archivo /etc/group se utiliza para controlar el acceso a los archivos que no pertenecen al usuario . Recuerde como funcionan los permisos : si el usuario no es el dueño del archivo, entonces el (los) grupo (s) al cual pertenece se verifica (n) para saber si el usuario es miembro del grupo que posee el archivo. La lista de membresía de grupo se encuentra en /etc/group. El formato del archivo se muestra a continuación:

```
tech:*:103:andrewg, patc,chare,erict,lorrainel,lens
group name: password:GID:userlist
```

La tabla 2.2 explica cada uno de los campos contenidos en el archivo /etc/group.

Tabla 2.2
El archivo /etc/group

Nombre del campo	Descripción	Ejemplos
Group name	Este es el nombre del grupo. Tal nombre se utiliza en primer lugar con fines humanos.	tech sales group
Password	Esta es la contraseña a usar cuando se regresa a este grupo.	*
GID	Este es el numero de ID del grupo numérico en todos los archivos.	0-30,000
userlist	Es una lista de usuarios separados por comas que son miembros del grupo.	chare,andrewg

La contraseña para el archivo group no se utiliza, y no hay mecanismos sencillos que esten disponibles para instalar una contraseña en el archivo. Si un usuario trata de cambiar hacia un grupo del cual no es miembro, entonces se le indica que introduzca una contraseña, como se ilustra en el siguiente ejemplo:

```
$ newgrp tech
newgrp: Password
newgrp: Sorry
$ grep tech /etc/group
tech:*:103:andrewg, patc
$
```

Si el usuario que ejecutó este comando hubiera sido miembro del grupo tech, entonces el comando newgrp habría sido exitoso. Varia versiones actuales de UNIX, sin embargo, permiten al usuario ser un miembro de más de un grupo al mismo tiempo, para reducir o eliminar la necesidad de un comando newgrp.

Es importante considerar que Unix Berkeley extiende la protección de la cuenta raíz con el grupo wheel . Sólo los usuarios miembros del grupo wheel pueden utilizar el comando su para convertirse en raíz.

2.3 Caducidad y control de la contraseña.

Varias versiones de Unix ofrecen un servicio de caducidad de contraseña. Este mecanismo controla en qué momento pueden los usuarios cambiar sus contraseñas mediante la inserción de un valor en el archivo de contraseña después de la contraseña encriptada. Este valor define el periodo mínimo de tiempo que debe pasar antes de que los usuarios puedan cambiar sus contraseñas, y el periodo máximo de tiempo que puede transcurrir antes de que la contraseña expire.

Esta explicación es más clara si se imagina una línea de tiempo como la que se muestra en la Figura 2.1

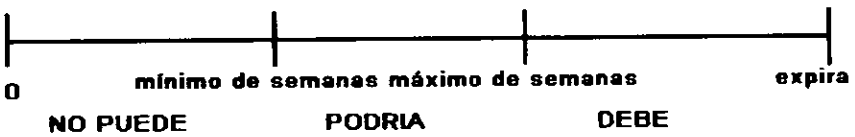


Figura 2.1 Caducidad de la contraseña y periodo de vida.

La información sobre el control de la caducidad de la contraseña se guarda junto con la contraseña encriptada, como una serie de caracteres susceptibles de impresión. Los controles se incluyen después de la contraseña, precedidos por una coma (,). En general, un número de caracteres después de la coma representan la siguiente información :

-El número máximo de semanas en que la contraseña es válida.

-El número mínimo de semanas que deben transcurrir antes de que el usuario pueda cambiar su contraseña otra vez.

-Cuándo se cambió la contraseña por última vez.

Los valores de caducidad de la contraseña se definen en la tabla 2.3

Tabla 2.3

Valores de caducidad en la contraseña

Carácter	Valor en semanas	Carácter	Valor en semanas
.	0	/	1
0	2	1	3
2	4	3	5
4	6	5	7
6	8	7	9
8	10	9	11
A	12	B	13
C	14	D	15
E	16	F	17

Tabla 2.3 Continuación

Carácter	Valor en semanas	Carácter	Valor en semanas
G	18	H	19
I	20	J	21
K	22	L	23
M	24	N	25
O	26	P	27
Q	28	R	29
S	30	T	31
U	32	V	33
W	34	X	35
Y	36	Z	37
a	38	b	39
c	40	d	41
e	42	f	43
g	44	h	45
i	46	j	47
k	48	l	49
m	50	n	51
o	52	p	53
q	54	r	55
s	56	t	57
u	58	v	59
w	60	x	61
y	62	z	63

Con esta información, y viendo una contraseña que tiene caducidad, podrá descifrar el significado del siguiente ejemplo:

chare: 2eLNss48eJ/GY,C2:215:100:Chris Hare:/usr1/chare:/bin/sh

En el ejemplo anterior, la duración de la contraseña se estableció con el valor C para definir el número máximo de semanas antes de que expire, mientras que 2 define el número mínimo de semanas que deberán pasar antes de que el usuario pueda cambiar otra vez la contraseña.

Dos condiciones especiales son reconocidas por los mecanismos de control de caducidad: uno obliga al usuario a cambiar su contraseña en su siguiente registro, y el otro evita que cualquier usuario pueda cambiarlo .

Para forzar a un usuario a cambiar su contraseña, como para un nuevo usuario, el campo de contraseña para ese usuario se puede modificar para incluir una coma (,), seguido por dos puntos (:) para los periodos máximos y mínimos. En este caso, se obliga al usuario a cambiar su contraseña en el siguiente registro. Después de esto, la información de control "force" se elimina de la entrada de una contraseña. Un ejemplo de la entrada forzada en la contraseña se muestra a continuación:

chare: 2eLNss48eJ/GY,./:215:100:Chris Hare:/usr1/chare:/bin/sh

El segundo caso especial le impide a un usuario cambiar su contraseña. Esta condición sucede al establecer el valor máximo para que sea menor que el valor mínimo (esto es, primero < segundo). En este caso , se le informa la usuario que su contraseña no puede cambiarse y se ilustra el siguiente ejemplo:

chare:,..:215:100:Chris Hare:/usr1/chare:/bin/sh

Con versiones más recientes y seguras de Unix en el mercado, quizá escuche el término periodo de vida de una contraseña. Este es un periodo de gracia después de un máximo periodo de tiempo en el cuál el usuario puede todavía registrarse en su cuenta con el uso de la contraseña expirada. Al llegar al término del periodo de vida, la cuenta se inhabilita. Cuando el usuario trata de registrarse en un sistema mediante el uso de una cuenta inhabilitada, se le informa que la cuenta ha sido inhabilitada y que vea al administrador del sistema.

El mecanismo de caducidad de la contraseña no evita que el usuario cambie su contraseña y más tarde la vuelva a cambiar a la original. Sólo algunas versiones del sistema Unix mantienen el rastro de las contraseñas que ha tenido el usuario. El proceso real de implantar la caducidad de una contraseña depende de la versión.

2.4 Vándalos y contraseñas.

El término hacker (destructor) no siempre tuvo una connotación tan negativa. En cambio era un término que denotaba a alguien que era persistente, que trataba de romper cosas y averiguar como funcionaban. Como resultado de esta reputación, y debido a que la mayoría de la gente que hacía destrozos eran sabios de la ciencia de la computación, el término hacker desarrolló una connotación negativa (vándalos). Sin embargo, para el propósito de esta discusión a los malos se les llamará vándalos.

Un vándalo desea ingresar a su sistema por una u otra razón. Alguna de esas razones pueden incluir lo siguiente:

*Sólo por diversión.
*Sólo para mirar.
*Para robar recursos de cómputo como tiempo de CPU.
*Para robar secretos del oficio u otra información propietaria.

Observe que no todos los intentos para ingresar a su sistema son dañinos. Sin embargo, en la mayoría de los casos deberían ser tratados como si lo fueran. Sin importar las razones que hay detrás del ataque, la pieza de información más codiciada por un vándalo es el archivo /etc/passwd. Cuando el vándalo tiene una lista de nombres de cuentas de usuario que es válida, resulta trivial crear un programa para simplemente adivinar las contraseñas. Sin embargo, muchos programas de registro modemo incluyen una demora de tiempo entre los indicadores de registro que se alargan más con cada intento

frustrado. Podrán incluir también un código de programa para inhabilitar el puerto de acceso en caso de registrarse demasiados intentos fracasados.

La fuente primaria de protección es utilizar /etc/shadow o la base de datos de contraseña protegida, ya que estos archivos o directorios requieren de acceso raíz
para poder ser observados. Esto dificulta al vándalo obtener la información de la contraseña encriptada. Sin embargo, recuerde que la información de contraseña en /etc/passwd está encriptada.

2.4.1 Para entender como " adivinan " las contraseñas los vándalos.

Es correcto decir que después de que una contraseña ha sido encriptada no puede ser desencriptada. Pero eso no significa que la contraseña ya esté segura. Un buscador de contraseñas, es un programa que utiliza el vándalo que intenta adivinar la contraseña en el archivo /etc/passwd mediante la comparación de éstas con las palabras de un diccionario. El éxito de un programa buscador depende de los recursos del cpu, la calidad del diccionario y el hecho de que el usuario tenga un a copia de /etc/psswd.

Los buscadores de contraseña son, en cierta manera, fáciles de crear.Uno que es simple aunque poco efectivo, puede escribirse en aproximadamente 60 líneas de código C o cuarenta líneas de código PERL. Para tratar de darles mejores contraseñas, un administrador de sistemas bien podría considerar escribirle uno. Si el programa es eficiente, o detecta algunas deficiencias en las contraseñas de su sistema, bien podría haber contribuido a debilitar la seguridad de su máquina. Un programa buscador eficiente puede ser robado y utilizado para ganar acceso a otras máquinas.Otras formas para mejorar la protección de su sistema están disponibles en las herramientas de seguridad lógicas.

Además, a causa de la posibilidad de que su programa de búsqueda pueda ser robado, habrían asuntos legales en caso de que hubiera daño directo como resultado del uso de ese programa. No debe tomarse esto a la ligera; es un ejemplo serio qué tan exitosos pueden ser estos programas.

En los últimos años, varios programas para buscar o adivinar contraseñas se han implantado en Internet. Esto no significa que debe ir corriendo a obtener uno. En realidad, éste podría ser el programa que usted no desea en su sistema. Los autores de estos programas dicen con claridad en sus documentaciones que no asumen responsabilidad por el uso de estos programas, aun así, aseguran que los programas son muy eficientes.

2.5 Seguridad C2 y la base computacional confiable.

La base computacional confiable(TCB) es parte del sistema de seguridad para los sistemas valorados como C2 de Unix. este sistema agrega un alto nivel de complejidad a la operación del sistema y a la administración del mismo. Este sistema trabaja mediante el traslado de bits en el archivo /etc/passwd hacia otros lugares, así como la adición de información sobre la información original. Los archivos que constituyen la base de datos para la base computacional confiable se encuentran diseminados en varias jerarquías de directorio. Sin embargo, no es una buena idea editar esos archivos, ya que podría causar severos daños a su sistema.

En un sistema que utiliza el TCB, se coloca un asterisco en el campo de contraseña de /etc/passwd. Esto es porque la contraseña de usuario real se guarda junto con otra información de usuario en la base computacional confiable (TCB). El usuario TCB no cambia la operación del sistema tanto como cuando Unix proporciona los mismos servicios al emplear TCB. En algunas versiones de Unix, como SCO Unix, incluso si no está utilizando la seguridad C2, la base computacional confiable todavía estará en uso para proporcionar servicios de seguridad.

La tabla 2.4 muestra los seis componentes de la base computacional confiable(TCB).

Tabla 2.4

La base computacional confiable

Componente	Descripción
/etc/passwd	El archivo de contraseña del sistema
/tcb/auth/files/	La base de datos de contraseña protegida.
/etc/auth/systems/ttys	La base de datos de control de terminal.
/etc/ayth/systems/files	La base de datos de control de archivos.
/etc/auth/subsystems/	La base de datos del subsistema protegido.
/etc/auth/system/default	La base de datos de los predeterminados del sistema.

Cuando se hace referencia a la base computacional de datos, incluye a todos los componentes de la tabla 2.4 de manera colectiva y no a cualquier componente individual.

La operación de un sistema confiable trae consigo algunos conceptos que deben entenderse para evitar poner el sistema en riesgo.

La base computacional confiable comprende una colección de software que incluye el kernel Unix y las utilerías que mantienen la base computacional confiable. estas utilerias incluyen authck para verificar y corregir problemas en la base de datos de contraseña y la integridad para verificar la exactitud de los archivos del sistema. Esta política es un grupo de reglas operativas que gobiernan la interacción entre los sujetos, como procesos, y objetos como archivos, dispositivos y objetos de comunicación interprocesal.

La contabilidad de una acción se define sólo si la acción puede rastrear hacia a una sola persona. En los sistemas tradicionales de Unix, en los cuales más de una persona conoce la contraseña raiz, es dificil, si no imposible, que la acción pueda ser rastreada a cualquier persona individual. Las pseudocuentas como cron y lp se ejecutan de manera anónima (su acción sólo puede ser rastreada después del cambio de información del sistema). Esto se corrige en un sistema de Unix confiable puesto que cada cuenta se asocia con un usuario real, cada acción se audita y cada acción se asocia con un usuario especifico.

Un breve proceso de identificación y autenticación (I&A) se realiza en los sistemas Unix tradicionales . En este sistema tradicional, el usuario se registra al proporcionar un nombre de registro y una combinación de contraseña, la que se valida mediante una búsqueda en el archivo /etc/passwd. Si el nombre de registro y la contraseña son correctos, entonces se le permite al usuario el acceso al sistema. En un sistema confiable, se utilizan algunas reglas adicionales para mejorar las técnicas estándares I&A . Por ejemplo, se han establecido nuevos procedimientos para cambiar y generar contraseñas, lo que brinda mejor protección a las partes de las bases de datos de la contraseña para evitar que curiosos intervengan.

El ejemplo siguiente ilustra una entrada típica de usuario a la base computacional confiable en un sistema SCO que detalla la información para un usuario específico. Esta información nunca se debe editar a mano, porque al hacerlo podría dejar su sistema en un estado de inutilidad.

```
chare: u_name=chare:\          #Nombre real del usuario
     : u_id#1003:\             #Identificación del usuario
     : u_pwd=rmwune/91rPqck:\  #Contraseña encriptada
     : u_type=general:\        #Tipo de usuario
     : u_succhg#746505937:\    #Ultimo cambio exitoso de contraseña
     : u_unsucchg#746506114:\  #Ultimo cambio fallido de contraseña
     : u_pswduser=chare:\      #
     : u_suclog#747066756:\    #Ultimo registro exitoso
     : u_suctty=tty02:\        #Ultimo registro exitoso en tty
     : u_unsuclog#747150039:\  #Ultimo registro fallido
     : u_unsuctty=tty04:\      #Ultimo registro fallido en tty
     : u_numunsuclog#1:\       #Número de registros fallidos
     : u_locka:\               #Estado del bloqueo
     : chkent:                 #
```

El ejemplo anterior ha sido modificado para insertar los comentarios en la entrada. El "# texto" no aparece en el archivo. Este ejemplo se incluye para ilustrar que otra información se está rastreando en el TCB .Esta no es toda la información, si no la que se encuentra en un archivo. Para cada usuario, se guarda un archivo con el nombre del usuario, el cuál contiene información mostrada en este ejemplo.

En el ejemplo anterior, la entrada para u_succhg muestra un valor de 746505937. Esta es la forma en que Unix mantiene un registro del tiempo. El valor es el número de segundos desde enero 1 de 1970. Es posible entregar el valor a una función en el Kernel de Unix que puede convertirla a la fecha y hora reales.

Los sistemas Unix tradicionales mantienen una cantidad limitada de información con respecto a la actividad del sistema, y en algunos casos sólo cuando han sido configurados para que así sea. En Unix confiable la auditoría es un elemento esencial para asegurar que las acciones tomadas se asocien con un usuario específico. El sistema audit escribe una serie de registros para cada acción que genera un rastro de auditoría de los eventos que han ocurrido en un sistema. Este rastro consiste de cada acción entre un sujeto y un objeto que es exitoso o no con respecto al acceso del objeto, los cambios hechos al sujeto o al objeto, y características del sistema. En consecuencia el sistema audit le brinda al administrador de auditoría una historia extensa de las acciones del sistema. Esto ayuda al administrador a determinar que ha sucedido, quién lo hizo y cuándo ocurrió.

2.6 Cómo entender la equivalencia de red.

Los dos tipos primarios de equivalencia de red son el acceso del anfitrión confiable y el acceso del usuario confiable. A través de este análisis, recuerde que muchos administradores de red prefieren utilizar estos recursos para entregar servicios mediante sus organizaciones sin entender cómo operan éstas, y qué efecto tienen en su anfitrión o la seguridad en la red.

2.6.1 Equivalencia de anfitrión.

La equivalencia de anfitrión, o acceso confiable, le ofrece al usuario un sistema para tener acceso a sus cuentas en sistemas remotos sin tener que utilizar los nombres de registro y contraseñas. Por medio del archivo /etc/hosts.equiv, el administrador del sistema puede listar a todos los sistemas confiables. Si no se identifican nombres de usuario para la entrada de la máquina, entonces todos ellos son confiables. De igual manera, si el administrador del sistema no especifica una máquina particular para todos los usuarios, cada usuario puede utilizar el archivo .rhosts en su directorio local para listar máquinas a las que desean dar acceso confiable.

Cada entrada en el archivo hosts.equiv es confiable. Esto significa que los usuarios en la máquina específica pueden introducir sus cuentas equivalentes en la máquina local sin una contraseña. Pero esto no se aplica para la raíz, pues sería un gran problema de seguridad. Para la raíz, y para el usuario que desea tener acceso a los sistemas no incluidos en la lista que aplica a todo el sistema, se requiere el uso del archivo .rhosts en el directorio base del usuario.

Considere el siguiente ejemplo del archivo /etc/hosts.equiv:

```
# cat /etc/hosts.equiv
macintosh.mydomain.com
delicious.mydomain.com
#
```

En el ejemplo anterior, las entradas permiten que cualquier usuario de estas máquinas se registre con el sistema local con el mismo nombre de cuenta sin utilizar una contraseña. Sin embargo, como se muestra en el siguiente ejemplo, es posible listar nombres de cuenta en este archivo.

```
# cat /etc/hosts.equiv
macintosh.mydomain. com andrewg
delicious.mydomain.com
#
```

Como el ejemplo anterior ilustra, el usuario andrewg en ese sistema puede registrarse mediante el uso de cualquier nombre de cuenta diferente a la raíz en el sistema local. En consecuencia, esto crea problemas potenciales, porque la cuenta de andrewg puede estar comprometida, y entonces un vándalo puede ingresar al sistema local desde esa máquina. Como resultado, no es recomendable usar nombres de cuenta en los archivos /etc/hosts.equiv y .rhosts.

Los asuntos de seguridad, con respecto a la equivalencia del anfitrión, incluyen la equivalencia raíz y los permisos de archivo. Es muy peligroso permitir la equivalencia raíz entre sistemas. Aunque esta equivalencia facilita completar las tareas de administración, también facilita la intrusión de vándalos en la red. Si la seguridad se compromete en una porción que tiene equivalencia raíz con otras porciones, al vándalo le toma unos segundos determinar esto y lograr el acceso a otras máquinas. En consecuencia, no es recomendable permitir la equivalencia raíz en el ambiente de red.

Los permisos en los archivos de acceso a la red /etc/hosts. Equiv y .rhosts, son otro problema. El archivo /etc/hosts.equiv deberá ser susceptible de ser escrito por raíz, aunque otros usuarios puedan leer el archivo. El archivo .rhosts sólo deberá aceptar escritura de su dueño. Tener el archivo editable por todo el mundo, e incluso para leer, puede crear problemas como permitir a los usuarios editar el archivo y añadir otros sistemas.

Algunas implantaciones de los comandos -r de Berkeley, que utilizan los archivos .rhosts y /etc/host.equiv, verifican sus permisos y rechazan el uso de éstos si los permisos se han establecido de manera inapropiada.

2.6.2 Equivalencia de usuario

El acceso de usuario confiable es más fácil de configurar, pero se puede dificultar mucho si se va a instalar después de que una lista de usuarios ya se configuró. La equivalencia de usuario es un concepto simple, diferente del acceso de anfitrión confiable. El acceso de anfitrión confiable no se requiere para que funcione la equivalencia de usuario. Para el uso de NFS (sistema de archivo de red), la equivalencia de usuario se convierte en algo obligatorio para evitar problemas de acceso.

La equivalencia de usuario se configura al darle a cada usuario de su red, y no sólo a la máquina, un nombre único de usuario y una UID numérica (identificación de usuario). Esto significa que

en cada máquina, el usuario tiene una cuenta con la misma UID. Si no desea permitir a un usuario el acceso a una máquina específica, entonces no necesita proporcionar una cuenta. Otra forma de explicar esto es decir que todos los archivos /.etc/passwd y /etc/group en las máquinas de la red serán los mismos.

Como podrá ver, si no utiliza equivalencia en sus redes puede poner en riesgo sus sistemas de archivo y datos al permitir el acceso no autorizado a ellos. Considere la siguiente lista de usuario:

Nombre del usuario	UID
chare	003
janicec	1009
terrih	1009
andrewg	1004

Cada uno de los usuarios de la lista anterior tiene un nombre único de usuario, pero sus números UID no son únicos. En la Figura 2.2 puede observar que janicec tiene su cuenta en macintosh.mydomain.com, y terrih tiene su cuenta en delicious.mydomain.com.

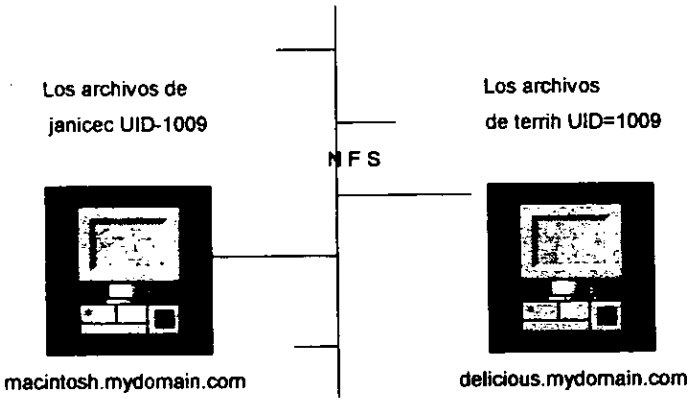


Figura 2.2

El sistema de archivo en el cuál terrih tiene su cuenta es parte de un sistema de archivo que ha sido exportado hacia los usuarios que trabajan en macintosh.mydomain.com. Cuando janicec establece acceso a los archivos en el directorio de terrih y los lista, el comando ls -l lista a janicec como dueño de los archivos porque los números UID son los mismos. Esto significa que janicec puede borrar o modificar la información en esos archivos como si fuera el auténtico dueño. En este caso, no importa que los nombres de usuario sean diferentes. La clave con NFS y equivalencia de usuario es la UID.

Otro tipo de problema puede ocurrir cuando la equivalencia de usuario no está configurada de manera correcta. Considere la siguiente lista de usuario:

Nombre de usuario	Sistema base
efudd (Elmer)	delicious
efudd (Elizabeth)	macintosh

En la lista anterior, existen dos usuarios llamados efudd. El usuario efudd en el sistema llamado delicious es Elmer Fudd, contrario a Elizabeth Fudd que utiliza el nombre de cuenta efudd en el sistema conocido como macintosh. En un esfuerzo para facilitar las cosas, el administrador del sistema en delicious ha configurado el acceso de anfitrión confiable, o equivalencia de anfitrión, para todos los demás sistemas en la red. Un día Elizabeth utiliza el comando rlogin para tener acceso a macintosh y se encuentra con que es posible registrarse sin tener que mostrar una contraseña. Puede observar todos los archivos que pertenecen a Elmer y tiene acceso a todos ellos. Esto sucede porque, hasta donde concierne a macintosh, Elizabeth es en realidad Elmer. En esta situación, el nombre de usuario es el factor determinante, y no la UID.

Como puede observar, ambos tipos de equivalencia de red, aunque crean un medio más productivo y apto para el usuario, podrán en realidad crear una lista de problemas de seguridad que son difíciles de detectar y corregir.

2.7 Como definir usuarios y grupos.

Como observó en la sección anterior sobre equivalencia de red, calcular con anticipación cómo manejará la inclusión de usuarios en su red es esencial para su defensa. El comentario de que sus archivos /etc/passwd y /etc/group serán los mismos en todos los sistemas es exacto. Sin embargo, una estrategia para asignar números UID y asegurar que son únicos se acerca más a la idea.

Esto es posible hacerlo de varias formas. Puede asignarlos en orden secuencial, repartir grupos de números a los departamentos o categorías de usuario, o a las oficinas. Sin importar el método, es esencial que sea el estándar para agregar usuarios.

2.8 Como entender los permisos.

Los permisos que utiliza Unix en cada archivo determinan cómo se controla el acceso a los archivos y directorios. Se pueden prevenir muchas situaciones con la aplicación correcta de este mecanismo simple pero poderoso. La siguiente sección analiza cómo se manejan los permisos en el ambiente Unix.

2.8.1 Una revisión a los permisos estándares.

Los permisos que se aplican a un archivo se basan en las UID y GID (identificación de grupo) estampadas en el archivo y en la UID o GID del usuario que trata de tener acceso al archivo. Los tres grupos de permisos son los siguientes:

*Los aplicables al dueño del archivo.

*Los usuarios que tienen la misma GID que tiene el archivo.

*Todos los demás usuarios.

Para cada categoría de usuario hay tres bits de permiso : leer, escribir y ejecutar. Esto significa que para cada archivo o directorio hay nueve bits de permiso. Estos bits están representados en el siguiente ejemplo:

```
$ ls -l output
-rw-r--r-- 1 chare    users 236 Aug 24 20:13 output
$
```

En la salida del ejemplo anterior, los permisos son :

-rw-r--r--

El primer guión representa el tipo de archivo, el cuál puede ser uno de los tipos que se listan en la tabla 2.5, y los caracteres sobrantes representan los permisos.

Tabla 2.5
Tipos de archivo en Unix

Símbolo	Descripción	Explicación
-	Archivo normal	Un archivo que contiene un programa, datos o texto.
d	Directorio	Un tipo especial de archivo que contiene una lista de archivos y elíndice para su localización en el disco.
b	Archivo para bloqueo de dispositivos	Un archivo que permite el acceso a dispositivos como unidades de disco.
c	Archivo para dispositivo de caracter	Un archivo que permite el acceso a dispositivos como terminales y módems.
l	Enlace simbólico	Un apuntador hacia otro archivo que puede estar en éste u otro sistema de archivos.
p	Conducto nombrado	Un método de comunicación entre dos procesos.

En los permisos, los símbolos r,w y x representan los permisos de lectura, escritura y ejecución, respectivamente. El permiso de lectura significa que el usuario solicitante puede analizar el contenido del archivo o directorio. El permiso de escritura garantiza al usuario la autoridad para modificar archivos o crear nuevos en un directorio. Por último, el permiso para ejecutar permite que el archivo sea ejecutado como un comando. La tabla 2.6 resume estos permisos y su efecto en un archivo o directorio.

Tabla 2.6

Permisos de archivo y directorio

Símbolo de permiso	Significado	Efecto en los archivos	Efecto en los directorios
r	Lectura	Analiza el archivo con cat o más.	Lista el directorio con ls.
w	Escritura	Modifica el archivo con vi	Crea o desaparece archivos.
x	Ejecuta	Ejecuta el archivo como un comando	Busca un archivo en el directorio; utiliza el comando cd.

Cuando los permisos de un archivo o directorio no se establecen en forma apropiada, se crean puntos de ingreso al sistema para los vándalos y también ofrecen el potencial para cualquier usuario a cometer errores e invitar al desastre. El siguiente ejemplo muestra algunos problemas relacionados con los permisos.

Este ejemplo ilustra un directorio que no tiene permiso de escritura, lo que significa que ese archivo no puede ser eliminado con rm.

```
$ id
uid=1009(terrih) gid=101(users)
$ ls -l
total 4
dr--r--r--      2 chare users   48 Aug 24 21:09 a
dr-xr-xr-x      2 chare users   48 Aug 24 21:12 a2
drwxr--xr--x            2 chare users   32 Aug 24 21:08 micrc_light
$ ls -ld a2
```

```
dr-xr-xr-x        2 chare users   48 Aug 24 21:12 a2
$ls a2
output
$ rm a2/output
rm: a2/output not removed. Permission denied
$ ls -l a2
total 1
-rw-rw-r--        1 chare users   133 Aug 24 21:12 output
$ date > a2/output
$ ls -l a2
total 1
-rw-rw-r--        1 chare users   29 Aug 24 21:14 output
```

Como advertirá el usuario, no puede eliminar el archivo con el comando rm porque el directorio no tiene permiso de escritura. Recuerde, un directorio es sólo un archivo de tipo especial. Cuando el usuario verifica el permiso en la misma salida del archivo, observa que el archivo tiene el permiso de escritura funcionando. Esto significa que el contenido del archivo se puede cambiar, como se mostró, al redirigir la salida del comando de fecha hacia el archivo.

Lo que no se ha mencionado aún es que el usuario que ejecuta estos comandos no es el dueño del archivo. Aquí, por medio de un error común, chare no protegió sus archivos de terrih porque el contenido de salida puede ser borrado aún cuando el pensó que había tomado las precauciones debidas.

Estas son las fallas del administrador del sistema. Muchos ataques tienen éxito, no porque el administrador del sistema no lleve a cabo su trabajo, sino porque a pesar de la mejor de las intenciones, el usuario deja este tipo de huecos donde sea. Algunos recursos, como ftp y los comandos -r , verifican los permisos de los archivos utilizados por estos comandos y evitan su uso si los permisos de archivo no son correctos.

El Set User ID (SUID) y el Set Group ID (SGID) son la mayor parte de la seguridad en un sistema Unix. Los bits SUID y SGID permiten al usuario asumir otra identidad mientras ejecuta el programa. Por ejemplo, el comando passwd utiliza el bit SUID para permitir a los usuarios convertirse en raíz para poder cambiar el contenido del archivo de contraseña.

En el caso del bit SUid, el usuario que ejecuta el programa asume la identidad del dueño del programa. Para SGID, el usuario que ejecuta el programa se convierte en un miembro del grupo que es dueño del programa.

2.8.2 Raíz y NFS.

Cuando piensa usted acerca de la raíz, piensa en un acceso no controlado a los archivos, directorios, programas y dispositivos en un sistema. Sin embargo, cuando la raíz trata de tener acceso a los archivos de un sistema remoto por medio de NFS, el usuario raíz obtiene limitado o ningún permiso. Esto se debe a las características de seguridad construidas dentro de NFS. Estas características de seguridad buscan un valor UID de cero. Cuando encuentran tal valor, saben que está es una raíz, y vuelven a hacer un mapa del valor UID a 65534. o -2. Esto significa que más allá de NFS, la raíz cae dentro de la otra categoría de usuario

La ventaja, en este caso, es que si no tiene equivalencia de raíz entre las máquinas de su red y una resulta comprometida, se hace más difícil para el intruso propagarse por sus sistema de archivo.

2.9 Como explorar los métodos de encriptación de datos.

La oportunidad de encriptar información para brindar un nivel más alto de seguridad para su sistema y sus datos es de interés para los usuarios y administradores del sistema. Sin embargo, hasta los datos encriptados pueden estar en riesgo sin la supervisión adecuada y el entrenamiento para los usuarios que deseen utilizar estos recursos.

2.9.1 Como encriptar las contraseñas.

En alguna ocasión, las contraseñas se guardaban en un formato de texto simple, y sólo el administrador y el software del sistema tenían acceso a tal archivo. Sin embargo, sucedían varios problemas al editar el archivo de contraseña (etc/passwd). La mayoría de los editores crearon un archivo temporal, el cual es el archivo editado en realidad. En este punto, el archivo sería leído por todos, al facilitar las contraseñas para todas las cuentas.

Como resultado, se desarrolló un método de encriptación de contraseña que utiliza un algoritmo de encriptación de un vía. Así los valores de encriptación se guardan en lugar del texto. Sin embargo, la seguridad del sistema es sólo tan buena como el método de encriptación elegido.

Cuando un usuario se registra en un sistema Unix, el programa getty le pide al usuario su nombre de usuario y luego ejecuta el programa de registro. El programa de registro indica la contraseña pero no lo desencripta. En realidad, el programa de registro encripta la contraseña y luego compara el valor encriptado reciente con el que está guardado en /etc/passwd. Si coinciden, entonces el usuario proporcionó el valor correcto.

El método de encriptación de Unix para el encriptado de contraseñas se introduce por medio de un mecanismo de kernel nombrado crypt(3). Gracias a los asuntos de licencias federales de Estados Unidos, las rutinas crypt quizá no están disponibles en su máquina. La razón es que mientras las rutinas que se necesitan para encriptar información están disponibles, esos programas que desencriptan información no están disponibles fuera de Estados Unidos.

El valor de la contraseña real guardado en /etc/passwd es el resultado de emplear la contraseña del usuario para encriptar un grupo de 64 bits de ceros al utilizar la llamada crypt(3). Clear text es la contraseña del usuario , la cual es la clave para la operación de encriptación. El texto a ser encriptado es de 64 bits de ceros, y el texto cifrado resultante es la contraseña encriptada.

El algoritmo crypt(3) se basa en el estándar de encriptación de datos(DES) desarrollado por el Instituto Nacional de Estándares y Tecnología o NIST. En operación normal de acuerdo con el estándar DES, una clave de 56 bits, como ocho caracteres de siete bits, se utiliza para encriptar el texto original,

el cual es llamado texto llano. El texto llano es, por lo general, de 64 bits de largo. El texto cifrado resultante no puede desencriptarse con facilidad sin saber la clave original.

La llamada crypt(3) de Unix utiliza una versión modificada de este método, al establecer que el texto llano se encripta en un grupo de ceros. El proceso es complicado al tomar el texto cifrado resultante y encriptandolo de nuevo con la contraseña del usuario como clave. Este proceso se realiza 25 veces. Cuando finaliza, los 64 bits resultantes se dividen en 11 caracteres y luego se guardan en el archivo de contraseña.

A pesar del hecho de que la fuente para crypt puede obtenerse de varios vendedores, aunque su distribución comercial se encuentra limitada fuera de U.S.A. (puede encontrar versiones publicadas del código en Internet), no hay método conocido disponible para traducir el texto cifrado o el valor encriptado de regreso a su texto llano original.

Robert Morris , padre y Ken Thompson, quienes implantarón la tecnología crypt(3) en Unix en sus inicios, temian que con el advenimiento del hardware de chips DES, la seguridad del sistema Unix sería traspasada con facilidad. Mediante el uso de " un grano de sal ", se las arreglaron para evitar esta amenaza.

El grano de sal, que se conoce como sal, es un número de 12 bits que utiliza para modificar el resultado de la función DES. El valor de este número de 12 bits va de cero a 4095. Así que para cada contraseña posible, existen 4096 formas de encriptación y almacenamiento para cada contraseña en el archivo de contraseña. Es posible para varios usuarios en la misma máquina utilizar la misma contraseña, y ninguno, incluido el administrador del sistema, sería el mas listo.

Cuando un usuario ejecuta el programa /bin/passwd para establecer una nueva contraseña, el programa /bin/passwd elige una sal basada en la hora del día. Esta sal se utiliza para modificar la contraseña del usuario.

El problema viene después, al encriptar la contraseña cuando el usuario se registra. Es posible, pero no probable, que el usuario se registre y la sal sea la misma. Para que las cosas marchen bien, Unix también guarda la sal en /etc/passwd. En realidad, hace los dos primeros caracteres de la contraseña encriptada. A continuación encontrará una muestra de un valor encriptado.

2eLNss48eJ/GY

En el ejemplo anterior, los dos caracteres iniciales (2e) son la sal para esta contraseña. Cuando el usuario se registra en el sistema, el programa de registro extrae la sal de la contraseña guardada y la utiliza para encriptar la contraseña que proporciona el usuario. Si la nueva contraseña encriptada y la almacenada son iguales, entonces la contraseña introducida por el usuario es correcta y estará registrado en el sistema. Si los valores no se igualan, entonces el usuario observará el mensaje Login incorrect, lo que indica que deberá intentarlo otra vez.

2.9.2 Cómo encriptar archivos.

La encriptación de contraseñas mediante el uso de un mecanismo que es difícil de desencriptar, ofrece un método relativamente seguro para evitar que usuarios no autorizados tengan acceso al sistema. Pero, ¿cómo pueden los usuarios evitar la entrada no autorizada a sus archivos? Esto puede lograrse por medio del comando crypt(1). Sin embargo, éste no es un método de encriptación muy seguro. Es interesante que algunos comandos Unix soporten la manipulación directa de estos archivos encriptados sin tener que desencriptarlos primero.

Encriptar archivos con el comando crypt es bastante simple. Si no se proporcionan argumentos en la línea de comando, entonces crypt indica la clave, lee los datos a encriptar de una entrada estándar e imprime la información encriptada en la salida estándar. Sin embargo, idealmente, la información a usar se proporciona en la línea de comando, como se ilustra en el ejemplo siguiente:

```
crypt key < clear> cipher
```

El comando anterior lee el archivo clear, encripta el texto con la clave de contraseña y guarda el texto encriptado resultante en el archivo cipher. Los archivos encriptados pueden observarse o desencriptarse mediante el uso de una línea de comando similar, como se muestra a continuación:

```
crypt key < cipher > clear
crypt key < cipher > | pr | lp
```

En el primer comando del ejemplo anterior , el texto encriptado en cipher se desencripta mediante el uso de key y se guarda en el archivo llamado clear. El segundo ejemplo de línea de comando utiliza crypt para desencriptar el texto y enviar el resultado del texto llano a pr para ser formateado y luego a lp para impresión. Los archivos generados por crypt pueden ser editados por de o vi, tomando en cuenta que la versión de éstos editores que tienen en un sistema soporta la edición de archivos encriptados.

El mecanismo exacto que utiliza crypt está bien documentado , varias versiones de éste comando están disponibles al público en Internet. Crypt (1) no usa las mismas rutinas de encriptación que crypt (3), el cuál se emplea para la encriptación de contraseñas. El mecanismo es una máquina de encriptación de un motor diseñado junto con las líneas de German Enigma, pero mediante un motor de 256 elementos. Aunque son conocidos los métodos de ataque a tales tipos de máquinas de encriptación , la cantidad de trabajo requerido puede ser grande.

La clave de encriptación que se utiliza es el factor limitante en la determinación del nivel de esfuerzo para desencriptar los datos. Mientras más larga es la contraseña, más complejos son los patrones de encriptación, y más tiempo toma transformar la clave a los arreglos internos utilizados por la máquina. Por ejemplo, el proceso de transformación es para que tome cerca de un segundo, pero si la clave se restringe a tres letras minúsculas los archivos encriptados pueden ser leídos en sólo una fracción sustancial de cinco minutos del tiempo real de la máquina.

Como la clave de crypt podría ser vista por curiosos que utilizan ps, crypt destruye cualquier rastro de la llave real luego de hacer la entrada. En consecuencia, como cualquier otro sistema de seguridad, la contraseña empleada para encriptar los datos es el componente más crítico, y el más sospechoso.

CAPITULO 3

CONCEPTOS SOBRE UNIX RELATIVOS A LA SEGURIDAD

3.1 Arquitectura de UNIX.
3.2 Cuentas de usuario.
3.3 Tipos de archivos.
3.4 Sistemas de archivos.
3.5 Permisos de los archivos.
3.6 Comandos.
3.7 Ejecutable binario,imagen y proceso

3.1 Arquitectura de UNIX.

UNIX es un sistema operativo, y como tal gestiona los recursos físicos del ordenador: UCP, memoria, disco, y otros periféricos. Técnicamente se caracteriza por ser:

-Multiusuario: permite que varios usuarios trabajen simultáneamente en el mismo ordenador.

-Multitarea: un usuario puede ejecutar simultáneamente varios programas.

-En tiempo compartido. El sistema operativo distribuye equitativamente los recursos del ordenador entre todos sus usuarios. A partir de UNIX System V Release 4, este sistema puede utilizarse para aplicaciones en tiempo real débil, es decir, aquellas cuyo tiempo de respuesta no puede garantizarse más que en términos estadísticos.

Tal y como se muestra en la figura 3.1, cuando un usuario entra a trabajar en un ordenador bajo UNIX le atiende un programa denominado intérprete de comandos ("Shell") que le permite ejecutar los más de 200 programas que lo integran. Sin embargo, desde un punto de vista técnico, el sistema operativo UNIX es tan sólo el programa que gestiona los recursos del ordenador, al cuál suele llamarse núcleo. Pero, por abuso del lenguaje, habitualmente cuando se habla de sistema operativo UNIX, o tan sólo UNIX, se está haciendo referencia al núcleo junto con los comandos disponibles. Éste es el significado que se aplicará en este capítulo.

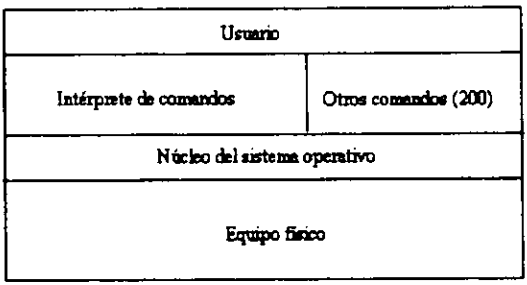


Figura 3.1 Perspectiva de un sistema UNIX

En la Figura 3.2 se presenta la evolución de la informática como un colchón de programas que se intercalan entre el usuario final y el equipo físico para que las personas puedan utilizar las computadoras de una forma conveniente a sus necesidades. Este colchón puede dividirse en un conjunto de niveles, donde cada uno de ellos representa a una computadora abstracta respecto al nivel superior. De esta manera, el equipo físico tan sólo es manejado por el núcleo del sistema operativo, y éste exporta un conjunto de llamadas al sistema para realizar peticiones a través de los compiladores que soportan las bibliotecas, comandos y programas que dan servicio al usuario final.

3.2 Cuentas de usuario.

Para que una persona pueda utilizar un ordenador bajo UNIX, el administrador del sistema debe abrirle una cuenta que físicamente se representa mediante un directorio perteneciente a dicho usuario. Para que éste pueda acceder a su cuenta, el administrador le asigna un nombre de conexión público junto con una

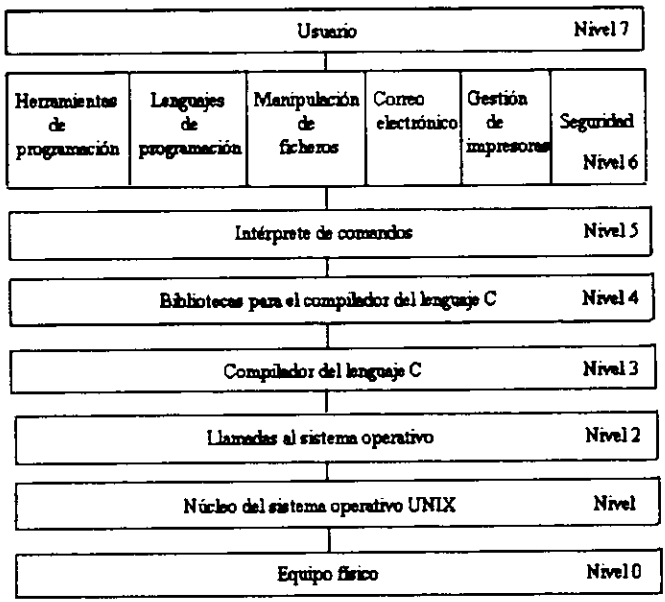


Fig 3.2 Arquitectura del sistema operativo UNIX

contraseña secreta. A partir de este momento, el usuario puede entrar en su cuenta a través de una terminal del ordenador, tal y como aparece en la Figura 3.3.

¿Cómo identifica UNIX a los usuarios que trabajan en el ordenador?. Cuando el administrador abre una cuenta significa que la registra en el archivo del sistema /etc/passwd en el que hay la siguiente información:

- Nombre de conexión del usuario.

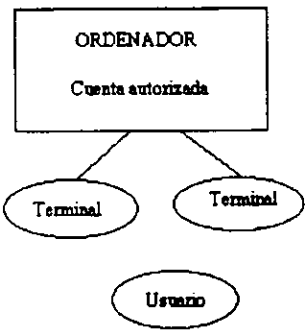


Figura 3.3 Relación entre usuario y ordenador a través de la cuenta

- Contraseña cifrada. A partir de UNIX System V Release 4 la contraseña cifrada no es de acceso público (/etc/shadow), por lo que es más difícil que un pirata pueda adivinarla mediante un programa rompedor. Desafortunadamente, en más del 90% de la base mundial instalada, la contraseña cifrada sí es de acceso público, por lo que estos sistemas tienen una brecha potencial de seguridad.

- Identificador de usuario ("User Identificator", UID). Es un número entero que representa al usuario ante el sistema operativo.

- Identificador de grupo ("Group Identificator", GID). En UNIX, cada usuario tiene su propia cuenta. ¿Pero qué sucede cuando varios usuarios desean compartir sus respectivas cuentas?. Lo mejor es crear un grupo de usuarios, al que el sistema operativo controla mediante su identificador. Por tanto, cada usuario pertenece por defecto a un grupo, aunque puede pertenecer a varios si está dado de alta en el archivo de grupos del sistema /etc/group.
- Campo con un comentario, habitualmente el nombre y apellidos del usuario.

- Directorio de trabajo que representa la cuenta del usuario.

- Programa que actúa como intérprete de comandos.

De esta manera, cuando el usuario completa la conexión al ordenador, el sistema operativo conoce sus identificadores de usuario, de grupo, directorio de trabajo e intérprete de comandos, con lo que el mismo está plenamente identificado ante el sistema y puede solicitar los servicios que necesite.

A continuación se muestra un ejemplo del contenido del archivo /etc/passwd, cuya estructura se ha explicado, y donde el carácter ":" separa a los diferentes campos:

```
root:*:0:1:Superuser:/:/bin/ksh
bin:*:2:2:Owner of system commands: /bin:
sys:*:3:3: Ownwer of system files : /usr/sys:
adm:*:4:4:system accounting:/usr/adm:
uucp:*:5:5:uucp administrator:/usr/lib/uucp:
asg: *:8:8:Assignable devices :/usr/tmp:
cron:*:9:16: Cron daemon:/usr/spool/cron:
sysinfo:*:11:11: System information :/usr/bin:
dos:*:16:11:DOS device: /tmp:
listen:*:37:4:Network daemons:/usr/net/nls:
lp:*:71:18:Printer administrator:/usr/spool/lp:
ingres:*:777:50:Database administrator:/usr/ingres:
ejemplo:*:200:50::/usr/juan:/bin/ksh
jgomez:*:202:50:Juan Gómez:/usr/prico:/bin/sh
prico:*:203:50:Pedro Rico : /usr/prico:/bin/sh
```

Seguidamente se reproducirá un ejemplo del contenido del archivo /etc/group, cuya estructura está formada por cuatro campos: nombre del grupo, contraseña (vacía), identificador de grupo (GID), y la lista separada por comas de nombres de los usuarios que constituyen el grupo.

Root::0:root
other::1:root,daemon,juan
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root, adm,daemon,listen
uucp::5uucp, nuucp
mail::7:root
asg::8:asg
network::10:network
sysinfo::10:network
sysinfo::11:sysinfo,dos
daemon::12:root, daemon
terminal::15:
cron::16:cron
audit::17:root, audit
lp::18:lp
backup::19:
mem::20:
sysadmin::23:
group::50:ingres, ejemplo, juan, jgomez, prico
desarrollo::51:juan

En UNIX C1 existen dos tipos de cuentas:

- Superusuario ("root") que posee todos los privilegios del sistema. Es decir, sobre esta cuenta no se aplican los mecanismos de control de accesos discrecional. Cualquier usuario cuyo identificador sea cero, es superusuario.

- Resto de usuarios, que pueden realizar las acciones que les permitan las protecciones de los ficheros sobre los que operan.

Un usuario puede desarrollar una sesión de trabajo en dos modalidades:

- Interactiva: conectándose a su cuenta a través de una terminal, y desarrollando la sesión de trabajo correspondiente.
- Por lotes: solicitando al comando at la ejecución temporizada de un conjunto de programas.

3.3 Tipos de archivos.

En UNIX hay tres tipos de archivos:

- Planos
- Directorios
- Especiales

3.3.1 Archivos Planos

Son los que contienen información generada durante la sesión de trabajo de los usuarios: documentos, programas, datos, ejecutables, etc. Desde el punto de vista del sistema operativo un archivo plano, a partir de ahora, archivo, de tamaño N es un conjunto consecutivo de octetos sin estructura física

alguna, tal y como se muestra en la Figura 3.4. Son los programas que actúan sobre los ficheros quienes interpretan una estructura lógica sobre su contenido.

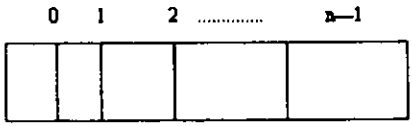


Figura 3.4 Estructura de un fichero plano

3.3.2 Archivos Directorios

Los directorios, son catálogos de archivos que manipula el sistema operativo para su organización en una jerarquía arborescente. Por cada archivo que haya en un directorio, existirá en éste una entrada que contendrá su nombre y un número índice que permitirá acceder a su estructura de control asociada, que en UNIX se denomina nodo índice, tal y como muestra el siguiente ejemplo:

```
$ ls -a          visualizar los nombres del archivo del directorio actual
.
..
f1
f2
$ ls - ia        visualizar el contenido del archivo directorio actual
5558
5289
 329 f1
5604 f2
```

```
$ rm f1          borra tanto el contenido del archivo f1 como su referencia en el directorio
$ ls -ia
5558
5289
5604
```

3.3.3 Archivos Especiales

Cada dispositivo de la computadora (memoria, disco, terminal, disquete, ratón, etc.) está representado por un archivo especial, a partir de ahora dispositivo.

Existen dos tipos de dispositivos: orientados a carácter y a bloque. Los primeros realizan sus operaciones de entrada/salida carácter a carácter: terminal, impresora, ratón.

Por el contrario, los segundos realizan sus operaciones en bloques de caracteres, y son fundamentalmente discos y disquetes.

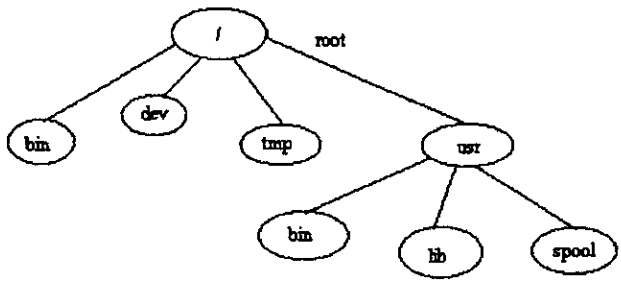
```
$ ls /dev/hd*   visualizar los discos duros del ordenador
hd00
hd01
hd02
```

3.4 Sistema de Archivos.

El sistema de archivos es la estructura lógica que utiliza el sistema operativo para manejar al conjunto de archivos que hay en un disco, o en parte de él, y que desde el punto de vista del usuario se observa como un árbol, cuya raíz está representada por el carácter barra inclinada ("/").

Todos los archivos están ubicados dentro de algún sistema de archivos. El camino de un archivo es la ruta de nombres que debe recorrerse desde la raíz hasta el lugar en el que está ubicado el archivo. Como este camino suele ser muy largo, pueden aplicarse dos sinónimos para abreviar su escritura: el carácter punto (".") representa al camino del directorio actual, mientras que la secuencia punto punto ("..") corresponde al camino del directorio padre del actual.

Todo computadora bajo UNIX posee un sistema de archivos, que reconoce al arrancar la computadora, denominado sistema de archivos raíz donde están ubicados los archivos que utiliza el propio sistema operativo para su correcto funcionamiento, y cuya distribución se muestra en la Figura 3.5.



/bin(ary)	comandos de usuario.
/dev(ice)	dispositivos.
/lib(rary)	bibliotecas para el compilador de C.
/tmp	archivos temporales
/usr	información relacionada con aplicaciones de usuario
/usr/bin	más comandos de usuario.
usr/lib	bibliotecas de comandos.
usr/spool	colas para impresoras

Figura 2.5 Directorios del sistema UNIX

Puesto que cada disco o partición del mismo es un sistema de archivos independiente, ¿cómo puede accederse a la información de varios discos? El sistema de archivos raíz siempre está accesible, y corresponde a un disco o partición del mismo. Para acceder a los demás, lo único que hace el administrador, es montar los otros sistemas de archivos en directorios vacios del sistema de archivos raíz, tal y como se muestra en la Figura 3.6. Una vez montados, el usuario no es consciente de cuál es el disco sobre el que trabaja, que incluso puede estar ubicado en una computadora remota.

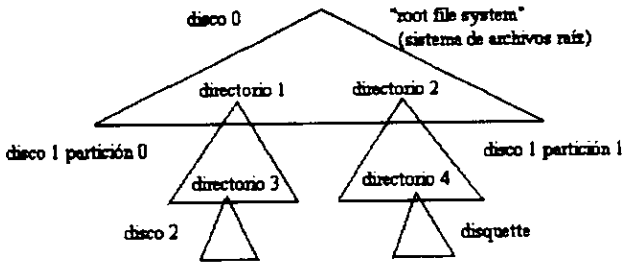


Figura 2.6 Montajes de los sistemas de archivos.

Internamente, un sistema de archivos se ve como un conjunto consecutivo de N bloques de disco. En el bloque 0, se ubica el programa para cargar al sistema operativo durante el arranque de la computadora ("boot"). El bloque 1 es el superbloque, que contiene información relativa al sistema de archivos: tipo, tamaño, espacio libre, etc. Del bloque 2 al k se encuentra el área de nodos índice, y cada uno de éstos es un registro con información administrativa relativa a un archivo. Entre el bloque k+1 y el N-1 están los bloques correspondientes al contenido de los archivos. En la Figura 3.7 se muestra esta distribución.

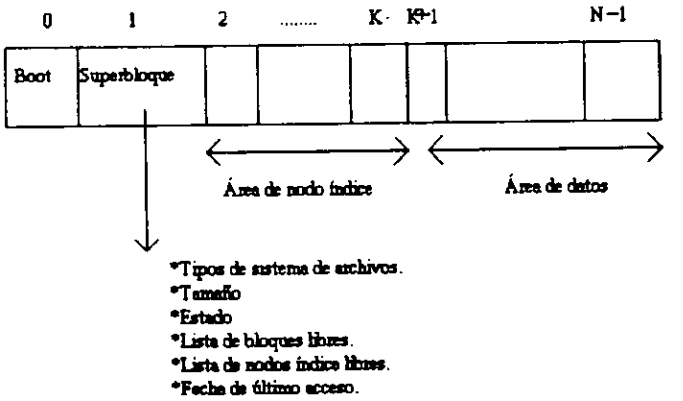


Figura 3.7 Formato interno de un sistema de archivos.

Cada nodo índice ("i-node") es un registro de 64 octetos, que contiene la siguiente información sobre un fichero:

-Identificador de usuario (UID) del creador (dueño) del archivo.

-Identificador de grupo (GID) al que pertenece el archivo. Corresponde al grupo en el que se encontraba el dueño cuando creo el archivo.

-Máscara de permisos (protecciones) del archivo.

-Tipo de archivo: plano, directorio, dispositivo.

-Tamaño en octetos del archivo.

-Número de sinónimos del archivo. En UNIX, un archivo puede tener varios nombres. Estos se encuentran en sus directorios respectivos, pero todos ellos apuntan al mismo nodo índice.

-Fecha de creación del archivo.

-Fecha de último acceso (lectura o escritura) al archivo.

-Fecha de última modificación del nodo índice.

-Tabla de apuntadores a los bloques de disco que constituyen el contenido del archivo. No es aplicable sobre dispositivos.

3.5 Permisos de los archivos

En este apartado se describe qué es el modo de un archivo, cómo se interpreta, establece por defecto, y modifica.

3.5.1 Modo de un archivo

En UNIX existen tres tipos de accesos a los archivos:

-Lectura ("read"). Permite visualizar el contenido de un archivo.

-Escritura ("write"). Permite modificar o borrar el contenido de un archivo.

-Ejecución ("execute"). Permite activar un ejecutable (binario o archivo de comandos). Aplicado sobre un directorio permite utilizar su nombre como parte del camino de un archivo.

Para cada archivo, existen los accesos anteriores a tres niveles:

-Dueño del archivo ("user"). Este nivel se aplica sobre el usuario que ha creado el archivo.

-Grupo del archivo ("group"). Este nivel corresponde al grupo al que pertenecía el dueño cuando creó el archivo.

-Otros ("others"). Se asigna a aquellos usuarios que no encajan en los dos anteriores.

En consecuencia, los permisos de acceso a un archivo (en adelante modo de un archivo) UNIX son una máscara formada por nueve componentes, generados a partir de los tres tipos de acceso (lectura, escritura, ejecución), y los tres niveles (dueño, grupo, otros):

r w x r w x r w x

Tercer terceto: corresponde al nivel de otros.

Segundo terceto: corresponde al nivel de grupo

Primer terceto: corresponde sl nivel de dueño.

Cuando no existe un permiso, aparece un guión en su nivel correspondiente:

r w - r w - r w -

El modo de un fichero puede representarse como una máscara binaria. Si existe el permiso, uno; sino, cero:

r w - r w - r w - r w x r w x r w x

1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1

A su vez, cada terceto binario puede representarse como el dígito octal correspondiente:

r w - r w - r w - r w x r w x r w x

```
110 110 110     111 111 111
 6   6   6       7   7   7
```

Por tanto, los números octales, que notacionalmente comienzan en cero por convenio, 0666 y 0777 corresponden a los modos anteriormente expresados.

3.5.2 Modo por defecto de un archivo.

El sistema operativo UNIX crea los archivos con un modo por defecto, que es, 0666 para los archivos planos y dispositivos, y 0777 para los directorios. Como puede observarse, estas máscaras son muy peligrosas para los archivos, puesto que a nivel de grupo y otros existe permiso de lectura y escritura, lo cuál no garantiza ni la confidencialidad ni la integridad de la información a ambos niveles. Estos modos por defecto no pueden cambiarse. Ahora bien, existe un comando de usuario, umask (máscara de usuario), que especifica en octal cuáles son los permisos que desean retirarse al modo por defecto. El sistema operativo siempre aplica la máscara de usuario cuando crea un archivo nuevo. El administrador del sistema tiene que establecer el valor de umask para los usuarios, y si no lo hace, cada usuario debe poner el valor que considera oportuno en su archivo de inicialización (.profile, .login), que se ejecuta siempre que se conecta a la computadora.

3.5.3 Comandos para la manipulación del modo de un archivo.

El orden en el que UNIX aplica los permisos cuando un sujeto intenta acceder a un archivo es :

1. Si el sujeto es el dueño del archivo, se aplica el terceto de dueño, y no se comparan más permisos. Si no, se pasa a la etapa 2.

2. Si el sujeto pertenece al grupo del archivo, se aplica el terceto del grupo, y no se comparan más permisos. Si no, se pasa a la etapa 3.
3. Se aplica el terceto de otros.

Esta manera de aplicar los permisos puede crear algunas situaciones paradójicas. Por ejemplo, si el dueño de un archivo ha retirado el permiso de lectura en su nivel, pero lo asigna en el de grupo y otros, todos los usuarios de la computadora podrán acceder a su archivo, salvo él mismo. Nada impide que el dueño se asigne el permiso de lectura para acceder a su archivo.

Sólo el dueño del archivo y el superusuario pueden cambiar sus permisos mediante los comandos:

-chmod: cambio de los permisos

-chown: cambio del dueño de un fichero
-chgrp: cambio del grupo dueño

El siguiente ejemplo muestra cómo funcionan los permisos sobre los archivos planos:

```
$ touch f1                          crear el archivo f1 vacío
$ ls -l f1                      ver sus permisos
-rw------- 1 juan                other          0 Aug 31  14:43 f1
$ cat f1                            se accede al archivo
$ chmod u-r f1                      retirado el permiso de lectura
$ ls -l f1
--w------- 1 juan                other          0 Aug 31  14:43 f1
$ cat f1
cat: cannot open f1                 no se puede leer el archivo
$ chmod u-w f1
$ ls -l f1
```

——— 1 juan other 0 Aug 31 14:43 f1
$ rm f1 no debería poder borrarse
f1: 0 mode? Y
$ ls se ha borrado por la confirmación
del dueño
$

El siguiente ejemplo muestra cómo funcionan los permisos sobre directorios:

$ mkdir d1 crear el directorio d1
$ ls -l
drwx—— 2 juan other 32 Aug 31 15:13 d1
$ touch d1/d1f1 crear el archivo d1f1 en el directorio
d1
$ ls -l d1
-rw—— 1 juan other 0 Aug 31 15:13 d1f1
$ chmod u-r d1 retirar el permiso de lectura sobre el
directorio d1
$ ls -l
d-wx—— 2 juan other 48 Aug 31 15: 15 d1
$ ls d1 no puede visualizarse el contenido del
archivo directorio
can not access directory d1
$ chmod u+r, u-x d1
$ ls -l se ha retirado el permiso de búsqueda
drw—— 2 juan other 48 Aug 31 15:15 d1
$ ls -l d1
d1/d1f1 not found no puede utilizarse el nombre d1 como
parte de un camino
total 0
$ cd d1

```
ksh: c1: permission denied
$ cat d1/d1f1
cat: cannot open d1/d1f1
$ chmod u+x d1
$ cat d1/d1f1
$ chmod u-w d1                    retirado el permiso de escritura
$ ls -l
dr-x------ 2 juan                 other          48 Aug 31  15:15 d1
$ touch d1/d1f2                   no pueden crearse archivos
touch: d1/d1f2 cannot create
$ rm d1/d1f1                      tampoco pueden borrarse archivos
rm: d1/d1f1 not removed.
Permission denied
$ chmod u+w d1
$ rm d1/d1f1
$ ls -l d1
```

El siguiente ejemplo muestra cómo cede el dueño de un archivo la propiedad del mismo, y en consecuencia sus derechos de acceso:

```
$ touch f1
$ ls -l
-rw------- 1 juan                 other          0 Aug 31  15:49 f1
$ chown root f1                   ahora root es el nuevo dueño del
        archivo
$ ls -l
-rw------- 1 root                 other          0 Aug 31  15:49 f1
$ chmod +rw f1                          se han perdido los derechos de acceso
chmod: WARNING: can't change f1                 no pueden cambiarse los
                                  permisos
$ ls -l
```

-rw——— 1 root other 0 Aug 31 15:49 f1

$ rm f1 podemos borrarlo porque es un archivo

que está en nuestra cuenta

f1: 600 mode ? y

$ ls

$

El siguiente ejemplo muestra cómo el dueño de un fichero puede cambiar su grupo:

$ id el usuario averigua su propia identidad

uid=201 (juan) gid=1 (other) el usuario juan pertenece por defecto

al grupo other

$ touch f1

$ ls -l

-rw——— 1 juan other 0 Aug 31 16:13 f1

$ newgrp desarrollo el usuario juan cambia al grupo

desarrollo

$ id

uid=201 (juan) gid=51 (desarrollo) ya se ha producido el cambio de

grupo

$ touch f2 el archivo f2 se crea bajo el grupo

desarrollo

$ ls -l

-rw——— 1 juan other 0 Aug 31 16:13 f1

-rw——— 1 juan desarroll 0 Aug 31 16:15 f2

3.5.4 Protección de la información

Supongamos que se plantea el siguiente problema: en la cuenta del usuario jgomez (UID: 202, GID: 0) existe un archivo llamado datos, que contiene una cierta información a la que jgomez desea puedan acceder, de una forma que él controle, todos los usuarios del sistema. Los permisos del archivo datos deberían ser:

```
$ ls -l datos
-rw- rw- rw-  1 jgomez          group   6300  Aug 31  13:00 datos
```

Como hay permiso de lectura a nivel de otros, cualquier usuario puede copiar el archivo datos en su cuenta, con lo cual jgomez ya no tiene control sobre la información. Además como hay permiso de escritura a nivel de otros, cualquier usuario puede destruir el archivo, o lo que es peor, puede modificarlo ligeramente de manera favorable a sus intereses personales, sin que otros usuarios se percaten de ello obligatoriamente. Para evitar que ambas situaciones se produzcan, jgomez tendría que retirar todos los permisos a nivel de otros, e incluso a nivel de grupo. Por tanto, el modo del archivo datos quedaría como:

```
$ ls -l datos
-rw- — —  1 jgomez          group   6300  Aug  31  13:00 datos
```

Ahora no hay problemas de confidencialidad, ni de integridad. Pero es esta situación, tan sólo el usuario jgomez puede acceder al archivo datos, lo cuál incumple uno de los requisitos iniciales del problema. El usuario jgomez decide resolver este problema construyendo el programa control_datos para acceder al fichero datos:

```
$ ls -l control_datos datos
```

```
-rwx --x --x  1  jgomez        group   2600  Aug 31  13:00
                                control_datos
-rw- --- --- y  jgomez         group   6300  Aug 31  13:00 datos
```

Puesto que el programa control_datos tiene permiso de ejecución a todos los niveles, cualquier usuario puede ejecutarlo, y en consecuencia acceder al archivo datos. ¡Incorrecto! ¿Qué sucede cuando el usuario juan (UID:201, GID:1) ejecuta el programa control_datos? Cuando se ejecuta un programa el sistema operativo almacena en el identificador de usuario real ("Real User Identificator. RUID") el UID del usuario llamador que, en este caso, es el usuario juan, luego el RUID es 201.

Cuando el programa control_datos ejecutado por juan accede al archivo datos, el sistema operativo aplica el RUID para determinar la concesión del acceso. ¿Es el usuario 201 dueño del archivo datos? No. ¿Pertenece al grupo? Tampoco, luego se aplica el nivel de otros, donde no hay permiso alguno. En consecuencia, sólo el usuario jgomez (y el superusuario) puede acceder al archivo datos.

Todos los problemas anteriores se resolverían si durante la ejecución del programa control_datos, sea quien fuere el usuario que lo ejecuta, su identidad se convirtiera en la del dueño del ejecutable, jgomez. Esto se consigue en UNIX asignando a los ejecutables binarios el permiso s ("Set UID: SUID, Set GID: SGID") a nivel de dueño (grupo), que permite reasignar el identificador del usuario (grupo) llamador al del dueño (grupo) del ejecutable, aunque tan sólo durante la ejecución de ese programa.

```
$ chmod u+s control_datos
$ls -l control_datos datos
-rws --x --x  1 jgomez     group   26000  Aug 31  13:00 control_datos
-rw- --- ---  1 jgomez     group    6300  Aug 31  13:00 datos
```

Ahora, cuando el usuario juan ejecuta el programa control_datos, su RUID sigue siendo 201. Pero el sistema operativo almacena otro valor, que es el identificador de usuario efectivo ("Effective User Identificator, EUID"). Cuando el ejecutable no tenía el atributo s, el EUID valía 201, pero ahora corresponde al del dueño del ejecutable, 202. Cuando se solicita un acceso a un archivo, el sistema operativo utiliza en

primer lugar el identificador efectivo, y luego el real. En este caso, como el efectivo es 202, que corresponde a jgomez, y éste es el dueño del archivo datos, el programa control_datos no tiene problema alguno para manipular la información bajo su control. Con esta solución se cumple con el requisito inicial planteado: garantizar el acceso controlado a la información.

Puede razonarse análogamente sobre el atributo s aplicado al nivel de grupo (SGID), respecto a los identificadores de grupo real y efectivo que el sistema operativo mantiene para cada programa que se ejecuta.

Los ejecutables con el atributo s pueden convertirse en un agujero para la seguridad, y es necesario vigilarlos estrictamente.

3.6 Comandos

Son los programas que vienen con el núcleo del sistema operativo UNIX. Hay unos 200 para usuarios y otros tantos para el administrador de red. Los de usuario están ubicados en los directorios del sistema /bin ("binary") y /usr/bin. Mientras que los del administrador están fundamentalmente en /etc y /usr/lib.

Cuando una persona entra en su cuenta aparece en la pantalla el símbolo dólar ($), que es el mensaje que indica al usuario que el intérprete de comandos puede ejecutar el programa, normalmente un comando, que solicite. ¿Cómo sabe el intérprete dónde se encuentra el programa por ejecutar? En su entorno posee la variable PATH que le indica los directorios en los que debe buscar el programa que se quiere ejecutar.

```
$ echo $PATH                          ver el valor de la variable PATH
/bin: /usr/bin: /usr/jgomez/bin:.  El carácter : separa directorios
```

La sintaxis de los comandos de UNIX es:

```
$ nombre  -opciones  argumento...
```

donde el nombre es una cadena alfabética que representa el nombre del comando. UNIX distingue entre mayúsculas y minúsculas, y generalmente el nombre del comando debe escribirse en minúsculas. Las opciones del comando se representan mediante letras precedidas por el símbolo guión. Por último, se escriben los argumentos, que suelen ser caminos (nombres) de archivos sobre los que pueden aplicarse los caracteres comodines.

-Asterisco(*), equivalente a cualquier cadena de caracteres, incluida la vacía, que no empiece por el carácter punto(.).

-Interrogación (?), equivalente a cualquier carácter, salvo el punto (.).

-Corchetes ([abcd), que es equivalente a un carácter cualquiera de los que hay en su interior.

UNIX posee una documentación interactiva para aprender a usar sus comandos que se activa mediante la orden:

$ man nombre_de_comando_sobre_el_que_se_solicita_ayuda

En realidad, la documentación accesible desde man está dividida en ocho secciones:

1. Comandos de usuario.

   $ man passwd información sobre la asignación de contraseñas

2. Llamadas del sistema operativo

   $ man getuid como obtener el UID de un usuario

3. Funciones de biblioteca del lenguaje C.

$ man getpwent como obtener la información de una cuenta

4. Formato de archivos del sistema operativo

$ man a.out cuál es el formato de un ejecutable binario

5. Otros formatos de fichero

6. Juegos

7. Información sobre dispositivos

$ man fd información sobre disquetes

8. Procedimientos de mantenimiento

$ man boot información sobre el arranque de la computadora.

Como puede observarse el comando de usuario man no sólo proporciona información sobre otros comandos, sino sobre cualquier aspecto de UNIX: llamadas al sistema operativo, bibliotecas, dispositivos, mantenimiento. De hecho, sólo el uso continuado de la documentación interactiva permite a los usuarios aprender lo que necesitan, por lo que se recomienda su uso intensivo.

3.7 Ejecutable binario, imagen y proceso.

En este apartado se explican los conceptos siguientes:

- Cuáles son los tipos de archivos ejecutables.

- Cómo se carga un ejecutable en memoria para convertirse en una imagen.
- Cómo se convierte una imagen en un proceso al recibir UCP.

3.7.1 Ejecutable binario

Un ejecutable es un programa que puede procesarse en la computadora, En UNIX hay dos tipos de ejecutables: archivos de comandos y ejecutables binarios. Los archivos de comandos o procedimientos "shell" son archivos que contienen comandos que ejecuta el intérprete de comandos. Los ejecutables binarios son archivos planos generados por la acción combinada de un compilador y un montador que contienen código máquina ejecutable directamente por el procesador de la computadora.

En UNIX, a los archivos ejecutables binarios se les denomina a.out, y están compuestos por cinco áreas:

-Cabecera. Aquí se encuentran los tamaños del resto de las áreas del ejecutable.

-Código. Donde está el código binario del programa. No hay datos, pero sí referencia a los mismos.

-Datos. Aquí se colocan las variables cuyo valor es conocido por el compilador en tiempo de compilación: constantes, cadenas de caracteres, etc.

-Tabla de símbolos. Se almacenan las referencias de todos los símbolos que utiliza el programa. Esta información la utilizan los programas depuradores para informar al usuario sobre el estado de las variables del programa.

-Información de reubicación. Contiene información sobre donde ha buscado el montador las referencias que no encontró el compilador.

Las dos últimas secciones no son imprescindibles para el ejecutable si ya está depurado, y pueden eliminarse mediante el comando de usuario strip. Por otro lado, si desea averiguarse el tamaño de las áreas de código y datos puede utilizarse el comando de usuario size.

La figura 3.8 muestra la estructura completa de un ejecutable binario.

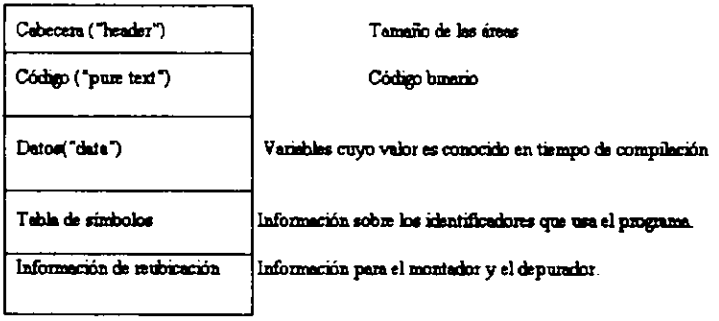


Figura 2.8 Formato de un archivo ejecutable binario (a.out)

3.7.2 Imagen

Cuando un usuario solicita ejecutar un archivo binario, el sistema operativo carga en un espacio virtual de direcciones independientes las áreas de código y datos del ejecutable. A continuación añade tres áreas: una para las variables cuyo valor no ha sido inicializado en tiempo de compilación ("bss"), otra para las variables dinámicas creadas en tiempo de ejecución ("heap"), y una tercera para gestionar las llamadas y retornos de las funciones que invoca el programa (pila, "stack"), tal y como se muestra en la figura 3.9.

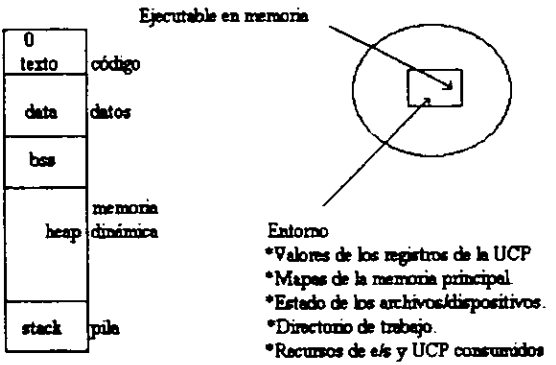


Figura 3.9 Estructura de un proceso

Pero el ejecutable cargado en memoria de esta manera no es suficiente. El sistema operativo tiene que multiplexar la UCP entre todos los programas que se ejecutan en la computadora, para lo que necesita añadir una información que le permita asignar y retirar la UCP a un programa sin que éste se percate de ello. Para ello, el núcleo del sistema operativo inserta al ejecutable en un entorno que contiene esta información: valor de los registros de la UCP, directorio de trabajo, estado de las archivos abiertos, recursos consumidos, etc. A la combinación de ejecutable en memoria más entorno se le denomina imagen, pues contiene toda la información necesaria para ejecutarse creyendo poseer todos los recursos de la computadora.

3.7.3 Proceso

Cuando una imagen recibe UCP se le denomina proceso. Sin embargo, como en un intervalo de tiempo todas las imágenes van a ejecutarse, por abuso del lenguaje, se asemeja el concepto de imagen al de proceso. El sistema operativo identifica cada proceso mediante un número entero único denominado

identificador de proceso ("Process Identificator, PID). Un proceso es interactivo cuando se ejecuta en una terminal, de lo contrario es no interactivo.

Evolución de un proceso.

Pero ¿cómo se crea un proceso? Se verá a través del ejemplo mostrado en la figura 3.10. Cuando un usuario está en su cuenta le atiende el intérprete de comandos, que es un proceso con un cierto identificador que puede averiguarse mediante el comando de usuario ps -f, que para el ejemplo se supondrá que es 100. Cuando el usuario solicita ejecutar un comando, como por ejemplo who, para ver cuáles son los otros usuarios conectados al sistema, es necesario que nazca un nuevo proceso que soporte la ejecución de dicho programa.

Cuando el intérprete de comandos (proceso padre-100) recibe la cadena "who", ejecuta la llamada al sistema operativo fork() para crear un proceso hijo-101 que es una copia idéntica a él mismo en lo que se refiere al ejecutable y una parte del entorno. Obsérvese que el PPID ("Parent PID") del proceso hijo-101 es el PID del proceso padre-100. Así pues, hay dos procesos distintos, 100 y 101, que ejecutan lo mismo, el intérprete de comandos. A continuación el proceso padre se queda esperando (wait()) hasta que su hijo termine de ejecutarse. Mientras tanto el proceso hijo ejecuta la llamada exec() para cambiarse a sí mismo su código ejecutable, con lo que sustituye el código del intérprete de comandos por el ejecutable binario del comando de usuario who, que corresponde al archivo /bin/who. A partir de este momento, el proceso hijo-101 está ejecutando el comando de usuario who, y no el intérprete como había sucedido hasta ahora. Cuando el proceso hijo concluye su programa, ejecuta la llamada exit() que envía una señal (signal()) al proceso padre, la cuál le despierta, y permite continuar su ejecución mostrando el símbolo dólar de nuevo en la pantalla de la terminal.

$ who

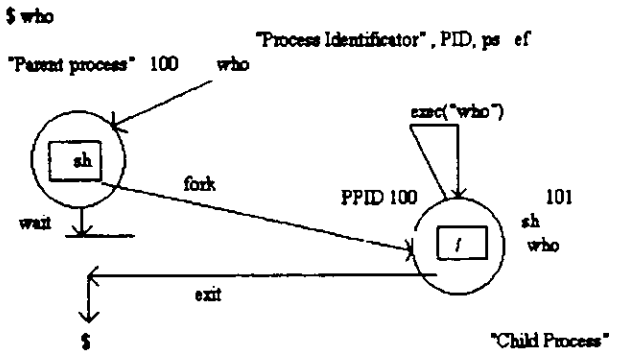


Figura 3.10 Mecanismos para la creación de un proceso

Mediante el comando ps -f puede comprobarse la jerarquía de los procesos.

ps -f

UID	PID	PPID	TTY	COMMAND
root	285	1	01	-ksh
root	797	285	01	ps -f

Generación de los procesos del sistema.

Mediante la descripción anterior queda explicada la genealogía de procesos en UNIX. Tan sólo falta un pequeño detalle: ¿cómo ha nacido el intérprete de comandos del cuál se ha partido? La explicación está en la

misma figura adjunta. Cuando arranca la computadora, se ejecuta un programa ("boot") cuya misión es cargar el núcleo del sistema operativo en memoria. El núcleo es el primer proceso de la computadora, y es padre de si mismo, puesto que su PID y PPID es cero. La misión del núcleo es controlar los recursos de la computadora, por lo que delega el nacimiento de nuevos procesos en uno de sus hijos, init. Para cada una de las terminales dadas de alta en el sistema init crea un hijo en el que se ejecuta el programa getty que solicita el nombre de conexión. Cuando éste se obtiene, el mismo proceso se transforma en el programa login que solicita la contraseña. Si ambas informaciones son correctas, este proceso se convierte en el intérprete de comandos.

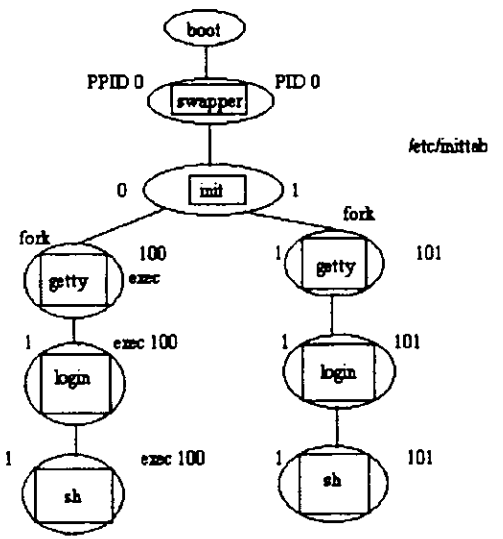


Figura 3.11 Ejemplo sobre la creación de procesos

Mediante el comando ps -ef puede comprobarse la genealogía de los procesos del sistema:

ps -ef

UID		PID	PPID	C	STIME	TTY	TIME	COMMAND	
root		0	0	0	Aug 24	?	0:00	swapper	
root		1	0	0	Aug 24	?	0:26	/etc/init	
root		285	1	3	12:22:29		01	0:04	-ksh
juan		375	1	0	12:43:57		02	0:09	/bin/ksh
prico	394	1	0	12:47:06		03	0:01	-sh	

CAPITULO 4

Mecanismos de seguridad

4.1 Criptografía.
4.2 Criptosistemas.
4.3 Tipo de criptosistemas.
4.4 Ataques.
4.5 Criptosistemas de clave secreta
4.6 Criptosistemas de clave pública.
4.7 Firma digital.
4.8 Criptosistemas irreversibles.
4.9 Técnicas criptográficas básicas.

Las amenazas que sufre la información durante su proceso, almacenamiento y transmisión son crecientes,multiformes y complejas. Para contrarrestarlas se han desarrollado numerosas medidas de protección, que se implementan en el equipo físico o lógico mediante los denominados mecanismos de seguridad.

La lista de estos mecanismos es ya muy numerosa y en ella encontramos, entre otros : identificación y autenticación de usuarios, control de acceso, control de flujo de información, registros de auditoría, cifrado de información, etc. De éstos, el mecanismo por excelencia es el cifrado de la información, cuya importancia en la actualidad es creciente, como lo demuestra el elevado número de investigadores que hoy en día están consagrados al desarrollo e implementación de algoritmos y protocolos de cifrado de la información, intercambio de claves, generación de éstas, etc.

4.1 Criptografía

La Criptografía (etimológicamente escritura oculta) es una ciencia antiquísima, cuyos primeros pasos hay que buscarlos en albores de nuestra civilización. Se puede definir como la disciplina que estudia los principios, métodos y medios de ocultar la información, intercambio de claves, generación de éstas, etc.

Tradicionalmente, esta ciencia se ha asociado con la protección de la confidencialidad de informaciones de interés militar o político. Sin embargo, hoy en día su interés ha desbordado ampliamente dicho campo, para merecer una atención preferente por todos los sectores públicos o privados para los que la información es un valioso activo. Por otro lado, si en la antigüedad la Criptografía se usaba para el mantenimiento de la confidencialidad de la información, actualmente se emplea, más si cabe, para preservar la integridad, característica de la información más expuesta, sobre todo en instrucciones privadas, que la confidencialidad.

La historia de la criptografía puede dividirse en tres periodos. Hasta fines de la década era más un arte que una ciencia y, aún hoy en día, así la define nuestra Real Academia. Aunque algunos algoritmos de cifrado tenían una consistente formulación algebraica, la criptografía carecía globalmente de una sólida base matemática. Sin embargo la publicación en 1949 de un artículo de Shannon sobre esta materia , cimentó la criptografía sobre sólidos principios matemáticos. Por ello, el periodo transcurrido entre la antigüedad clásica y el año citado se denomina se denomina de la criptografía precientífica, llamándose período de la criptografía científica al que comienza en 1949. No hubo de esperar ni treinta años para que un nuevo artículo iniciase otra época en la criptografía. En efecto, en 1976 Diffie y Hellman publicaron sus trabajos sobre los criptosistemas de clave pública, dando lugar al período, que se extiende hasta nuestros días, conocido como período de la criptografía de clave pública.

4.2 Criptosistemas.

Tal y como se muestra en la figura 4.1, un sistema de cifrado está constituido por un emisor, que genera un mensaje denominado texto en claro, un dispositivo cifrador , que transforma el texto claro en un mensaje ininteligible denominado texto cifrado o criptograma, un canal (de almacenamiento o transmisión), un dispositivo descifrador , cuya función es la inversa del cifrador, y un receptor.

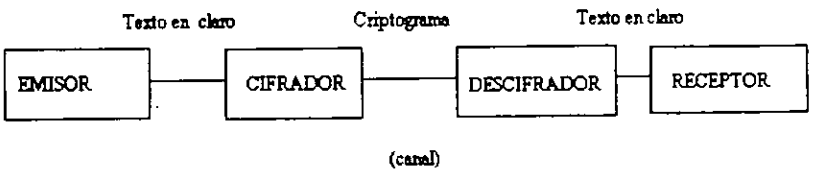


Figura 4.1 Estructura de un sistema de cifrado

Un intruso que interceptase el canal obtendría un criptograma que, es de esperar, sería incapaz de interpretar, manteniéndose así la confidencialidad de la información. Al tiempo, no siendo capaz de traducir el criptograma, tampoco sería capaz de alterarlo subrepticiamente a su favor, con lo que se protegería también la integridad de la información.

Evidentemente, el elemento fundamental de un criptosistema cifrador. Está compuesto por un dispositivo físico, o por un programa, que implementa el algoritmo de cifrado , que suele ser el mismo en el caso de cifrado que en el descifrado. La función es computacionalmente irreversible; es decir, es imposible de invertir con la actual potencia del cálculo, a no ser que se posea una información adicional: la clave de descifrado. Así cualquier conocedor de esta última podrá, a partir del criptograma , obtener el texto en claro, siempre que el algoritmo, como suele ser habitual, sea conocido por todo el mundo. Sin dicho conocimiento, ni aun sabiendo la función matemática, se podría descifrar el criptograma.

En el estudio de los criptosistemas se distinguen cinco conceptos básicos:

1. El espacio de mensajes M= {m1,m2,...}, constituido por todos los posibles textos en claro que un emisor puede generar. Los textos en claro que un emisor puede generar. Los textos en claro se forman a partir de un alfabeto mediante ciertas reglas sintácticas y semánticas.
2. el espacio de cifrados C= {c1,c2,...}, formado por todos los posibles criptogramas que el cifrador puede producir. El alfabeto de los cifrados puede ser igual al de los mensajes o distinto.
3. El espacio de claves K= {k1,k2...}, integrado por todas las posibles claves, tanto de cifrado como de descifrado, de algoritmo cifrador.
4. Una familia de transformaciones de cifrado Ek : M- C, con k e K, la clave de cifrado. Habitualmente, el proceso realizado en el cifrador se representa así:
5. nxxnxnh hhhjjkjxk

$$E(k,m)=c$$

con k e K, m e M, c e C.

6. Una familia de transformaciones de descifrado Dk : C- M, con k e K, la clave de descifrado. Esta puede ser igual a la de cifrado o diferente. Normalmente, la operación se produce que se produce en el descifrador se expresa como:

$D(k,c)=m$

donde $k \in K$, $c \in C$, $m \in M$.

4.3 Tipo de criptosistemas.

Hasta la década de los setenta todos los criptosistemas conocidos funcionaban con una clave de cifrado igual a la de descifrado o, si eran diferentes, una podía obtenerse de la otra en un tiempo y con recursos razonables. L a invulnerabilidad de tales sistemas dependía, por tanto, del mantenimiento en secreto de la clave. Consiguientemente, debía transferirse del emisor al receptor a través de un canal seguro, obviamente diferente del canal de criptosistemas, que por hipótesis está amenazado por posibles interceptores.

Estos criptosistemas se denominan de clave secreta, de clave única y también simétricos.
Sin embargo, en 1976 Diffie y Hellman demostraron la posibilidad de construir criptosistemas que no precisaban transferir una clave secreta entre el emisor y el receptor, evitando así los problemas inherentes a la búsqueda de canales seguros para tal transferencia, previamente al establecimiento de una transmisión cifrada.

En estos criptosistemas la clave de cifrado, denominada clave pública, se hace de general conocimiento. Sin embargo, la clave descifrado, denominada clave privada, se mantiene en secreto. Obviamente ambas claves no son independientes, pero del conocimiento de la pública no se infiere la privada, a no ser que se tenga algún dato adicional que también habrá de mantenerse en secreto o, mejor destruirse una vez el par clave pública-clave privada.

Cuando un usuario desea recibir informaciones cifradas, da a conocer a todos los potenciales remitentes su clave pública. Así estos cifrarán los mensajes destinados al citado usuario con la misma clave, la pública del destinatario. Sin embargo, no conociendo nadie más la clave privada, ni pudiéndola obtener a partir de la pública, sólo el destinatario podrá descifrar la información a él remitida.

Estos criptosistemas se denominan de clave pública o asimétricos, pues del conocimiento de una clave no se deduce la otra.

Todavía se puede considerar un último tipo de sistemas de cifrado denominados criptosistemas irreversibles, basados en algoritmos no invertibles. En estos criptosistemas, dado el criptograma no es posible obtener el mensaje en claro. Figuradamente, se puede decir que sólo trabajan en un sentido, el que lleva del mensaje en claro al texto cifrado, pero no el contrario, que conduce de este último a aquél.

Estos cifrados se emplean en aplicaciones que no requieren descifrar los criptogramas . Una de estas aplicaciones consiste en la determinación de si un cierto mensaje se corresponde con un determinado texto cifrado almacenado en un sistema. Para ello el algoritmo de cifrado produce un criptograma, que se compara con el guardado en la memoria. Para esta aplicación se requiere, además de que el algoritmo sea o no invertible, que sea unívoco , es decir , que dos mensajes distintos produzcan dos cifrados diferentes. Un ejemplo de este tipo de aplicaciones se tiene en al almacenamiento de las contraseñas de usuarios de un sistema. El mantenimiento de memoria de las mismas sin cifrar representa un grave riesgo, pues alguien puede averiguarlas y suplantar a los legítimos usuarios . Almacenarlas cifradas mediante criptosistemas de los tipos vistos hasta ahora también es inseguro, pues la clave de descifrado puede ser conocida por el administrador del sistema (y posiblemente por más profesionales informáticos que trabajen sobre el ordenador), quien con el conocimiento del algoritmo de cifrado y de las contraseñas criptografiadas es capaz de averiguar las contraseñas en claro.

Por lo anterior, para el almacenamiento de contraseñas se emplean algoritmos no invertibles . cada vez que el usuario introduce su contraseña, se cifra y compara con la contraseña cifrada almacenada en la tabla correspondiente. El acceso se permite sólo si coinciden ambas . esto implica que una contraseña olvidada no se puede recuperar, debiéndose generar otra que será nuevamente almacenada como un criptograma.

4.4 Ataques.

La ciencia que se ocupa del estudio sistemático de los métodos de descifrar informaciones criptografiadas se denominan criptoanálisis y criptoanalistas sus practicantes . Por tanto se puede decir, que es la ciencia opuesta a la criptografía.

En el estudio de los métodos de defensa frente al criptoanalista se supone que éste tiene acceso a todo el criptograma. También es generalmente admitida la hipótesis de Kerckhoff que establece que la seguridad del cifrado debe residir , exclusivamente ,en el secreto de la clave, y no del mecanismo de cifrado . Así pues, se acepta que el criptoanalista conoce el método de cifrado siendo, por tanto, su único interés obtener la clave de descifrado.

Son varias las situaciones que se puede encontrar, o que puede elaborar, un criptoanalista en su trabajo.

En casos más desfavorables para él, se presenta cuando sólo tiene acceso al texto cifrado, en cuyo caso el ataque se denomina sólo al criptograma. En estas condiciones, aun conociendo el algoritmo de cifrado, únicamente puede intentar vulnerar dichos algoritmos, o bien proceder a probar todas las claves del espacio de claves. Por motivos obvios en este último caso, el ataque también se denomina de búsqueda exhaustiva o simplemente exhaustivo.

Una segunda posibilidad, más ventajosa para el atacante, se produce si este conoce el criptograma y el correspondiente texto en claro, así como el algoritmo de cifrado. Partes del texto en claro son frecuentemente conocidas o pueden ser conjeturadas. Por ejemplo, los mensajes transmitidos mediante protocolos normalizados (EDI, X-400) tienen los mismos símbolos en las mismas posiciones , así como ciertos campos (remitente y dirección de remisión, etc) cuyo texto en claro es fácil de adivinar. Este tipo de criptoanálisis se denomina ataque mediante texto en claro conocido.

Finalmente, la última y mejor posibilidad se le presenta si puede elegir un texto en claro de cualquier longitud y obligar a su cifrado. Este tipo de intrusión se denomina ataque mediante texto en claro escogido. Se presenta, por ejemplo, cuando un usuario de un ordenador tiene acceso a la tabla cifrada de contraseñas. Introduciendo
repetidamente distintas contraseñas puede llegar a recopilar un gran número de pares: texto en claro-texto cifrado.

Aunque se pueden presentar más casos, los anteriores son los más frecuentes.

4.5 Criptosistemas de clave secreta.

En estos criptosistemas las claves de cifrado y de descifrado son iguales o, caso contrario, del conocimiento de una de ella se deduce, en un tiempo y con unos recursos razonables, la otra. En la exposición que sigue consideraremos las claves iguales (lo que además es la situación usual)ñ en caso de que sean distintas , las fórmulas correspondientes se pueden estudiar en el apartado siguiente. Llamando a esta clave única Ki, y dando un mensaje en claro mi, el criptograma ci se obtendrá mediante el proceso de cifrado

$$Ci= E(ki, mi)$$

y el receptor obtendrá el mensaje en claro mediante la operación de descifrado:

$$mi= D(ki,ci)$$

Obviamente, la consistencia del criptosistema comporta que:

$$mi=D(ki,e(ki,mi))$$

para todo $ki \in K$ y todo $mi \in M$.

La principal ventaja es la simetría del criptosistema, pues los papeles del emisor y receptor son intercambiables fácilmente al emplear ambos la misma clave.

Por otro lado el receptor tiene la seguridad de que el emisor es quien dice ser y no un suplantador, pues de otro modo no podría descifrar el texto cifrado con la clave secreta, sólo conocida, hipotéticamente, por el emisor y el receptor. Es decir, este tipo de criptosistemas mediante la confidencialidad de la información y confiere autenticidad al emisor.

Sin embargo, también presenta dos importantes desventajas. Primeramente, la distribución de claves exige un canal seguro, que no puede ser por el que transita el criptograma, pues éste, por hipótesis, es vulnerable a posibles interceptaciones. Habitualmente, el canal seguro se consigue mediante un mensajero.

Además en un criptosistema simétrico con numerosos interlocutores el número de claves implicadas es muy grande. En efecto por cada dos interlocutores se precisa una clave, así, si el número de éstos es n se tendrán en total:

$$(n(n-1))+2$$

claves, Por ejemplo, para n=20, el total de claves sería de 190, número evidentemente muy elevado.

4.5.1 Cifrado DES

En 1977 el National Bureau of standard de EE.UU. (actualmente National Institute of standard and Technology, NIST) anunció un nuevo algoritmo criptográfico , que denominó Data Encription Standard, DES, para ser usado obligatoriamente en el cifrado de informaciones gubernamentales no clasificadas.

A partir del algoritmo DES se construyen criptosistemas simétricos, cuya clave está formada por 64 bits, de los cuales sólo 56 son significativos, pues el octavo de cada octeto es de paridad. El texto que se va a cifrar se descompone en bloques de 64 bits, cifrando cada bloque con la clave citada. Esta clave, por los motivos que inmediatamente se expondrán, se denomina algunas veces clave externa y es la única que manejan tanto el emisor como el receptor.

En síntesis, el algoritmo consta de 16 etapas o iteraciones. Cada etapa utiliza interiormente una clave (denominada interna), diferente de una iteración a otra, generada a partir de la clave externa ya

citada. En cada una de estas etapas, los únicos procesos que se realizan son operaciones lógicas: "o exclusivos " permutaciones y sustituciones. Estas últimas, más sofisticadas que las restantes, son complejas sustituciones no lineales y constituyen el núcleo del algoritmo.

El algoritmo DES se puede implementar tanto en hardware como en software, prefiriéndose la primera por la rapidez de cifrado conseguida, que llega a ser de 50 Mbs.

Desde su aparición, la controversia acerca de la seguridad de este algoritmo no ha cesado, a pesar de lo cual todavía hoy en día sigue siendo el procedimiento criptográfico más usado en todo el mundo. Como prueba de ello se estiman en varios cientos de miles los dispositivos físicos DES vendidos anualmente.

Aunque no se haya demostrado matemáticamente que presente ninguna vulnerabilidad , lo que parece claro es que, con la actual potencia de cálculo , un algoritmo basado en una clave de tan sólo 56 bits es vulnerable a un ataque exhaustivo en poco tiempo, probando con todas las posibles combinaciones de tal número de bits . Por ello sigue el convencimiento generalizado de que próximamente será retirado como estándar federal estadounidense.

4.5.2 IDEA

Debido a la previsible caducidad del DES, prestigiosos criptógrafos de todo el mundo han venido trabajando desde los últimos años de la década de los 80 para encontrar algoritmos adecuadamente compatibles con el mismo (dado el gran número de equipos de cifrado que en todos los países utilizan DES), a la par que suficientemente robustos para sustituirle.

Entre los numerosos algoritmos desarrollados con el objetivo citado, el mejor, en opinión de muchos expertos, es el denominado IDEA (International Data Encryption Algorithm) . Desarrollado en Suiza por Xuejia Laiy James Massey en el Instituto para el Tratamiento de Señales e Información del Instituto Federal de Tecnología Suizo, fue bautizado en 1990 con el nombre de PES (Proposed Encription Standard) y conocido posteriormente, tras algunas mejoras como, IPES (Improved Proposed Encryption Standard), para concluir, en 1992, denominándose IDEA.

Concebido para que fuese resistente a ciertos ataques a los que el DES es vulnerable, IDEA es un cifrador de bloques, que opera sobre textos en claro de 64 bits con una clave de 128 bits. El algoritmo usado en el descifrado es el mismo que en el empleado en el cifrado.

Como otros muchos cifradores de clave secreta, IDEA usa tanto técnicas de confusión como de difusión, para dificultar la tarea al potencial atacante. IDEA realiza complejas operaciones matemáticas (o-exclusivo , adiciones módulos 2 a la 16 y multiplicaciones módulos 2 a la 16) sobre sub-bloques de 16 bits de entrada. Esto último es especialmente destacable, pues significa que el algoritmo es eficiente incluso con procesadores de 16 bits.

Esencialmente, el bloque de 64 bits de entrada se divide en 4 sub-bloques de 16 bits, que constituyen la entrada a la primera de las 8 iteraciones, o pasos, idénticos por los que transita el texto en claro. En cada paso, cada sub-bloque es sometido a las operaciones ya citadas con los restantes, y con los 8 sub-bloques de 16 bits obtenidos de la clave de 128 bits.

Los programas actuales que implementan IDEA son tan rápidos como los que implementa DES. En efecto, en Intel 386 a 33 MHZ se llegan a obtener velocidades de 880 kbps y en un VAX 9000 esta velocidad se multiplica por cuatro. Si la implementación se realiza físicamente, un microprocesador desarrollado con este fin por el ETH en Zurich alcanza los 177 Mbps con una velocidad de reloj de 25 Mhz.

Una medida de la importancia actual de este algoritmo, y de su grado de penetración en el mercado, se tiene en los numerosos paquetes de programas de seguridad para ordenadores personales o redes que incluyen IDE a como algoritmo de cifrado. Es el caso, por ejemplo, del PGP, conocido producto de seguridad para redes que también se verá en este capítulo.

IDEA se utiliza mediante un comando con la siguiente sintaxis:

idea -k "valor" fichero_de_entrada fichero_de_salida

donde valor es una cadena alfanumérica que sirve para inicializar el algoritmo de cifrado, comportándose como una clave privada, por lo que debe tener un tratamiento, en cuanto a su custodia, similar al de una contraseña. Cualquier otra persona que conozca su valor podrá descifrar la información del fichero.

A continuación se muestra un ejemplo de cómo cifrar, primero, y descifrar, después, el texto en claro " Este es el contenido de un fichero original" mediante la contraseña " H1opelo" .

```
$ pg foriginal
Este es el contenido del fichero original
$ idea -k "h1opelo" foriginal fcifrado
$ rm foriginal
$ pg fcifado
Í S ûr B1   n
$ idea -d -k "hlopelo" fcifrado foriginal
$ rm fcifrado
$ pg foriginal
Este es el contenido del fichero original.
$
```

4.6 Criptosistemas de clave pública

En estos criptosistemas, también llamados asimétricos, las claves de cifrado y de descifrado son diferentes. Además , del conocimiento de una no se deduce la otra .

Llamando a la clave de cifrado (clave pública) Ku y dando un mensaje en claro mi, el criptograma ci se obtendrá mediante el proceso de cifrado:

$$ci = E(Ku, mi)$$

y el receptor recuperará el mensaje en claro con la clave de descifrado (clave privada) kv mediante el proceso de descifrado.

$$Mi= D(kv,ci)$$

Obviamente, la consistencia del método conocida que

$$Mi=D(kv,E(ku,mi))$$

para todo mi $\in$ M,kv,ku, $\in$ K,

Es de reseñar para las claves públicas y privadas son intercambiables, pudiéndose cifrar con la primera y descifrar con la segunda, que es lo usual, o bien cifrar con la privada, en cuyo supuesto se descifraría con la pública.

Este tipo de criptosistemas obvia las dos dificultades inherentes a los simétricos. Por un lado, no exige un canal seguro para la distribución de la clave de cifrado (pública), pues ésta es común a todos los que deseen remitir informaciones cifradas al receptor. Consiguientemente, debe ser de universal conocimiento.

En segundo lugar, si el criptosistema está compuesto por n emisores-receptores que desean intercambiar mensajes cifrados, el número total de claves implicadas será 2n. En efecto, por cada interlocutor tendremos dos claves una pública con la que los restantes cifrarán la información a él dirigida, y una privada con la que descifrará estos criptogramas. Por ejemplo, en caso de tener 20 interlocutores, el número de claves sería de 40, frente a las 190 necesarias en los esquemas de cifrado simétrico.

En contra, estos métodos de cifrado no permiten la autenticación inmediata del emisor. Evidentemente , al cifrar todos los emisores con la misma clave pública del receptor, éste no tiene medio de cerciorarse de que la información que recibe de un supuesto emisor procede de él y no de un suplantador. El inconveniente se puede solventar mediante la firma digital.

4.6.1 Cifrado RSA

En 1977 Rivest, Shamir y Adleman propusieron un algoritmo de clave pública basado en la factorización de grandes números. El algoritmo, denominado con las iniciales de sus autores, RSA, es el prototipo de los de clave pública, por lo que se expone aquí.

Un usuario, A, que desee recibir informaciones cifradas mediante el esquema RSA, elegirá dos números primos muy grandes (usualmente de unos cien dígitos decimales) p,q, que mantendrá en secreto y cuyo producto se denominará n. A continuación, determinará un número d, tal que

$$m.c.d.\ (d,(p-1)\ (q-1))=1 \quad (3.1)$$

y seguidamente calculará su inverso multiplicativo e, respecto de módulo (p-1) (q-1)

$$e.d= 1 \bmod (p-1)\ (q-1)$$

Realizando estos cómputos, A daría a conocer la clave pública (e,n) y mantendría a buen recaudo el número d, que junto con n constituyen la clave privada (d,n).

Cuando un emisor, B, deseara remitir un mensaje cifrado m a A , obtendría el criptograma mediante la operación de cifrado

$$ci=mi \bmod n$$

y A recuperaría mi realizando la operación de descifrado

$$mi = c^{d}_{i} \bmod n \quad (3,2)$$

La fortaleza del método radica en la dificultad de factorizar números muy elevados. Así no pudiendo hallar los factores primos de n, es decir, p, q, es imposible calcular d a partir de (3.1).

Se podría pensar en obtener d a través de un ataque mediante texto en claro conocido o escogido (apartado 3.4). Para ello , conocido mi y su correspondiente si se podría despejar d de (3.2) obteniendo:

$$d = lg\ mi\ (mod(p-1))$$

con lo que se habría descubierto la clave privada. Desgraciadamente para los criptoanalistas, el cálculo de logaritmos discretos es un problema de enorme complejidad, lo que hace impracticable este tipo de ataque.

Naturalmente, cualquiera de los ataques anteriores sería viable si n fuera un número pequeño. Por ello, los autores del algoritmo recomiendan tomar p y q del orden de 100 cifras decimales; o sea n de aproximadamente 200 dígitos , o lo que es igual unos 664 bits. Prácticamente se toman n = 512 bits, aunque para mayor seguridad se puede elegir n = 1024 bits o más. Por el contrario, si las amenazas previstas no son muy grandes se puede seleccionar n =256 bits.

La implementación del algoritmo RSA se realiza, para valores de n elevados, exclusivamente en hardware , con el auxilio de dispositivos DSP (Digital Signal Processing); de otro modo el proceso de cifrado y, sobre todo, de generación de los números primos p y q sería prohibitivamente lento. Aun así, la velocidad lograda no supera algunas decenas de kilobits por segundo; o sea, mucho menor que la obtenida en el algoritmo DES.

Por lo concerniente a la seguridad, el método de cifrado expuesto ha sido criptoanalizado exhaustivamente, sin que se haya encontrado ninguna vulnerabilidad intrínseca. Con todo para reforzar la fortaleza del esquema, se recomienda elegir los números primos p, q, de la forma 2p + 1, donde p es un número primo muy elevado. De este modo la dificultad de factorizar n se incrementa considerablemente.

A diferencia del DES, el algoritmo RSA es propiedad de una firma comercial, RSA Data Security Inc. , quien licencia los derechos de uso. Ello incrementa el costo de su empleo.

4.7 Firma Digital.

Es un mecanismo de seguridad que se apoya en las técnicas criptográficas de clave pública contempladas en anteriores apartados.

Se realiza añadiendo un conjunto de datos a un mensaje, o bien transformando este, con el fin de proteger el mecanismo contra alteraciones, al tiempo que el receptor pueda probar el origen e integridad de los datos recibidos.

Así pues, dicho mecanismo (conocidos por algunos como firma electrónica) cumple tres cometidos: validar el remitente, imposibilitar la alteración del mensaje y resolver disputas por arbitraje. De este modo, la firma digital añade a las conocidas propiedades de la firma manuscrita la característica de inhibir cualquier posible alteración de los datos.

Como se señaló, la firma digital se basa en la criptografía de clave pública. El procedimiento es muy simple: Creando el remitente A, de un mensaje desea firmar el mismo, lo cifra mediante un algoritmo asimétrico con su clave privada K_{vA}

De esta manera, cualquiera que conozca la correspondiente clave pública de A, K_{uA} (por hipótesis todo el mundo al ser ésta de general conocimiento) podrá descifrar el criptograma y averiguar su contenido; sin embargo, si lo modificase en su provecho no podría cifrarlo posteriormente al desconocer la clave privada de A, con lo que su trabajo habría sido en vano. Por otra parte, el legítimo receptor concluirá sin duda que el mensaje remitido por A, pues es capaz de descifrarlo usando la clave pública de éste.

Si además A desea preservar la confidencialidad del mensaje remitido a B puede cifrarlo primeramente con clave pública de B, K_{uB} , y después con su clave privada K_{vA}, para garantizar la integridad y autenticidad del remitente. Evidentemente estas dos transformaciones se pueden invertir (cifrado primero con K_{vA} y después con K_{uB}) considerándose el mismo efecto: mantener la confidencialidad e integridad del mensaje y validar la autenticidad del emisor.

El proceso efectuado en la práctica es ligeramente diferente, pues el consumo de tiempo y recursos que supone cifrar con un algoritmo asimétrico todo el mensaje aconseja la elección de otro procedimiento distinto de firma. La solución viene de la aplicación al mensaje de una función de comprobación aleatoria (función hash)que, como es sabido, reduce el mensaje a un conjunto de datos,

denominados resumen, de longitud mucho menor que aquél , usualmente 128 ó 254 bits. Adicionalmente, una función de comprobación aleatoria debe satisfacer la condición de que sea despreciable la probabilidad de que mensajes diferentes produzcan resúmenes idénticos. De las numerosas funciones de este tipo que se pueden encontrar en el mercado, merece destacar el algoritmo MD5 de la firma RSA Data Security Inc.

Con el auxilio de estas funciones, el proceso habitual de firma se desarrolla como sigue: en primer lugar se aplica la función de comprobación aleatoria que produce un resumen de pequeña longitud; a continuación, se cifra el resumen mediante un cifrador asimétrico usando la clave privada del remitente. Finalmente, éste remite el mensaje en claro concatenando con un apéndice constituido por el resumen firmado de la manera indicada.

Si además A desea preservar la confidencialidad del mensaje puede cifrar éste con la clave pública de B. Con todo ello , A remitiría a B lo siguiente:

$$E (M, K_{uB}) + e(R,K_{uA})$$

donde se ha representado mediante el signo + la concatenación de los dos criptogramas formulados.

4.7.1 PGP

Estas siglas corresponden a las iniciales de " Pretty Good Privacy ", que es un programa de cifrado de dominio público creado por Philip Zimmermann . Las dos principales funciones del PGP son el cifrado de mensajes y la firma digital del mismo. Para la primera el usuario puede elegir entre los algoritmos RSA (con longitud de clave 512, 1024, ó 2048 bits) e IDEA , mientras que la firma se obtiene cifrando vía RSA con clave privada del remitente el resultado de la aplicación al mensaje de la función de comprobación aleatoria MD5.

He aquí un resumen de los comandos de usuario más significativos del PGP.

- Cifrar un mensaje con clave pública de un receptor:

$ pgp -e foriginal clave_publica_receptor

esto crea un fichero cifrado foriginal. PGP que contiene el texto cifrado.

- Firmar un mensaje:

  $ pgp -s foriginal

- Firma y cifrado de un mensaje:

  $ pgp - es foriginal clsve - publica-receptor

4.8 Criptosistema irreversibles.

Son aquellos que nos permiten el descifrado del criptograma. Es decir, una vez cifrado un cierto texto en claro, es imposible recuperar éste a partir del correspondiente texto cifrado. Se basa en funciones matemáticas computacionalmente no invertibles; o lo que es igual, funciones cuya inversa no es posible computar, ni aun con las máquinas más potentes de hoy en dia, en un tiempo razonable. Este tipo de esquemas criptográficos encuentra su principal aplicación en la autenticación de entidades, es decir, en la verificación de que una entidad (por ejemplo, un usuario) es quien pretende ser y no un suplantador.

Una forma sencilla de construir una función matemática computacionalmente no invertible, consiste en elegir un algoritmo computacionalmente seguro de cifrado E, y un mensaje en claro prefijado y conocido, por ejemplo, m_0. Para cada clave, k se obtendrá un criptograma diferente:

$$c_k = E(k, m_0).$$

Considerando ahora el mensaje a cifrar, m_i, como la clave del algoritmo, se tendrá:

$$c_i = f(m_i, m_0), \qquad m_0 \text{ fijo}$$
$$m_i \in K$$

En tanto en cuanto E sea computacionalmente seguro , es decir, imposible de violar con la actual potencia del cálculo , no será factible determinar la clave de cifrado, o sea m_i , mediante un ataque del tipo de texto en claro conocido; es decir, apartir de pares mensajecriptograma .

Este tipo de esquemas se usa con gran generalidad para el almacenamiento seguro de las contraseñas de los usuarios de un sistema de informático. Así procede, por ejemplo, el sistema operativo UNIX, empleando como algoritmo la función de biblioteca crypt(3) , basada en el DES. La función de biblioteca crypt(3) toma la contraseña de usuario como clave de cifrado del DES, usándola para cifrar un texto en claro consistente en un bloque dado de 64 bits (en algunos sistemas todos ceros). El resultado es un texto cifrado de 64 bits, que vuelve a ser criptografiado con la misma contraseña del usuario. El proceso se repite 25 veces. El bloque final de 64 bits se concatena con dos bits cero obteniéndose 66 bits, que se transforman en 11 caracteres imprimibles, mediante un código de 6 bits por carácter, que se almacenan en el fichero /etc/passwd.

La razón de repetir el proceso anterior 25 veces es complicar el trabajo a cualquier potencial atacante que trate de adivinar una contraseña mediante una búsqueda exhaustiva.

Una vulnerabilidad de este proceso se presenta si la tabla de contraseña es de acceso público, como sucedía con las primeras versiones de UNIX. En este caso, eligiendo aleatoriamente contraseñas, cifrándolas y observando el criptograma resultante (ataque mediante texto en claro escogido), se podría comprobar si éste coincide con alguno de los almacenados en la tabla de contraseñas. De ser así, se habría averiguado la contraseña de algún usuario del sistema.

Para dificultar este tipo de ataque, crypt(3) genera aleatoriamente un número de 12 bits denominado salt (condimento) , cifrándose el resultado de concatenar la contraseña de usuario y el " salt ".

Cuando se cambia de contraseña (o sea da de alta a un usuario), el programa /bin/passwd genera un "salt" basándose en la hora del día. El "salt" es convertido en una cadena de dos caracteres y almacenado en el fichero /etc/passwd junto con el texto cifrado y el código de identificación (ID) del usuario. Cuando se introduce el ID en el sistema se recupera el "salt", se concatena con la contraseña de usuario y , tras cifrar el resultado se compara con el texto cifrado contenido en /etc/passwd.

En la tabla 4.1 se observan distintas contraseñas y sus "salt" correspondiente (según el momento de su creación). Igualmente, se muestra en la tabla 4.2, el aspecto de los campos: ID y texto cifrado, del fichero /etc/passwd. Este fichero contiene además los campos Usuario y Contraseña.

ID	Contraseña	"SALT"
Maria	3Petaca	14
Andrés	rOmerin5	Y5
Eugenia	Hospi??	K6
Ramiro	Golf56	48
Sonia	&VelaP	t9
Pedro	Real/8	QB

Tabla 4.1 Identificación contraseña (sin cifrar) y "SALT"

Maria	14C7P2ioSNfrs
Andrés	Y57mz2TF2OW1o
Eugenia	k6a2cN3TPPmEE
Ramiro	48HQAUNOUbeH2
Sonia	t9dOu42NpXQw
Pedro	QBhnAYg2rmg3E

Tabla 4.2 Aspecto del fichero /etc/passwd

Como se ve, el sistema operativo UNIX almacena el "salt" como los dos primeros caracteres del texto cifrado. Es evidente que dos usuarios con la misma contraseña pueden tener 2 a la 12 igual a 4096 textos cifrados diferentes.

Análogamente, el sistema operativo de IBM, MVS con RACF (" Resources Access Control Facility "), posibilita el cifrado de contraseñas usando el algoritmo DES en la misma forma explicada para el sistema operativo UNIX.

4.9 Técnicas Criptográficas básicas.

Durante la primera etapa de la historia de la criptografía, los métodos de cifrado empleados eran rudimentarios y se pueden calificar más de artísticos que de científicos o técnicos. Ejemplo de ello es la escítala lacedemonia la escritura de un mensaje en el cuero cabelludo de un esclavo, que el crecimiento del pelo hacía inobservable ; aunque en propiedad este último procedimiento más se podría calificar de esteganografía que de criptografía.

Sin embargo, en tiempos de la Roma clásica aparece un método de cifrado, denominado CESAR, de base matemática y que todavía ejemplifica las técnicas de sustitución que a continuación se exponen.

4.9.1 Métodos de sustitución y de permutación.

Las técnicas clásicas de cifrado pertenecen a dos tipos denominados de sustitución y de permutación. En las primeras, los símbolos (caracteres o bits) del texto en claro se reemplazan por otros, siguiendo una o varias reglas de sustitución. En las segundas, estos símbolos se reordenan para aparecer en posiciones distintas de las que tenía en el texto en claro.

El caso más sencillo, dentro de las técnicas de sustitución, es el denominado método de sustitución simple monoalfabeto . Este procedimiento consiste en cambiar cada carácter de un alfabeto por aquel situado k_m posiciones por delante de él en dicho alfabeto.

En el caso de k=3, el método se denomina CESAR, por ser utilizado en la época de Julio César para cifrar los mensajes intercambiados entre sus generales y senadores. Representando con mayúsculas los caracteres del alfabeto del texto en claro y en minúsculas los del texto cifrado, el método se expresa:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20...26
A B C D E F G H I J K L M N Ñ O P Q R S T... Z

d e f g h i j k l m n ñ o p q r s t u v w... c

donde, como es habitual, se han suprimido las letras dobles "ch" (recién eliminada, en las fechas en que se escribe este texto, de nuestro alfabeto) y "ll".

Así el mensaje:

"ESTO ES UN EJEMPLO DE CIFRADO CESAR "

se convierte en :

" hvwrhvxphmhosñrghfliudgrfhvdu"

habiéndose eliminado, como se hace usualmente, los blancos entre palabras para no dar pistas a los posibles criptoanalistas.

En general los métodos de sustitución monoalfabeto se formulan mediante la expresión

$$E(m_i) = (a_i + b) \bmod n^3$$

donde a y b son constantes naturales, n es el numero de letras del alfabeto (en español n = 27, pues no se incluyen las letras dobles), i y E(i) representan las posiciones de los caracteres en claro y cifrado, respectivamente, dentro de su correspondiente alfabeto.

Para que el algoritmo se biunívoco debe cumplirse que m.c.d. (a,n) =1. Evidentemente, el método César es un caso particular, con a=0 y b=3.

Como ejemplo, sea la transformación E (m_i) = (5i + 15) mod 27. Los alfabetos del texto en claro y del texto cifrado serán:

0 1 2 3 4 5 6 7 8 9 ... 18 19 20 21 22 23 24 25 26
A B C D E F G H I J ..... R S T U V W X Y Z

15 20 25 3 8 13 18 23 1 6 ... 19 24 2 7 12 17 22 0 5
o t y d i n r w b g... s x c h m q v a f
Así . el texto.

" ESTO ES UN EJEMPLO DE SUSTITUCION MONOALFABETO "

Quedaría cifrado :

" ichjicmzigiuñpjdicmchbhmybjzujzopnotihj "

Otros métodos del mismo tipo son la sustitución de polialfabeto, la sustitución poligrámica . En la primera, en vez de utilizar un solo alfabeto se emplean dos o más. En el caso de dos alfabetos, un carácter se transforma ,según que ocupe una posición par o impar en el texto en claro, en uno de entre dos posibles.

Las situaciones homofónicas transforma cada letra del texto en claro carácter elegido de entre un conjunto de caracteres del alfabeto cifrado , llamados homófonos . Habitualmente los homófonos son números decimales.

Los tres tipos de métodos de sustitución expuestos reemplazan carácter a carácter el texto en claro. A diferencia de estos métodos , los cifrados poligramáticos reemplazan bloques de caracteres por otros bloques de igual longitud. Si la sustitución es de dos caracteres por otros bloques de igual longitud. Si la sustitución es de dos caracteres por otros tantos, el método se llama diagrámicos . Se tendrá un sistema trigrámico si el reemplazamiento se hace de tres caracteres en tres caracteres, etc.

Los métodos de transposición, también llamados permutación, cambian la posición de los caracteres dentro del texto en claro, produciendo así un texto cifrado en el que aparecen los mismos caracteres pero en diferente orden.

Son varios los procedimientos que se emplean. En uno de ellos, los caracteres se disponen siguiendo un patrón geométrico y después se extraen según una pauta determinada. Así se puede considerar una matriz bidimensional de determinado número de filas y columnas que se rellenan por filas, y del que se van obteniendo los caracteres por columna. Por ejemplo, sea una matriz de 8 x 5 y el mensaje

" ESTE ES UN EJEMPLO DE TRANSPOSICION COLUMNAR "

rellenando la citada matriz se obtiene:

ESTEE
SUNEJ
EMPLO
DETRA
NSPOS
ICION
COLUM
NARXX

donde se han completado con XX los huecos resultantes. El texto cifrado sería:

" esednicn sumescoa tnptpilr eelrooux ejoasnmx"

en el que, por facilidad de lectura , se han separado con blanco las distintas columnas.

Otro procedimiento consiste en dividir el texto en claro en bloques de caracteres, y permutar éstos dentro de cada bloque siguiendo un cierto esquema. Por ejemplo, se podría considerar la permutación:

{1,2,3,4,5,6}, f(1)=4, f(2)=1, f(3)=6,
f(4)=2, F(5)=3,f(6)=5

con lo cual, el texto en claro:

" EJEMPLO DE TRANSPOSICION"

se dividirá en bloques de seis letras, permutando a continuación éstas:

EJMPL ODETRA NSPOSI CIONXX
meljep toader onisp ncxiox

El texto cifrado sería:

" meljeptoaderonispncxiox"

4.9.2 Confusión y difusión.

Desde Shannon es generalmente admitido que los principios básicos que deben orientar la creación de cifradores son la confusión y la difusión.

Se entiende por confusión que la frecuencia de aparición de los símbolos cifrados no revele ninguna información sobre los símbolos en claro de los que proceden. De esta manera, las propiedades estadísticas de los símbolos del lenguaje (por ejemplo, en castellano la letra q va siempre seguida de la u, la letra e es la que más frecuentemente aparece, no hay palabras con cuatro consonantes consecutivas, etc) no proporcionan ninguna información al criptoanalista.

Por contra, difusión significa que la influencia de un símbolo en claro se disperse sobre el mayor número posible de símbolos cifrados. De esta manera, las propiedades estadísticas de las palabras del mensaje se diseminan por todo el criptograma.

De uno u otro modo se consigue que las propiedades estadísticas de los símbolos del lenguaje (confusión) o de la palabras del mismo (difusión) no proporcionen ninguna información a un atacante.

4.9.3 Cifrado producto.

Un modo usual de crear confusión y difusión es usar un cifrado producto, que consiste en la aplicación sucesiva (también denominada composición) de diferentes cifradores, cada uno de los cuales actúa sobre el criptograma (texto en claro para él obtenido de aplicación del cifrado anterior. Representado por E_i, $1 \leq i \leq n$, los distintos métodos criptográficos empleados, por M el texto en claro y por C el criptograma obtenido, el cifrado producto se puede expresar como:

$$C= E_n (...E_2(E_1(M))...)$$

Los cifrados productos elevan notablemente la seguridad de cada uno de los cifrados que lo componen, siendo muy empleados cuando se desean usar métodos criptográficos de dudosa fortaleza.

Un ejemplo clásico de cifrado producto es el método DES que, como se ha estudiado, está compuesto de sucesivos cifrados de sustitución y transposición.

El propio método DES puede ser usado iterativamente para incrementar su fortaleza. En el cifrado producto ideado por Matyas y Meyer , el emisor cifra mediante una clave k_1, descifra con otra clave k_2 y vuelve a cifrar el resultado con la primera clave, k_1, empleando siempre el citado DES. Simbolizado por E_{k1} el cifrado DES con clave k_1 y por D_{k2} la transformación de descifrado con el mismo método y clave k_2, el criptograma C se obtiene mediante la secuencia de operaciones:

$$C = E_{k1}(D_{k2}(E_{k1}(M))).$$

El texto en claro M es recuperado por el receptor a través de las transformaciones.

$$M= D_{k1}(E_{k2}(D_{k1}(C))),$$

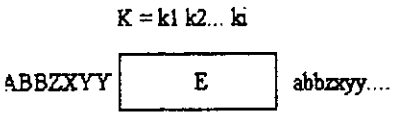
siendo D_{k1} y E_{k2} , respectivamente, el descifrado DES con la clave k_1 y el cifrado con el mismo método con clave k_2.

4.9.4 Cifradores de flujo y cifradores de bloque.

Los cifradores de flujo descomponen el mensaje en símbolo (caracteres o bits), $M= m_1,m_2,...m_n$, y cifran cada uno de éstos con el correspondiente símbolo de una clave de cierta longitud (idealmente infinita) $K= k_1,k_2...k_n$. El criptograma se expresa como

$$E_k (M) =E_{k1}(m_1)E_{k2}(m_2)...E_{ki}(m_i)$$

donde el algoritmo de cifrado E_{ki} depende k_i y , quizás, de una cierta información de control. Una ilustración de un cifrado de flujo se tiene en la figura 4.2.



CIFRADOR DE FLUJO
Figura 4.2

En contraste con estos procedimientos criptográficos, los cifradores de bloque descomponen el mensaje en claro en bloques de símbolos (caracteres o bits) de igual longitud, cifrando cada uno de ellos con la misma clave. Representando por m_1, m_2,...,m_n los símbolos de bloque de longitud n, y la clave y criptograma del bloque por K y C respectivamente, el cifrado de bloque se puede escribir como

$$C = E_k(m_1)\ E_k(m_2)...\ E_k(m_n).$$

El ejemplo típico de un cifrador de bloque se tiene un cifrador de transposición.

Los cifradores de flujo presentan, frente a los de bloque, una importante ventaja: en aquellos la transformación se hace carácter a carácter, o bit a bit, del mensaje en claro, por lo que cada uno de éstos se cifra nada más llegar al dispositivo criptográfico, consiguiéndose altas velocidades de cifrado respecto de las alcanzadas en los cifradores de flujo.

Por contra, la información de cada símbolo en claro está contenida exclusivamente en el correspondiente símbolo cifrado, con lo que la difusión de la información es muy pequeña. Sin embargo, en los cifradores de flujo de información un símbolo en claro se difunde en numerosos símbolos cifrados.

CAPITULO 5

LIBRO NARANJA

5.1 Conceptos sobre seguridad.
5.2 Requisitos para una política de seguridad.
5.3 Requisitos de responsabilidad.
5.4 Requisitos de confiabilidad.
5.5 Requisitos de documentación.
5.6 Características de las clases.

Los objetivos del libro naranja son:

- Medición. Proporciona a los usuarios una métrica para valorar el grado de fiabilidad que pueden depositar en el ordenador para procesar de forma segura información sensible. Por ejemplo, un usuario puede fiarse más de un sistema B2 que de uno C2.
- Guía. Proporcionar una directriz a los desarrolladores para construir productos comerciales fiables, que satisfagan los requisitos de las aplicaciones sensibles.
- Adquisición. Proporcionar una base para indicar requisitos de seguridad en las especificaciones de adquisiciones informáticas.

5.1 Conceptos sobre seguridad.

El libro naranja habla de sistemas fiables más de que sistemas seguros. Ningún sistema es completamente seguro puesto que en cualquiera puede penetrarse con las herramientas adecuadas y el tiempo suficiente, Lo que se puede es valorar la fiabilidad de los sistemas informáticos para que hagan lo que se les pide. El concepto fundamental del Libro Naranja es que se pueda medir la fiabilidad y certificar un sistema respecto a un conjunto de criterios de seguridad.

La política de seguridad es un conjunto de reglas y prácticas que regulan la forma en que una organización maneja, protege y distribuye información sensible. Como parte de toda política de seguridad hay sujetos y objetos. Un sujeto es una entidad activa dentro del sistema (usuarios, procesos); por el contrario un objeto es una entidad pasiva sobre la que actúa un sujeto. Ejemplos de objetos son ficheros, directorios, dispositivos, ventanas, mecanismos para la comunicación entre procesos.

La política de seguridad técnica expresa las reglas de seguridad impuestas a un sistema de información. Por ejemplo, las normas que determinan si un usuario puede acceder a un elemento de información.

La confiabilidad es la garantía que puede depositarse en un sistema y las vías fiables que se han desarrollado, probado, documentado, mantenido y entregado al cliente. Conforme aumenta la clase de seguridad disminuyen las características añadidas, pero aumenta la confiabilidad que un usuario

puede depositar en la arquitectura del sistema y en la implantación de su política de seguridad. En las clases de C1,C" y B1, la confiabilidad se realiza mediante pruebas de los componentes de seguridad del sistema. En las clases intermedias B2 y B3 la confiabilidad se obtiene através del diseño y la realización. En la clase A1 se aplican herramientas de verificación formal

La Base Fiable de Computo (TCB) es el conjunto de mecanismos relevantes a efectos de la seguridad del sistema. No todas las partes del sistema operativo necesitan comprobarse. Un aspecto importante de la evaluación de un sistema informático es identificar la arquitectura, los mecanismos de confiabilidad y las características de seguridad que constituyen la TCB y demostrar cómo está protegida contra interferencias y ataques.

En las clases con una fiabilidad elevada, la TCB se construye en tomo a un monitor de referencias que impone las relaciones de acceso autorizadas entre los sujetos y objetos de un sistema. Cuando el sujeto solicita acceso a un objeto, el monitor obtiene de la base de datos de control la información que le indica si el acceso puede realizarse o no, tras lo cual registrará el suceso acaecido en un fichero de auditoria. Un monitor debe cumplir las siguientes condiciones. ser a pruebas de ataques. mediar en todas las decisiones de acceso. ser imposible de soslayar, y ser lo suficientemente pequeño para que pueda analizarse y garantizarse que se ha probado por completo. En la figura 5.1 se muestra el funcionamiento de un monitor de referencias.

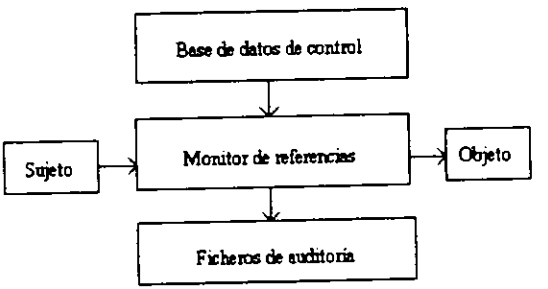


Figura 5 1 Mediación del monitor de referencias entre sujeto y objeto

Por ejemplo, en UNIX C1 los mecanismos de control de acceso del núcleo del sistema operativo constituyen el monitor de referencias. En la figura 5.2 se muestra un caso de funcionamiento cuando el proceso (sujeto) pg intenta abrir el fichero (objeto) .profile para lectura. En esta situación, la base de datos de control son los permisos del objeto, y el fichero de auditoría es una marca de tiempo en el nodo índice del fichero .profile con la fecha y hora en la que se accedió al mismo.

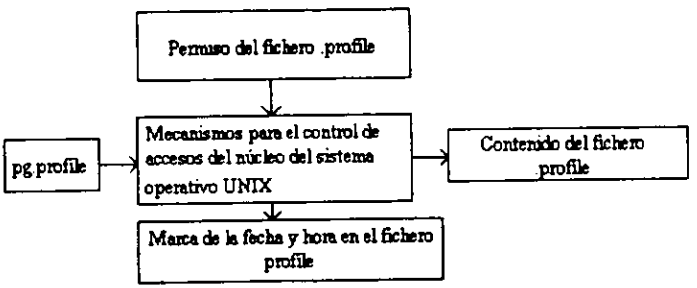


Figura 5.2 Funcionamiento del monitor de referencia en UNIX C1

Un modelo de seguridad expresa matemáticamente los requisitos de la política de seguridad de un sistema de forma precisa. El Libro Naranja se basa en un modelo de máquina de estado desarrollado por David Bell y Leonard LaPadula en 1974.

puede probarse rigurosamente para demostrar que la TCB es completa y a prueba de ataques. En la figura 4.3 se muestra la relación entre política, modelo y mecanismos de seguridad.

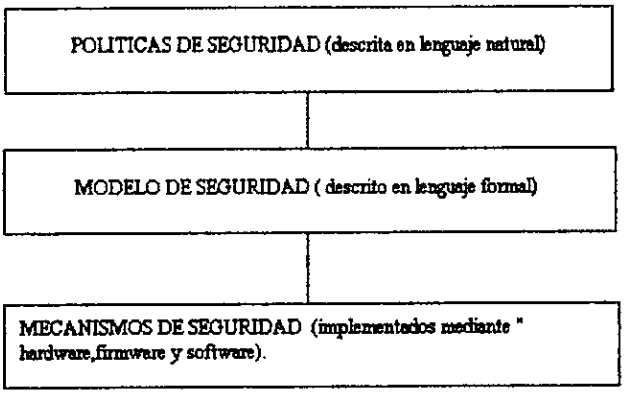


Figura5 .3 Relación entre políticas y mecanismos de seguridad

5.2Requisitos para una política de seguridad.

Una política de seguridad establece las reglas que se deben implementar en el sistema para suministrar el grado necesario de seguridad. El Libro Naranja define requisitos para las siguientes categorías :

- Control de acceso discrecional.
- Reutilización de objetos.
- Etiquetas.
- Control de accesos obligatorios.

En las clases inferiores de seguridad su política es informal y ni siquiera es necesario escribirla. Esto cambia a partir de B1, y en la clases superiores debe expresarse matemáticamente.

5.2.1 Control de acceso discrecional.

Es un método para restringir el acceso a los objetos basados en la identidad de los usuarios y/o grupos de usuarios a los que pertenecen. Los usuarios protegen sus objetos indicando quién puede acceder y el tipo de acceso permitido (lectura, escritura y ejecución). Es decir, el usuario decide a quién cede los derechos de acceso sobre su información.

En la clase C1 el sistema no necesita distinguir entre los usuarios individuales, sino tan sólo entre los tipos de acceso permitidos o rechazados. Un usuario no necesita ser el dueño del objeto para ceder sus derechos de acceso. El sistema no necesita proteger los nuevos objetos creados. En el caso de UNIX C1 hay que ser dueño de un objeto para ceder sus derechos de acceso a otro usuario y siempre se protege a los objetos de nueva creación.

En las clases C2, B1 y B2 el sistema debe ser capaz de distinguir entre los usuarios individuales (controles de acceso capaces de incluir o excluir con la granularidad de un usuario). El usuario debe tener algún privilegio para ceder los derechos de acceso sobre el objeto, generalmente ser su dueño. Asimismo el sistema protege a los objetos nuevos.

En las clases B3 y A1 el sistema debe ser capaz de distinguir entre los diferentes tipos de acceso (lectura, escritura) . Es necesario de disponer de listas de control de acceso, debiendo poder indicarse la denegación de acceso a un usuario concreto. En la mayoría de los sistemas UNIX entre C2 y B2 existen listas de control de acceso.

5.2.2 Reutilización de objetos.

Implica proteger ficheros, memoria y otros objetos de acceso por parte de un usuario tras su uso por otro. Un sistema de control de acceso normal determina quién puede acceder a los objetos una vez que han sido asignados a los usuarios. Los requisitos de reutilización de objetos (a partir de C2) se aplican cuando esos mismos objetos se reasignan.

A continuación se verá mediante un ejemplo la problemática de la reutilización. Supóngase que al crear un fichero el sistema le asigna un conjunto de bloques de disco. A continuación se almacena información confidencial, se imprime y borra el fichero. Si el sistema tan sólo reescribe la cabecera del fichero y no borra físicamente los bloques, podría construirse un programa que leyera discretamente la información confidencial del disco. La reutilización de objetos (residuos) proporciona seguridad garantizando que cuando un objeto se asigna, ubica, o reubica, no contiene información de un uso previo. En el ejemplo, borrando físicamente los bloques correspondientes al fichero.

Con la aparición de periféricos (terminales e impresoras) que poseen memoria local propia se hacen más necesarios los requisitos de reutilización . Algunas de las situaciones que se controlan habitualmente son:

- Borrado de bloques de memoria antes de su asignación.
- Borrado de bloques de disco antes de su reutilización.
- Desmagnetización de cintas o cartuchos que no se vayan a utilizar .
- Borrado de objetos gráficos (ventanas) antes de reasignarse a otro usuario.
- Borrado de memorias de cifrado tras su uso.
- Borrado de tampones en dispositivos con memoria local, tales como terminales o impresoras.

5.2.3 Etiquetas.

A partir de la clase B1 es necesario que cada sujeto (usuario , proceso) y cada objeto (fichero, directorio, ventana) tenga una etiqueta de confiabilidad asociada, mientras que a partir de B2 todos los recursos del sistema, incluidos los dispositivos , deben poseerla. La etiqueta de confiabilidad del usuario indica su nivel de autoridad asociado y se denomina habilitación. La etiqueta de confiabilidad de un fichero especifica el nivel de autoridad que un usuario debe tener para acceder al mismo.

La integridad de las etiquetas garantiza que éstas se asocian a sujetos y objetos que son representaciones correcta de los niveles de seguridad, por ejemplo, la etiqueta confidencial [CONTABILIDAD] debe estar en un fichero confidencial que contenga información sobre contabilidad, y que la habilitación CONFIDENCIAL [NÓMINAS AUDITORIA] se asigne a la persona responsable de las nóminas y auditorias.

Un sistema fiable debe garantizar que cuando se escriba información se mantengan los mecanismos de protección asociados. Para ello es necesario asignar niveles de confidencialidad a los dispositivos de salidas y escribir las etiquetas junto a los datos.

Desde el punto de vista de las etiquetas existen dos tipos de dispositivos mononiveles y multiniveles. En los dispositivos mononiveles tan sólo puede escribirse información con un nivel de confiabilidad. Suelen corresponder a terminales, impresoras, cintas y puertos de comunicación, y el nivel asignado depende de la ubicación física y la seguridad inherente del periférico. Por ejemplo una impresora en una zona pública puede designarse como DESCLASIFICADA , mientras que la utilizada por una persona es SECRETA.

En un dispositivo multinivel puede escribirse información con diversos niveles de confidencialidad. El sistema debe proporcionar una forma para indicar el menor y el mayor nivel permitido, y cuando se escribe información es necesario que el sistema pueda asociar un nivel de seguridad mediante una cabecera para evitar que los usuarios puedan soslayar al sistema de control copiando el fichero a un sistema o dispositivo inseguro, En general, tan sólo los discos están clasificados como dispositivos multinivel.

Es necesario etiquetar aquellas salidas legibles por las personas : impresos, mapas, gráficos. El administrador debe poseer un mecanismo que le permita especificar las etiquetas que deben aparecer en la salida. En primer lugar, cada salida debe etiquetarse al principio y al final de manera que represente la confidencialidad. Por ejemplo, al imprimir un fichero se le antepone y pospone una página que muestra su etiqueta. En segundo lugar, debe etiquetarse cada página impresa con una cabecera y un pie que representen a la etiqueta de confidencialidad.

A partir de B2, los requisitos sobre las etiquetas de confidencialidad del sujeto indican que el sistema debe notificar al usuario cualquier cambio en el nivel de seguridad asociado durante una sesión interactiva . El usuario siempre debe saber en qué nivel está trabajando. Lo habitual es que el sistema muestre la habilitación al conectarse , o cuando hay alguna modificación, o petición del usuario. Esto es necesario porque en general las personas no trabajan con el máximo nivel de seguridad permitido, variándolo según sus necesidades de trabajo.

5.3 Requisitos de responsabilidad.

Mediante los requisitos de responsabilidad el sistema es capaz de identificar a los usuarios, legitimar sus acciones y guardar información sobre todas las actividades que han realizado. Esto agrupa tres tipos de requisitos:

- Identificación y autenticación.
- Vías fiables.
- Auditoría.

5.3.1 Identificación y autenticación.

Es necesario que los usuarios se identifiquen antes de realizar cualquier actividad que implique una interacción con la TCB (ejecutar un programa, leer un fichero =. En los sistemas UNIX la identificación se realiza mediante un nombre de conexión y la autenticación mediante una contraseña.

En C1 el sistema distingue entre usuarios autorizados y los que no lo están. No es necesario que cada usuario tenga un identificador propio ante el sistema. En el caso de UNIX cada usuario tiene su propio identificador de usuario y de grupo. Puede haber varios nombres de conexión con el mismo identificador.

A partir de C2, se exige que el sistema disponga de auditoría. Cada usuario tiene un identificador único que se utiliza para comprobar todas las acciones solicitadas.

5.3.2 Vía fiable

A partir de la clase B2 se requiere que el usuario pueda conectarse desde un terminal hasta el sistema a través de una vía fiable. Para ello existe una secuencia de teclas que al pulsarse elimina todos los procesos actuales y establece una conexión segura con la TCB permitiendo su autenticación. Esto evita ataques sistemáticos contra el sistema mediante programas marcadores y la introducción de caballos de Troya en los programas de conexión al sistema.

5.3.3 Auditoría.

Es el registro y examen de las actividades relacionadas con la seguridad en un sistema fiable. Las actividades relacionadas con la seguridad son los accesos (y sus intentos) de un sujeto sobre un objeto y suele denominarse SUCESOS: conexión, desconexión , manipulación de objetos, alteraciones de privilegios. Incluso el sistema más seguro es vulnerable; auditando sus sucesos se puede saber quién , cómo y cuándo se han producidos los ataques.

La auditoría permite la vigilancia de las actividades de los usuarios , lo que puede prevenir la aparición de problemas de seguridad si los usuarios saben que están siendo observados. Pero también permite la reconstrucción de los hechos para identificar al atacante y eliminar las vulnerabilidades .

Cada vez que se produce un suceso debe almacenarse:

- Fecha y hora del suceso.
- Identificador del usuario generado del suceso.
- Tipo de suceso.
- Si ha tenido éxito o fracaso.
- Punto de origen de la petición (terminal).
- Nombre del objeto involucrado.
- Descripción de las modificaciones en la base de datos de seguridad.
- Clases de seguridad del sujeto y del objeto (a partir de B1).

La información generada mediante la auditoría tiende a ser muy voluminosa, por lo que se convierte en intratable. Un sistema fiable debe proporcionar mecanismos para permitir que el administrador realice una auditoría selectiva tanto de los usuarios como de la propia información auditada .

Según aumenta la clase evaluada se incrementa el número de sucesos auditables. En C2 el sistema debe auditar los sucesos relacionados con la seguridad y proteger la información de la auditoría . El sistema debe asociar una identidad a las acciones realizadas por cada usuario. También debe ser capaz de auditar a nivel de usuario. En B1 el sistema debe ser capaz de auditar cualquier cambio o anulación en los niveles de seguridad, y también hacerlo selectivamente por nivel de seguridad (por ejemplo, todos los usuarios con habilitación SECRETOS). En B2 el sistema debe auditar los sucesos que puedan utilizarse para explotar los canales ocultos de almacenamiento. En B3 y A1 el sistema debe poder monitorizar la acumulación de sucesos de seguridad que puedan indicar una violación inminente de la política de seguridad.

5.4 Requisitos de confiabilidad.

Permiten establecer que la política de seguridad de un sistema fiable se ha construido correctamente y que las características de seguridad del sistema cumplen la política de seguridad. Existen dos tipos de confiabilidad, la operacional se centra en la arquitectura básica y en las características del sistema, mientras que la del ciclo de vida se basa en los controles y estándares para construir y mantener el sistema. Según subimos de clase existen menos características de seguridad adicionales y más requisitos de confiabilidad.

Los requisitos de confiabilidad se agrupan en :

- Arquitectura del sistema.
- Integridad del sistema.
- Análisis de los canales ocultos.
- utilidad para la administración de la fiabilidad.
- Recuperación segura.

Los requisitos de confiabilidad del ciclo de vida se agrupan en:

- Especificación y verificación del diseño.
- Gestión de configuración.
- Distribución segura.

5.4.1 Confiabilidad operacional.

Los requisitos de arquitectura del sistema están relacionados con el diseño de un sistema para que sea posible la seguridad. Hasta B2 no es necesario que haya un diseño orientado hacia la seguridad, aunque deben existir unos principios que permitan incorporar características de seguridad. En todas las clases debe existir un dominio de ejecución protegido para las funciones relevantes de seguridad, que impida su manipulación o modificación inadvertida, así como la protección de los recursos sobre los que se va aplicar el control de acceso y la auditoría . A partir de B1 debe existir un aislamiento de los procesos para que no haya interferencia entre ellos.

A partir de B2 se fuerza la aplicación de una política de privilegio mínimo en el diseño de los módulos para garantizar que los procesos no tengan más privilegios que los que necesita para realizar sus funciones. Los módulos que requieren privilegios del sistema se localizan dentro del núcleo de seguridad, y los demás llaman a funciones privilegiadas tan sólo durante la realización de sus operaciones correspondientes. Por otro lado, se aplica segmentación a la memoria virtual para aislar a los procesos del sistema respecto de los de los usuarios. A partir de B3 el sistema se construye por niveles, que se comunican mediante interfaces bien definidas y ocultan información de los niveles inferiores.

Los requisitos para la integridad del sistema indican que tanto el hardware como el firmware deben probarse para garantizar su funcionamiento. Esto se resuelve mediante la existencia de un conjunto de pruebas de integridad que se ejecutan siempre que se arranca el ordenador , o periódicamente siguiendo un mantenimiento preventivo.

Un canal oculto es una ruta de información que habitualmente no se utiliza como medio de comunicación en el sistema y, por tanto, no está protegido por sus mecanismos de seguridad. Estos canales son una manera subrepticia de comunicar información, y en teoría cualquier dato almacenado o procesado es un canal secreto potencial. Por ejemplo, un usuario puede dar como nombre de un fichero su identificador y contraseña, por lo que cualquier otra persona que vea dicho nombre puede obtener esta información. Existen dos tipos de canales ocultos: de almacenamiento y de tiempo. Los canales ocultos de almacenamiento transportan información mediante cambios pequeños en los datos o por su propia presencia. Por ejemplo, un mensaje de error sobre la sintaxis de un nombre de usuario puede dar información a un pirata que intenta introducirse en el ordenador. Los canales ocultos de tiempo se aprovechan del reloj del sistema, o de algún otro dispositivo de temporización , utilizando información sobre el tiempo necesario para realizar una operación, el porcentaje de uso de UCP, o el tiempo que transcurre entre dos sucesos.

La utilidad para la administración de la fiabilidad implica asignar las actividades de seguridad a una persona distinta al administrador del sistema. Es obligatoria a partir de la clase B2, aunque la mayoría de los sistemas inferiores están organizados para cumplir este requisito.

Esta separación de tareas se basa en el principio de que es mejor asignar las actividades de seguridad a varias personas para que ninguna tenga el control total sobre los mecanismos de seguridad del sistema y pueda comprometerlo. Aquí también se aplica el concepto de privilegio mínimo para que los usuarios tengan el menor número de privilegios y durante el período más corto de tiempo para realizar su trabajo.

A partir de B 3· se requiere de una recuperación segura que garantice la inexistencia de brechas de seguridad cuando haya una caída del sistema. Esto se consigue realizando copias de seguridad periódicas de los ficheros críticos del sistema. Cuando haya una caída inesperada del sistema se aplicará un procedimiento de recuperación para garantizar la continuidad de su seguridad. Los requisitos de recuperación segura involucran tanto mecanismos de software como procedimientos administrativos tales como:

- Rearancar el sistema en modo mantenimiento para su reparación.
- Restaurar los sistemas de ficheros activos en el momento de la caída.
- Recuperar los ficheros dañados usando las copias de seguridad.
- Regenerar las etiquetas de seguridad de los ficheros alterados.
- Comprobar los ficheros que sean críticos para la seguridad.

5.4.2 Confiabilidad del ciclo de vida.

Estos requisitos aseguran que un sistema se haya diseñado, construido y mantenido conforme a un conjunto rígido de estándares formales.

Los requisitos de prueba de seguridad implican que el desarrollo haya probado todas las características de seguridad, garantizando que el sistema funciona tal y como se describe en los manuales , y que se han documentado los resultados de estas pruebas. Hay dos tipos de pruebas : de mecanismos y de interfaz. En las pruebas de mecanismo se comprueban los mecanismos de seguridad tales como el control de accesos discrecional/obligatorio, etiquetas, identificación, autenticación, auditoría, y vías fiables. Las pruebas de interfaz implican la comprobación de todas las rutinas de usuario que llaman a funciones de seguridad. Hasta la clase B1 es suficiente con probar los mecanismos, mientras que a partir de B2 también se requieren las pruebas de interfaz.

A partir de la clase B1 la especificación y verificación del diseño requiere una demostración matemática automatizada de cómo la descripción de un sistema es consistente con su modelo de seguridad. En esta demostración se comprueban matemáticamente las condiciones bajo las cuales determinados sujetos pueden acceder a ciertos tipos de objetos, y se prueba cómo los usuarios no pueden soslayar estas condiciones de acceso.

La gestión de configuración protege a un sistema fiable mientras se diseña, desarrolla y mantiene. Implica controlar todos los cambios realizados en la TCB (Trusted Computig Base), incluyendo hardware, firmware y software. Aunque tan sólo se requiera a partir de B2, también se recomienda para las clases inferiores. Mediante la gestión de configuración se mantiene un control del sistema durante su ciclo de vida asegurando que el sistema que se utiliza no es una versión antigua del

mismo. También es posible retroceder a una versión previa para resolver un problema que haya surgido en una etapa posterior. Para cumplir con los requisitos de configuración es necesario:

- Asignar un identificador único a cada elemento de la configuración (componentes hardware, programas, manuales) para realizar su seguimiento individual.
- Desarrollar un plan de gestión de configuración para decidir quién propone, evalúa y aprueba los cambios.
- Registrar todos los cambios producidos para verificar la configuración.
- Establecer un comité para el control de cambios para asegurar que tan sólo se desarrollen los cambios aprobados.

La distribución segura se aplica en la clase A1 e implica proteger el sistema mientras se envía a un cliente. Estos requisitos garantizan protección, es decir, que el sistema que recibe el cliente es idéntico a suministrado por el vendedor con las actualizaciones correspondientes incluidas, así como la validación por parte del cliente para detectar sistemas falsificados o modificados. Algunos de los métodos que se aplican son:

- Empaquetado contra ataques del entorno que garanticen el anonimato del emisor y del contenido.
- Mensajeros.
- Código para la autenticación de mensajes . se compra el código generado por el vendedor y el producido por el cliente. Si no coinciden es porque el contenido ha sido modificado.
- El cliente prueba en la instalación del suministrador el producto.
- El suministrador comunica al cliente lo que envía, cuándo llegará y por qué método.

5.5 Requisitos de documentación.

Los desarrolladores de sistemas seguros afirman que documentarlos lleva más tiempo que construirlos. Los requisitos de documentación se agrupan en:

- Guía de usuarios sobre características de seguridad.

- Manual para la administración de la seguridad.
- Documentación de pruebas.
- Documentación de diseño.

La guía de usuarios sobre características de seguridad está dirigida a los usuarios no privilegiados del sistema para indicarles todo lo que deben saber sobre la seguridad del sistema y su implantación. Incluye cuestiones como:

- Conexión a un sistema seguro y cambio de contraseña.
- Protección de los ficheros.
- Importación y exportación de ficheros.
- Restricciones que impone el sistema.

El manual para la administración de la seguridad está dirigido a administradores del sistema y/o de seguridad. Proporciona toda la información necesaria para establecer e implantar la seguridad del sistema. Contiene advertencias sobre las funciones y privilegios que deben controlarse en un sistema seguro.

La documentación de pruebas debe contener un plan de pruebas , hipótesis sobre el entorno de pruebas , procedimientos, resultados esperados y reales. El objetivo es encontrar, si existe, algún error en el diseño o la construcción de la Base Informática de Cómputo que permita a un usuario acceder a una información no autorizada.

La documentación de diseño permite conocer la construcción interna de la Base Informática de Cómputo, ayudando a los equipos de diseño y desarrollo a definir el modelo de seguridad del sistema y su construcción. Está constituida por la arquitectura del equipo físico y del sistema operativo, junto con una descripción de sus componentes principales (gestión de procesos, de ficheros, auditoría).

5.6 Características de las clases.

En la tabla 5.1 se resumen la aparición o mejora de requisitos de seguridad, mientras que en los siguientes apartados se resumen los aspectos más destacados de las clases indicadas en el Libro Naranja.

	C1	C2	B1	B2	B3	A1
Control de acceso discrecional	nue	mej	sin	sin	mej	sin
Reutilización de objetos	no	nue	sin	sin	sin	sin
Dispositivos mononivel/multinivel	no	no	nue	sin	sin	sin
Control de acceso obligatorio	no	no	nue	mej	sin	sin
Etiquetas	no	no	no	nue	sin	sin
Identificación y autenticación	nue	mej	mej	sin	sin	sin
Auditoria	no	nue	mej	mej	mej	sin
Vias fiables	no	no	no	nue	mej	sin
Arquitectura del sistema	nue	mej	mej	mej	mej	sin
Integridad del sistema	nue	sin	sin	sin	sin	sin
Pruebas de seguridad	nue	mej	mej	mej	mej	mej
Especificación y verificación del diseño.	no	no	nue	mej	mej	mej
Canales ocultos	no	no	no	nue	mej	mej
Facilidad para la administración de la fiabilidad.	no	no	no	nue	mej	sin
Gestión de configuración	no	no	no	nue	sin	mej
Recuperación segura	no	no	no	no	nue	sin
Distribución segura	no	no	no	no	no	nue
Guías de usuario sobre las características de seguridad.	nue	sin	sin	sin	sin	sin
Manual para la administración de la seguridad.	nue	mej	mej	mej	mej	sin
Documentación de prueba	nue	sin	sin	mej	sin	mej
Documentación de diseño	nue	sin	mej	mej	mej	mej

Leyenda:

no: no existe criterio en esta clase

nue: criterio nuevo para esta clase

mej: nuevos requisitos para el criterio en esta clase
sin: no existen requisitos adicionales para el criterio en esta clase.

TABLA 5.1 Evolución de los requisitos de los criterios del Libro Naranja

5.6.1 D: seguridad mínima.

Está reservada para los sistemas que se han valorado y no cumplen los requisitos para la evaluación en una clase superior. No hay, pues, requisitos en la división D, tan sólo es un cajón de sastre para los sistemas que no poseen las características de seguridad de las clases superiores.

De hecho, no se ha evaluado ningún sistema en esta clase, pues ningún vendedor quiere asumir el coste elevado de valorar un sistema que no posea un conjunto razonable de características de seguridad. Por ejemplo los ordenadores bajo MS-DOS, Macintosh o Amiga estarían en esta categoría si fueran evaluados.

5.6.2 C1: Protección mediante seguridad discrecional

En esta categoría todos los usuarios manejan datos al mismo nivel de seguridad. Las características de seguridad están orientadas a evitar que los usuarios cometan errores involuntarios que puedan dañar al sistema como, por ejemplo, escribiendo en su memoria o sobre una aplicación crítica, o puedan interferir en el trabajo de otros usuarios borrando o modificando sus programas o datos. En esta clase no hay mecanismo suficientes para mantener a un intruso fuera del sistema.

Las características más importantes de C1 son:

- Contraseña. Identificación (nombre de conexión) y palabra clave para identificar y autenticar a un usuario antes de permitirle entrar en el sistema.

- Protección discrecional de los ficheros y otros objetos. Mediante este mecanismo el dueño de un objeto puede decidir cómo protegerlo durante los accesos al mismo.

Los sistemas C1 deben poseer una arquitectura capaz de proteger al código del sistema de ataques procedentes de los programas de usuario. En este sentido, un proceso bajo UNIX no puede salirse de su espacio virtual de direcciones, y si lo intenta el sistema operativo le envía una señal que provoca su muerte. El sistema debe probarse para garantizar que funciona correctamente y que las características de seguridad no pueden soslayarse de una forma evidente.

Casi todas las instalaciones UNIX previas a System V Release 4 (1991) están en la clase C1, es decir, más del 90% de la base mundial instalada. En la actualidad se considera que C1 no es lo suficientemente seguro para entornos comerciales. A modo de ejemplo, la administración estadounidense acuñó un lema :" C2 para el 92", para motivar la emigración anterior.

5.6.3 C2 : protección mediante accesos controlados.

Los sistemas C2 añaden las siguientes características respecto a la clase anterior:

- **Responsabilidad** de los usuarios individuales. Esto se consigue mediante un mecanismo de auditoría que almacena un registro de cada acción realizada por el usuario, lo que permite conocer exactamente qué ha hecho el sistema.
- **Controles discrecionales** más detallados capaces de aceptar o denegar el acceso a un usuario utilizando un mecanismo similar al de las listas de control de acceso.
- **Reutilización de objetos** que garantice que cualquier objeto que un proceso haya dejado en un soporte de almacenamiento (memoria, disco) no será accesible accidentalmente a otro usuario.

La arquitectura de un sistema C2 debe permitir que los recursos del sistema se protejan mediante accesos controlados . Por ejemplo, en UNIX el acceso a los periféricos sigue un esquema de permisos idénticos al de los ficheros de los usuarios.

La mayor parte del UNIX que se vende comercialmente en la actualidad corresponde a esta clase: Hewlett-Packard HP-UX, IBM AIX, Santa Cruz Operation -SCO UNIX , Sun Solaris, puesto que lo único que han tenido que añadir estos fabricantes es un paquete de auditoría que suele proporcionarse de forma opcional.

5.6.4 B1: protección mediante seguridad etiquetada.

A partir del nivel B1, los sistemas poseen un sistema de control de acceso obligatorio que implica colocar una etiqueta a cualquier objeto (en particular sobre los ficheros). El sistema utiliza estas etiquetas de confidencialidad junto con el nivel de habilitación de los usuarios para reforzar la política de seguridad del sistema. En estos sistemas la protección no quedan a discreción del dueño de un objeto, puesto que no se permite el acceso al mismo salvo que se disponga de la habilitación necesaria.

Los sistemas B1 deben poseer una arquitectura que separe los componentes relacionados con la seguridad de los que no lo están . Debe existir una documentación que incluya el modelo de seguridad soportado por el sistema. No es necesaria una demostración matemática, pero si una exposición de las reglas implantadas por las características de seguridad del sistema.

Entre los sistemas UNIX que están calificados como B1 figuran AT&T System V/MLS y OSF/1

5.6.5 B2 : Protección estructurada.

A partir de la clase B2 no se añaden nuevas características de seguridad que sean visibles desde el punto de vista del usuario respecto a B1. Por el contrario, se profundiza en la confiabilidad adicional que debe proporcionarse sobre su diseño y funcionamiento correcto.

En B2 todos los objetos del sistema están etiquetados , incluidos los dispositivos. Existen vías fiables que garantizan la comunicación segura entre el usuario y el sistema. También soportan el concepto de privilegio mínimo, que asigna tanto a los usuarios como a los programas el mínimo número de privilegios, y durante el intervalo de tiempo más breve, para que puedan realizar las funciones del sistema.

Desde el punto de vista del diseño los sistemas deben ser modulares y utilizar componentes físicos para aislar las funciones relacionadas con la seguridad de las que no la están. Requiere una declaración formal del modelo de seguridad del sistema, y que haya una gestión de la configuración para controlar los cambios en el código del sistema y en la documentación. También deben buscarse los canales ocultos para evitar que un usuario pueda usarlos para revelar información.

Trusted XENIX y USL SVR4 Enhanced Security son dos UNIX que están siendo evaluados para obtener la clase B2.

5.6.6 B3 : dominios de seguridad

En este nivel, el diseño del sistema y las características de confiabilidad son muy rigurosas. Es necesario que exista un administrador de seguridad, que sea alertado automáticamente si se detecta una violación inminente de la seguridad.

Deben existir procedimientos para garantizar que la seguridad se mantiene aunque el ordenador se caiga y luego rearranque. Es obligatorio la existencia de un monitor de referencias sencillo, a prueba de agresiones e imposible de soslayar. La TCB debe excluir todo el código fuente que no sea necesario para proteger el sistema.

Los sistemas B3 no son habituales , pues al ser tan difícil obtener esta certificación los fabricantes prefieren intentar directamente A1.

5.6.7 A1: diseño verificado.

Hasta el momento ésta es la clase de certificación más alta, aunque el Libro Naranja no descarta la posibilidad de definir requisitos que la excedan en aspectos tales como arquitectura del sistema, pruebas y verificación formal.

Los sistemas A1 son funcionalmente equivalentes a B4. Tan sólo se añade la distribución fiable que refuerza la seguridad mientras que se transporta un sistema a un cliente. Lo que diferencia a un sistema A1 es la confiabilidad adicional que ofrece el análisis formal y la demostración matemática de que el diseño del sistema cumple el modelo de seguridad y sus especificaciones de diseño.

CAPITULO 6

Prácticas de seguridad para los usuarios de sistemas seguros.

6.1 Uso de sistemas C2.
6.2 Conexión.
6.3 Uso de los comandos mediante privilegios.
6.4 Utilización de dominios protegidos
6.5 Auditoría.
6.6 Cifrado de la información.
6.7 Recomendaciones para el mantenimiento seguro de una cuenta.

En este capítulo se presenta la información sobre la seguridad que todos los usuarios de un sistema deben conocer. Un sistema UNIX, incluso con la incorporación de características adicionales de seguridad, requiere que los usuarios apliquen algunas prácticas de seguridad sencillas tales como:

- Uso de buenas contraseñas.
- Permisos adecuados para los ficheros.
- Inicialización de las características elementales de seguridad mediante los ficheros de configuración.
- Implicaciones de los comandos sobre la seguridad.
- Cifrado de información sensible.

6.1 Uso de sistemas C2.

Pese a que en 1983 ya existía el Libro Naranja, no ha sido hasta principios de los 90 cuando se ha notado comercialmente su influencia. En 1991, la empresa multinacional de las telecomunicaciones AT&T , que es la propietaria de UNIX, publicó la versión UNIX System V Release 4, que incorporó de forma estándar seguridad C2. El problema es que el más del 90% de la base instalada en UNIX corresponde a la Release 3, o incluso es anterior, y estos sistemas suelen tener nivel de seguridad C1 , que actualmente se considera insuficiente. Pese al tiempo transcurrido, aún quedan muchos fabricantes de ordenadores que no se han actualizado a la Release 4, por lo que continúan vendiendo sistemas con una seguridad insuficiente . Esta situación suelen resolverla los fabricantes vendiendo un paquete adicional que soporta seguridad C2 o superior, pero como este software tiene un coste extra, pocas son las empresas que suelen adquirirlo, por lo que sus ordenadores acaban siendo potencialmente inseguros. Por lo cual es más de lamentar en tanto en cuanto hoy en día pueden adquirirse seguridad C2 con un pequeño coste adicional.

En este capítulo se ha elegido UNIX Santa Cruz Operation (SCO) System V como representante de un UNIX C2 por los siguientes motivos:

- Una de cada cuatro instalaciones UNIX del mundo utiliza SCO.
- A partir de UNIX SCO System V Release 3.2 el sistema operativo incorpora de forma estándar (sin pagar un costo adicional) seguridad C2.
- La seguridad C2 de SCO ha sido desarrollada por la compañía SecureWare , cuyos productos han sido aceptados como desarrollo estándar " de facto" en seguridad informática acogida a los criterios del Libro Naranja.

6.2 Conexión.

En este apartado se describe cómo conectarse a un sistema C2, cambiar la contraseña y utilizar otra cuenta.

Al conectarse aparecen los mensajes:

login: nombre de usuario
password: contraseña (sin eco)

Tras escribir correctamente la contraseña aparecen las últimas fechas de conexión correcta e incorrecta:

Last successful login for usuario : fecha y hora
Last unsuccessful login for usuario : fecha y hora

Si las fechas y horas no coinciden con las acciones del usuario, éste debe consultar al administrador inmediatamente, puesto que alguien podría haber utilizado su cuenta.

Cuando el usuario no puede conectarse es posible que ocurra alguna de las circunstancias siguientes:

- El administrador de seguridad ha establecido una vigencia a su contraseña, y ésta ya ha expirado. Hay que solicitar al administrador una nueva contraseña y la reapertura de su cuenta.
- El administrador ha establecido un límite al número de conexiones incorrectas que pueden realizarse sobre su cuenta (terminal). Cuando se excede este límite la cuenta se bloquea automáticamente. Es necesario solicitar al administrador la reapertura de su cuenta . Si el usuario considera que ha realizado el proceso de conexión correctamente, debe indicárselo al administrador , es posible que alguien haya interferido el sistema.
- El administrador de seguridad ha bloqueado la cuenta o el terminal. Para continuar trabajando hay que solicitar al administrador la reapertura de la cuenta.
- El administrador de seguridad decide si cada usuario puede cambiar su propia contraseña mediante el comando de usuario passwd , e incluso establece un periodo mínimo entre dos cambios consecutivos de la contraseña. En el primer caso, al utilizar el comando passwd aparece el mensaje:

  Password cannot be changed. Reason : Not allowed to execute password for the given user.

Cuando un usuario pueda usar el comando passwd , éste solicitará la contraseña actual:

Old *password:*

Si es correcta se muestra la fecha y hora del último cambio en la contraseña:

Last successful password change for usuario: Fecha y hora.
Last unsuccessful password change for usuario: Fecha y hora

Es conveniente comprobar que estos mensajes reflejan los últimos cambios. Si no es así, hay que avisar al administrador inmediatamente.

A continuación aparece el siguiente mensaje:

You can choose wether you pick your own password
or have the system create one for you.

1. Pick your own password.
2. Pronounceable password will be generated for you
Enter choice (default is 1):

Si se elige la opción 1 el comando solicitará la nueva contraseña junto con una confirmación. Si se elige la opción 2 el sistema genera una contraseña:

Generating random pronounceable password for usuario.
The password, along with the hypenated version, is shown.
Hit<Return> or <Enter> until you like the choice.
When you have choosen the password you want, type it in.
Note: Type your iterrupt character or quit to abort at any time.
Password: xxxxxxxxxx Hypenation: xxx-xxx-xxx-x
Enter password:

Algunos ejemplos de contraseñas generadas son :

akiggalics	ak-igg-al-ics
rhaloquejo	rhal-o-quej-o
areojimcau	ar-e-oj-mi-cau

La contraseña generada se presenta en una versión silabizada para ayudar en su memorización. Jamás debe escribirse en un papel la contraseña . Si no se cambia la contraseña, debe escribirse la palabra quit o pulsarse la tecla para interrumpir programas (habitualmente en Supr), con lo que aparecerá el mensaje:

password request denied. Reason: user stopped program.

6.3 Uso de los comandos mediante privilegios.

En un sistema C2 se restringe el uso de los comandos, por lo que algunos tan sólo pueden utilizarse si el administrador de seguridad proporciona el **privilegio** correspondiente.

Los mecanismos de seguridad proporcionan dos tipos de privilegios: núcleo y subsistemas. Un privilegio de núcleo permite ejecutar procesos específicos que afectan al sistema operativo. Un privilegio de subsistema permite utilizar los comandos de un determinado subsistema protegido, que es una conexión de ficheros , dispositivos y comandos que protegen un conjunto de recursos o realizan tareas de seguridad.

Los privilegios de núcleo son los siguientes:

- Privilegios para ejecutar programas con atributos SUID, lo que facilita al programa que lo posee el acceso a todos los ficheros, procesos y recursos que pertenecen a la persona que ejecuta el programa o al dueño del fichero ejecutable.
- Privilegios para definir el comportamiento de un programa con el atributo SUID. Si existe este privilegio, los programas con el atributo SUID se ejecutan como en UNIX C1. De lo contrario, se crea un **dominio protegido** dentro del cual los programas tienen menos posibilidades para corromper o robar datos privados.
- Privilegios para cambiar los atributos SUID y SGID de un fichero ordinario o directorio utilizando el comando de usuario chmod.
- Privilegios para cambiar el dueño de los ficheros utilizando el comando de usuario chown.

Existen dos niveles de privilegios de subsistema: primario y secundario. Un **privilegio primario de subsistema** permite utilizar los comandos de un subsistema protegido como administrador:

- privilegios para utilizar el comando de usuarios ps para comprobar el estado de los procesos de otros usuarios , y el comando de usuario ipcs para informar sobre el estado de los recursos para la comunicación entre procesos (tuberías, colas de mensajes, semáforos, memoria compartida y "sockets"). Sin este privilegio los comandos tan sólo proporcionan información correspondiente a los procesos propios.

- Privilegios para utilizar el comando de usuario write para comunicarse con otros usuarios. Si se utiliza este comando sin este privilegio, todos los códigos de control y las secuencias de escape del mensaje se convierten a caracteres ASCII.

Un privilegio **secundario de subsistema** permite utilizar los comandos de un subsistema como un usuario normal, es decir, sin privilegio administrativo alguno:

- Privilegios para ver las peticiones de otros usuarios en la cola de la impresora.
- Privilegios para utilizar los comandos de usuario enable y disable para cambiar el estado de las impresoras.
- Privilegios para utilizar el comando de usuario df para consultar el espacio libre disponible en disco.

6.4 Utilización de dominios protegidos.

Los **dominios protegidos** permiten controlar el daño que pueden realizar un programa con el atributo SUID sobre los ficheros. Cuando un programa con este atributo comienza su ejecución e identificador de usuario efectivo es igual al dueño del fichero, mientras que el identificador de usuario real corresponde al usuario llamador del programa. En UNIX C1 un programa con el atributo SUID tiene acceso a todos los ficheros, procesos y objetos para la comunicación entre procesos a los que el llamador o el dueño tiene acceso, porque el programa puede utilizar la llamada al sistema setuid() para conmutar entre el identificador del llamador y el del dueño. Fuera de un dominio protegido , este poder se restringe a los recursos a los que tanto el llamador como el dueño tienen acceso.

En UNIX C1 el atributo SUID se aplica cuando un usuario (o función del sistema) necesita proteger el acceso a ciertos ficheros, salvo a través de un conjunto bien definido de programas. Por ejemplo, el conjunto de comandos para manejar impresoras debe manipular ficheros de configuración, de estado y ficheros de comandos para controlar cuáles son las peticiones encoladas en la impresora.

Tanto usuarios como administradores utilizan comandos para enviar y cancelar peticiones, cambiar y consultar el estado de la impresora, y añadir o quitar impresoras del sistema. Todos los ficheros de las impresoras, pertenecen al pseudousuario lp, cuyo identificador de usuario es dueño de todos los ficheros que utiliza el subsistema de impresoras, incluyendo los ficheros especiales que representan a los dispositivos de impresión. Al llamar al comando de usuario lp para imprimir un fichero, el programa puede acceder a los ficheros que gestionan las impresoras, pero también accede a los ficheros que se solicitan imprimir, puesto que lp ajusta su identificador de usuario al llamador. Un comando de usuario lp malicioso podría acceder a los ficheros protegidos de una cuenta y copiarlos en un lugar al que tan sólo tuviera acceso este programa. Cuando se ejecuta un programa con el atributo SUID y no existe el privilegio de núcleo nopromain, se crea un dominio protegido, convirtiéndose el directorio actual en la **raíz del dominio protegido** (" promain root"). Los ficheros que hay dentro de este subárbol están dentro del dominio, mientras que los demás están fuera. Los dominios protegidos aseguran al usuario contra programas SUID maliciosos, restringiendo el tipo de acceso que pueden hacer fuera del dominio cuando los ejecuta un usuario llamador.

Cuando el programa se ejecuta con el identificador efectivo del usuario llamador fuera del dominio, sólo puede acceder a los ficheros a los que tanto el llamador como el dueño tengan acceso, es decir, ficheros públicos. Dentro del dominio el programa tiene acceso según las reglas habituales .

6.5 Auditoría.

En la mayoría de los ordenadores bajo UNIX C1 la única información que se registra sobre los programas ejecutados es la correspondiente a la contabilidad que proporciona el comando acct, que sirve para facturar los recursos consumidos . Desafortunadamente esta información puede ser manipulada y no es suficiente para determinar las acciones que ha realizado un intruso. Por otra parte, esta información o no se registra, o se aplica sobre todos los usuarios, pero como en este caso consume demasiado recursos de procesador y de disco, finalmente se termina desinstalando. .

A partir de UNIX C2 se ha añadido un sistema de **auditoría** que no sólo registra los sucesos que se producen en el sistema, sino que además almacena sus argumentos y resultados. El administrador puede configurar su granularidad a los niveles de sucesos y de usuario.

La mayoría de los sucesos se asocian a las llamadas del núcleo del sistema operativo que realizan los programas durante su ejecución : open ( ) , creat( ) , fork( ), exec ( ) , read ( ) , write ( ) , etc. Éstas pueden agruparse en tipos, permitiendo que su activación se realice mediante combinaciones sobre ellas. Cuando la auditoría está activada, siempre se auditan unos sucesos fijos que son críticos para la integridad del sistema. El resto de los sucesos son seleccionables.

Cada suceso auditable genera un registro compuesto por marca de fecha y hora, identidad del usuario, nombre del objeto, identificadores del proceso causante del suceso, identificación del tipo de suceso, y una identificación sobre su éxito o fracaso.

6.6 Cifrado de la información.

Para cifra información UNIX dispone del comando de usuario crypt, así como de una opción -x en sus tres editores (de, ex, y vi). Desafortunadamente, nada de esto está disponible fuera de los Estados Unidos de América. Una ley de este país, prohibe exportar tecnología sofisticada de cifrado, y actualmente se considera que el algoritmo DES (Data Encryption Standard) que utiliza UNIX para realizar esta operación es suficiente para que no sea exportable. De hecho, en Estados Unidos tampoco se vende directamente , sino que cuando un cliente compra una instalación UNIX debe solicitar posteriormente a su proveedor la incorporación de los comandos de cifrado correspondientes.

El único comando que está disponible para cifrar información en UNIX fuera de Estados Unidos es /usr/lib/makeykey . Este programa recibe desde el teclado una cadena de ocho caracteres , seguida por otra de dos que constituye la semilla, y devuelve por la pantalla la semilla junto con la cadena cifrada mediante DES.

$ / usr/lib/makeykey

juanito1BP cadena de ocho caracteres y 2 de semilla

BPd6fDVft2tRw$ devuelve semilla y cadena cifrada. Observa que makeykey no añade un salto de linea, por lo que la impronta del intérprete de comandos aparece a continuación.

$ grep juan /etc/passwd ahora puede compararse con nuestra contraseña cifrada

juan: BPd6fDVft2tRw:.....

$

Si un usuario necesita programa para cifrar información tiene dos alternativas, o se lo compra a su proveedor, o bien obtiene uno de dominio público como IDEA o PGP.

6.7 Recomendaciones para el mantenimiento seguro de una cuenta.

Los usuarios de un ordenador constituyen su seguridad. En este apartado se proporciona un conjunto de consejos útiles para mantenerla.

6.7.1 Conexión y desconexión.

- Teclear cuidadosamente la contraseña.
- Si tras escribir la contraseña el sistema notifica un error, y se cree haberla escrito correctamente, avisar al administrador de seguridad inmediatamente. Si es posible, debe entrarse en la cuenta para comprobar la última fecha de conexión incorrecta contra la hora actual para detectar si algún programa ha capturado la contraseña. Como medida de precaución, si es posible, hay que cambiar la contraseña.
- Al conectarse , comprobar que la fecha de última conexión y desconexión son correctas. Buscar intentos de conexión realizados fueras de las horas habituales de

trabajo. Informar inmediatamente al administrador de seguridad sobre las discrepancias.

- Nunca debe abandonarse la terminal con una sesión de trabajo abierta. Si se desea hacerlo, es conveniente bloquear la terminal con un programa que evite su manipulación.
- Antes de conectarse, pulsar las teclas CTRL d (fin de conexión) o CTRL c (terminación de programa) para forzar a que el sistema genera un nuevo programa de conexión. Con esto se evita los Caballos de Troya que pueden estar ejecutándose en la terminal.
- Utilizar el comando exit (o pulsar las teclas CTRL d) para terminar la sesión de trabajo. Esperar a ver el mensaje login: antes de apagar la terminal.

6.7.2 Seguridad de la contraseña.

Son numerosos los consejos que pueden ofrecer en el uso de las contraseñas, siendo todas ellas de sentido común. Un estudio exhaustivo sobre la seguridad de la contraseñas puede encontrarse en el " Libro Verde". Seguramente se han recogido algunas de las recomendaciones habituales.

- Que tenga una longitud, al menos de ocho caracteres, siendo éstos letras, dígitos o símbolos de puntuación.
- No empezar o terminar la contraseña con un solo dígito. Por ejemplo, *maria8.*
- No usar palabras fáciles de adivinar. No debe ser una palabra de diccionario, nombre, o apodo. No usar fechas de nacimiento, números de teléfonos o de direcciones.
- No utilizar palabras al revés.
- Usar diferentes contraseñas en distintos ordenadores. Procurar que la contraseña no refleje el nombre de los ordenadores.
- Mantener la contraseña secreta. No escribirla, ni compartirla con otra persona, ni decirla, ni enviarla por correo electrónico.
- No almacenarla en las teclas de función del terminal o en la memoria del módem.

- No permitir que otras personas puedan verla al teclearla. Si se sospecha que alguien ha podido adivinarla, lo mejor es modificarla.
- Cambiarla periódicamente . Como mínimo una vez cada medio año, aunque es mejor hacerlo mensualmente.
- No cambiar entre dos contraseñas predefinidas, ni reutilizar contraseñas antiguas .
- Si la terminal que utiliza tiene memoria propia, controlar que otros usuarios no envíen secuencias de control que permitan registrar nuestras actividades, tales como averiguar nuestra contraseña.

6.7.3 Seguridad de los archivos.

- No permitir que los archivos y directorios de un usuario sean modificables por cualquier otro.
  - Establecer una máscara 002 como argumentos del comando umask.
  - Si no hay confianza con los componentes de nuestro grupo colocar la máscara 022.
  - Asegurarse de que el archivo .profile tan sólo es legible y modificable por nuestra cuenta.
  - Asegurar de que los directorios de trabajo no es modificable por otros usuarios.

- Si no se desea que otros usuarios puedan leer nuestros archivos o echar un vistazo al contenido de nuestros directorios, hacer que no sean legibles por nadie.
- Establecer la máscara a 006 (007 si se desea proteger al directorio por completo).
- Si no se desea que los usuarios de nuestros grupos accedan a nuestros archivos o directorios colocar la máscara a 066 ó 077.
- Los archivos temporales con información sensible no deben ser legibles por lo que debe utilizarse la máscara 077.
- Asegurarse de que el directorio de trabajo no es legible por otros usuarios.
- Cifrar aquellos archivos que contengan información sensible para garantizar que , salvo el propietario, nadie pueda leerlos.

6.7.4 Ejecución de programas.

- No escribir programas con el atributo SUID/SGID. Si es imprescindible tenerlos , utilizar una lista de control para mantenerlos adecuadamente.
- Ser cuidadoso al copiar o renombrar archivos.
  - Cuando se copie un archivo mediante el comando de usuario cp, hay que recordar que los permisos del archivo destino serán idénticos a los del origen , incluyendo los atributos SUID/SGID, y que el dueño y el grupo del archivo se ajustará al UID y GID efectivo, respectivamente. Si el archivo destino ya existe, entonces sus permisos y dueño no cambiarán.
  - Cuando se renombren archivos utilizando el comando de usuario mv los permisos del archivo destino serán iguales que los del origen, incluyendo los permisos SUID/SGID. Si el renombrado se produce dentro del mismo sistema de archivos no cambiará ni el dueño ni el grupo. Si no es así , se aplicará el UID y GID efectivo.
  - Usar el comando de usuario cpio cuidadosamente al restaurar copias de seguridad. Puede destruir archivos que no se encuentran en la estructura del directorio actual de trabajo. Utilizar la opción t para obtener un listado de los archivos que van a transferirse.

- Antes de borrar un programa que tenga los atributos SUID/SGID comprobar cuántos sinónimos tiene. Si hay más de uno, quitar todos los permisos (chmod 000) , borrarlo o vaciarlo, y luego eliminarlo. Averiguar el número índice mediante el comando de usuario ls -li y dárselo al administrador para que busque los otros sinónimos.
- No ejecutar los programas de otros usuarios, salvo que hayan sido comprobados.
- Para evitar caballos de Troya, en la variable de entorno PATH deben colocarse en primer lugar los directorios del sistema , luego los locales y por último el directorio actual.

CAPITULO 7

MANTENIMIENTO DE SISTEMAS SEGUROS

7.1 Diferencias entre sistemas seguros e inseguros.
7.2 Funcionamiento de un sistema seguro.
7.3 Protección de datos.
7.4 Detección de intrusiones en el sistema.
7.5 Tratamiento de sistemas de archivos corruptos.
7.6 Auditoría.
7.7 Recomendaciones para el mantenimiento seguro de un sistema.

Cada sistema informático necesita protegerse de las personas que acceden de una manera no autorizada a los recursos del ordenador. Las características de seguridad presentes a partir de UNIX SCO System V Release 3.2 están diseñadas para cumplir los requisitos del nivel C2 del Libro Naranja. En este capítulo se explica cómo utilizar las características de seguridad para mantener un sistema seguro: protección de datos, detección de intrusiones, y tratamiento de sistemas de archivos corruptos.

Las características de seguridad del sistema operativo no son útiles si no se protege ni el equipo físico ni los medios para el intercambio de datos. Es necesario proteger el ordenador, los disquetes originales de las aplicaciones y las copias de seguridad frente a un acceso no autorizado, aplicando las siguientes reglas:

- Cuando el operador no esté presente, cerrar con llave la terminal, y la habitación en la que se encuentre el sistema.
- Proteger todas las copias de seguridad.
- Proteger las líneas de comunicaciones (TCP/IP, UUCP, Ethernet y terminales) frente a un acceso no autorizado.

7.1. Diferencias entre sistemas seguros e inseguros.

No existe ningún sistema informático que esté totalmente libre de riesgos. Un sistema seguro es aquel que alcanza un elevado nivel de control sobre el acceso a la información, proporcionando mecanismos para prevenir (o al menos detectar) los accesos no autorizados, así como la posibilidad de confirmar que esos mecanismos funcionan correctamente.

Las acciones del administrador son cruciales para el mantenimiento de un sistema seguro. Cualquier desliz en el estado de la seguridad invita a la entrada en el sistema. Para ser un administrador eficaz es necesario comprender la política de seguridad del sistema, cómo se controla la información del sistema operativo, y cómo afectan los cambios que realiza el administrador a sus propias acciones y a las de los usuarios.

7.1.1 Base segura de cómputo.

La base segura de cómputo (TCB) mantiene la parte del sistema relativa a su seguridad. Está formada por el núcleo del sistema operativo y las utilidades seguras que referencian y mantienen los datos relevantes a la seguridad. La TCB implementa la política de seguridad del sistema, que es el conjunto de reglas para inspeccionar y vigilar las interacciones entre sujetos (procesos) y objetos (archivos y mecanismos para la comunicación entre procesos tales como tuberías, colas de mensajes, semáforos, memoria compartida y "sockets").

En el nivel C2 la política de seguridad consiste en:

- Control de accesos discrecional, que permite a cada usuario señalar que otros usuarios, y con que derechos, pueden acceder a los objetos creados por él.
- Reutilización de objetos (residuos), que indica cómo debe inicializarse o desinstalarse un objeto de almacenamiento antes de reasignarse.

7.1.2 Identificación y autentificación.

UNIX C1 posee mecanismos limitados para realizar la identificación y autentificación de los usuarios. En primer lugar, el sistema busca en el archivo /etc/passwd el nombre de conexión para identificar al usuario. Si lo encuentra, lo autentifica cifrando la contraseña introducida concatenada con dos caracteres, y comparando el valor obtenido con el almacenado en el archivo citado.

Además, suelen añadirse algunas reglas para establecer las características que debe poseer la contraseña y sus posibilidades de cambio. Sin embargo, estas últimas reglas se han mostrado insuficientes para proteger contra las intrusiones. En UNIX C2 se amplían los mecanismos de identificación y autentificación:

- Hay más reglas que implantan los tipos de contraseñas que pueden establecerse.

- Existen nuevos procedimientos para generar y modificar las contraseñas.
- Se ha modificado la ubicación y protección de ciertas partes de la base de datos de contraseñas.
- El administrador tiene mayor control sobre el proceso de conexión.
- Existe una función separada que se denomina administrador de autentificación o dos cuentas, que está soportada por el privilegio de subsistema auth para realizar el mantenimiento de este aspecto del sistema.

7.1.3 Control de accesos discrecional.

El control de accesos discrecional determina si un proceso de usuario puede acceder a la información que hay dentro de un objeto. En UNIX C1 la protección de los objetos se implanta mediante la relación entre el dueño y el grupo de un proceso, y los permisos a nivel de dueño, grupo y otros del objeto. Los atributos que representan la protección de los objetos se establecen a discreción de su dueño, quien puede cambiarlos, e incluso ceder sus derechos de acceso a otro usuario. En UNIX C2 se amplian las reglas para realizar el control de accesos discrecional de la siguiente manera:

- Capacidad para establecer el atributo que permite reasignar el identificador de usuario (SUID) en los archivos ejecutables.
- Capacidad para cambiar el dueño de los archivos mediante el comando de usuario chown.
- Control del abuso potencial de los permisos SUID, SGID y "sticky", eliminándolos siempre que se escriba sobre los archivos que lo poseen.

7.1.4 Privilegios

Un privilegio es un atributo de usuario necesario para realizar determinadas acciones. En UNIX C1 todas las decisiones de acceso se basan en los permisos de los archivos, o en si el proceso que realiza el acceso pertenece al superusuario ("root"). La cuenta root es la única que puede realizar ciertas

operaciones sobre el sistema.En UNIX C2 se definen dos tipos de privilegios: de núcleo y de subsistema. Los privilegios de núcleo se almacenan en un conjunto de privilegios asociado a cada proceso, y es una lista de autorizaciones que permiten realizar un tipo de acción sólo si existe el privilegio correspondiente. Por defecto, se establecen unos privilegios para el sistema, aunque pueden definirse para un usuario específico. Los privilegios de subsistema se asocian a usuarios, permitiéndoles realizar una acción especial a través de los comandos (utilidades y programas seguros) de un subsistema protegido, que es una colección de archivos, dispositivos y comandos para la realización de una función particular.

7.1.5 Auditoría.

En UNIX C1 se mantiene un registro limitado de las acciones del sistema a través de la contabilidad soportada por el comando de mantenimiento acct. Este programa escribe un registro contable único tras la conclusión de cada proceso. Por el contrario, en UNIX C2 se proporciona una pista de auditoría donde se registra cada acceso (satisfactorio o no) de un sujeto a un objeto, y cada cambio en las características del sujeto, objeto y sistema. El subsistema de auditoría se controla a través de una función específica denominada administrador de auditoría soportada mediante un privilegio de subsistema. Este administrador decide que información se registra y con cuánto nivel de confianza, junto con el mantenimiento de la información recogida. El subsistema de auditoría proporciona a su administrador una historia detallada sobre las acciones del sistema, que le ayudan a identificar que ha sucedido, cuándo ocurrió, y quién estuvo involucrado.

7.1.6 Subsistemas protegidos.

UNIX C1 proporciona los mecanismos para la reasignación de los identificadores de usuario y de grupo a través de las llamadas al sistema setuid ( ) y setgid ( ) para construir programas que deben mantener confidencial su información, de forma que ésta tan sólo sea accesible o modificable a través

de las operaciones construidas por estos programas. Pero en UNIX C2 existe el concepto de subsistema protegido, que es una colección de información confidencial, junto con sus dispositivos, utilidades y comandos para mantener dicha información. Los subsistemas protegidos utilizan los mecanismos SUID y SGID para proteger sus recursos confidenciales frente a accesos no restringidos, lo cual conlleva:

- Proporcionar un control más preciso de los usuarios y grupos que mantienen determinadas colecciones de recursos del sistema.
- Mantener una base de datos separada de los usuarios que pueden ejecutar programas para mantener la información confidencial.
- No es necesario que los usuarios se conecten como administradores de subsistemas, sino que se utiliza esta base de datos para comprobar el privilegio de subsistema.

7.2.Funcionamiento de un sistema seguro.

Las primeras decisiones que deben tomarse en un sistema seguro son:

- Asignación de una persona o un grupo para administrar el sistema.
- Asignación de privilegios de núcleo a las personas que necesiten privilegios adicionales.
- Decidir el grado de control y monitorización de accesos al sistema.

7.2.1 Asignación de las funciones administrativas.

La primera elección que debe realizarse es la persona o conjunto de personas que mantendrán el sistema. Puede elegirse un único superusuario como root, o pueden asignarse fragmentos de las funciones administrativas a varios usuarios, concediendo a cada uno de ellos estrictamente el poder necesario para administrar un aspecto del funcionamiento del sistema. Para poder realizar las actividades asociadas a una función administrativa, es necesario que el administrador posea el privilegio

de subsistema apropiado, que se muestra en la tabla 1, en la que se asocia a cada función administrativa su privilegio de subsistema correspondiente, junto con el área del sistema que mantiene cada función.

Función administrativa	**Area de actividad**
Administrador de autentificación	Cuentas del sistema
Administrador de impresoras	Subsistema de impresoras
Administrador de la ejecución temporizada de programas (cron)	Subsistema at y cron
Administrador de auditoría	Bases de datos y registro de auditoría
Operador	Copias de seguridad de los sistemas de archivos
Administrador del sistema	Acceso a la cuenta root a través del comando su

Tabla 1. Funciones administrativas en un sistema C2.

Es fundamental comprender las responsabilidades de cada función y el impacto de cada una de sus acciones sobre la seguridad del sistema.

7.2.2 Asignación de los privilegios de núcleo.

En la tabla 2 se muestran cuáles son los privilegios de núcleo existentes:

Privilegios de núcleo
Capacidad para ejecutar programas con el atributo SUID
Capacidad para establecer los atributos SUID y SGID en archivos
Capacidad para cambiar el dueño de un objeto
Capacidad para modificar los parámetros de auditoría
Capacidad para suspender la auditoría de un proceso
Capacidad para escribir registros en la pista de auditoría

Tabla 2. Privilegios de núcleo en un sistema C2

La mayoría de los usuarios tan sólo necesitan el privilegio para ejecutar programas con el atributo SUID, pero si también necesitan crear archivos con este atributo deberán poseer el privilegio correspondiente. Para ceder los derechos de acceso de un archivo es necesario cambiar su propietario, para lo cual se requiere otro privilegio. Si no existe, tan sólo root puede cambiar de dueño a un archivo. Los privilegios de núcleo para la auditoría nunca deben asignarse a otro usuario que no sea el administrador de la auditoría, puesto que están pensados para que los use un programa diseñado como aplicación segura.

7.2.3 Control de accesos al sistema.

Uno de los aspectos importantes para el funcionamiento de un sistema seguro es la localización de los problemas que pueden afectar a la seguridad. Los mecanismos para el control de accesos al sistema se clasifican en tres categorías:

- Restricciones sobre las contraseñas.
- Restricciones sobre el uso de las terminales.
- Restricciones de conexión.

7.2.4 Restricciones sobre las contraseñas.

Para determinar las restricciones que pueden establecerse sobre una contraseña se aplican los criterios del Libro Verde publicado por el Ministerio de Defensa estadounidense. En UNIX C2 el administrador de autentificación puede permitir a los usuarios que elijan sus propias contraseñas, o bien que las genere el sistema. Una vez elegida, puede someterse a comprobaciones para detectar si resulta obvia.

Las contraseñas pueden ser, desde el punto de vista de su tiempo de vida:

- Contraseña válida.
- Contraseña expirada. El usuario puede conectarse al ordenador y cambiarla, si le autorizan a ello.
- Contraseña muerta. La cuenta del usuario está bloqueada, por lo que no puede conectarse, y el administrador debe desbloquearla.

7.2.5 Restricciones sobre el uso de terminales.

Las terminales son la puerta del sistema, por lo que deben protegerse contra intentos de penetración. Hay que establecer un número máximo de conexiones fallidas para inutilizar la acción de los programas marcadores, que ensayan reiteradamente combinaciones de nombres de conexión y contraseñas. Cuando se excede este límite se bloquea la terminal, y ningún usuario puede conectarse a través de él con el ordenador hasta que el administrador de cuentas lo desbloquea. Así mismo, puede

establecerse el intervalo de tiempo que debe transcurrir entre dos intentos consecutivos de conexión, lo cual disuade a los potenciales atacantes, que deben esperar un tiempo entre intentos sucesivos.

7.2.6 Restricciones de conexión.

Al igual que las terminales, las cuentas de usuario tienen parámetros asociados que establecen el número de intentos de conexión antes de bloquearse, junto con el intervalo de reintento.

7.3. Protección de datos.

La primera línea de protección del sistema es el uso de los permisos estándar de UNIX sobre los archivos y directorios. Comprender los permisos que pueden asociarse a los archivos es crucial para la seguridad del sistema. Los permisos por defecto de los archivos están controlados por el comando de usuario umask , cuyo valor puede establecerse por el sistema o por el usuario.

7.3.1 Eliminación de los permisos SUID, SGID y "sticky" en las escrituras.

Los "bits" de los permisos SUID y SGID en los archivos ejecutables permiten cambiar el identificador de usuario y de grupo de un proceso. En UNIX C2 los usuarios normales no pueden establecer estos permisos porque carecen del privilegio chmodsugid . Pero además, se garantiza que estos atributos se eliminan del archivo cuando se escribe sobre él. De esta manera, se previene que un usuario pueda sustituirlo por otro programa que desee aprovechar estos atributos, dado que no puede colocárselos por sí mismo al archivo.

Permiso "sticky" en los directorios.

Cuando un directorio posee el permiso "sticky" significa que tan sólo los respectivos dueños de los archivos que haya en el directorio y el superusuario pueden borrarlos. Tan sólo el superusuario puede establecer este permiso sobre un directorio, y permanece en él hasta que su dueño, o el superusuario, lo retiran mediante el comando chmod. Obsérvese que el dueño del directorio puede quitar el permiso, pero no ponerlo. El mayor beneficio de este permiso se obtiene situándolo en todos los directorios públicos, que son aquellos en los que cualquier usuario puede depositar un fichero. Todos los directorios públicos deberían poseer el permiso "sticky" que se asigna mediante el comando de usuario:

$ chmod u+t directorio

Esto incluye, pero no está limitado, a los siguientes:

/tmp
/usr/tmp
/usr/spool/uucppublic

7.3.2 Importación de datos.

Tanto archivos, como sistemas de archivos importados desde otros sistemas, son una amenaza, sobre todo si no se manejan apropiadamente. No es conveniente suponer cuáles son los permisos de un archivo importado. Los mismos archivos pueden tener permisos diferentes en distintos sistemas debido a sus diversas políticas de seguridad. Para que la intervención del administrador sea mínima es conveniente entrenar a los usuarios para que utilicen las opciones de los programas de salvaguardia que no reasignan el dueño. Los archivos pertenecen al usuario que los importa. El comando de usuario cpio sólo cambia el dueño de los archivos cuando lo llama el superusuario.

Para minimizar el efecto de los permisos existentes en el soporte, conviene utilizar opciones que muestren su contenido sin extraerlo, como por ejemplo, -tv de los comandos tar y cpio . Cuando se

reciban soportes con los que no se está familiarizado, es conveniente recuperarlos en sistemas aislados y, tras ajustar los permisos manualmente de acuerdo con la política del sistema, ubicarlos en su lugar definitivo.

Cuando se instalan sistemas de archivos que han sido creados o manipulados fuera del sistema, se presentan los mismos problemas que en la importación de archivos. Pero además surgen otros dos nuevos: es posible que el sistema de archivos esté corrompido, y que los permisos del importado no sean aceptables por el sistema importador. En cualquier caso, instalar un sistema de archivos defectuoso puede causar la caída del sistema, corromper los datos del sistema de archivos importado, e incluso causar daños colaterales sobre otros sistemas de archivos. En cualquier caso, antes de instalar un sistema de archivos importado debería comprobarse su consistencia, y si además contiene archivos del sistema debería comprobarse que sus permisos correspondan con los del sistema receptor.

7.4.Detección de intrusiones en el sistema.

Ningún sistema puede considerarse totalmente seguro. La penetración en un sistema es fácil cuando un usuario utiliza una contraseña obvia, o alguien abandona la terminal durante una sesión de trabajo abierta. El sistema está diseñado para identificar y autentificar a los usuarios apropiadamente, y además, el acceso a los datos relacionados con la seguridad en el sistema se basa en los privilegios de subsistema. Si un usuario tiene el privilegio apropiado, entonces puede utilizar los programas del sistema para modificar las bases de datos de seguridad.

El sistema evita que los usuarios no autorizados realicen tales cambios, pero la identificación y autentificación es un paso crítico para esta protección. Estos mecanismos resultan burlados cuando alguien obtiene el acceso a una cuenta que posee más privilegios de los que le corresponden. Después de haber hecho lo posible para minimizar las posibles intrusiones en el sistema, lo único que puede hacerse es descubrir si ya se ha producido alguna. Las intrusiones tienen dos orígenes:

- Alguien hurta una contraseña de otro usuario, u obtiene el acceso a su cuenta (como cuando se abandona una terminal conectada).
- Un usuario conocido accede de forma no supervisada al equipo físico.

7.4.1 Contraseñas hurtadas.

Cada vez que se conecta un usuario, el sistema presenta al mismo la fecha y hora de su última conexión. La forma más evidente de descubrir que una contraseña ha sido sustraída es cuando la última fecha de conexión es distinta a la que recuerda su dueño. Si éste no se percata del hecho, aún tiene otra oportunidad si detecta algún cambio en sus archivos. Es necesario informar a los usuarios para que observen la fecha de la última conexión y notifiquen inmediatamente las discrepancias al administrador.

7.4.2 Acceso no supervisado al equipo físico.

Uno de los requisitos más elementales de seguridad es prevenir el acceso no supervisado al equipo físico. Aunque es habitual que los usuarios tengan que acceder al servidor para usar el disquete o los cartuchos, resulta peligroso abandonar durante horas, y sin supervisión alguna, el equipo físico. Un usuario legal del sistema puede neutralizarlo e introducirse en la cuenta del superusuario, lo cuál es el problema de seguridad más grave posible. Cuando no hay usuarios trabajando en el sistema, puede pasar desapercibida esta situación. Esta forma de intrusión tan sólo puede detectarse comprobando cuáles han sido las conexiones sobre la cuenta root, o detectando actividades sospechosas del superusuario, u observando re-arranques inexplicables del sistema. Lo mejor para resolver esta situación, es colocar al servidor bajo llave e impedir el acceso fuera de las horas establecidas a las personas que no hayan sido designadas por los administradores.

7.5. Tratamiento de sistemas de archivos corruptos.

El costo de reparar un sistema seguro que se ha vuelto inseguro es mucho mayor que el de su mantenimiento. Una vez que es de nuevo seguro, puede utilizarse un conjunto de procedimientos para monitorear la integridad del perímetro de seguridad. La corrupción de un sistema de archivos es un suceso raro, pero puede generar la eliminación de archivos críticos para el funcionamiento del sistema. El concepto de integridad del sistema es distinto al tratamiento de intrusiones, pues la acción deliberada de un usuario malicioso altera o accede a los datos. En este apartado se explican cuáles son las bases de datos más importantes relacionadas con la seguridad, y cómo recuperarlas en caso de que se haya producido una caída del sistema.

7.5.1 Archivos de la base de datos de autentificación.

Existen varias bases de datos que almacenan las características del sistema, sus usuarios, administradores y subsistemas, de forma que cada instalación puede controlar sus propios parámetros de seguridad. Esta información reside en el sistema, y es mantenida por el administrador.

Las bases de datos de auditoría y de control de archivos son independientes. Al resto de las bases de datos (contraseñas protegidas, control de terminales, subsistemas y asignación de dispositivos) se les conoce globalmente como la base de datos de autentificación, que es responsabilidad del administrador de autentificación. A continuación se proporciona una breve descripción de cada una de las bases de datos:

- Auditoría. Controla el comportamiento del subsistema de auditoría: tipos de actividad, registros del sistema en la pista de auditoría, atributos de rendimiento/fiabilidad y dispositivos en los que se recoge la información. Modificando los parámetros almacenados en esta base de datos, el administrador de auditoría puede ajustar este subsistema para corresponder al rendimiento y a los requisitos de seguridad de la instalación.

- Asignación de dispositivos. Almacena los diversos nombres de los dispositivos que corresponden al mismo periférico físico. Por ejemplo: /dev / ttyla y /dev / tty1A corresponden al mismo puerto serie. Los programas init y getty utilizan esta base de datos para detener los ataques contra el programa de conexión del sistema.
- Contraseñas protegidas. Almacena la información relativa identificación y autentificación de cada usuario. Incluye la contraseña cifrada, que ya no aparece en el archivo /etc / passwd , su vigencia, privilegios, y parámetros de auditoría. Ajustando adecuadamente esta base de datos, el administrador de autentificación controla cómo se identifican y autentifican los usuarios ante el sistema, los tipos de privilegios permitidos, y el alcance de las acciones que se registran en la pista de auditoría. La base de datos de valores por defecto del sistema, que contiene los valores por defecto de los parámetros de seguridad del sistema, también forma parte de la base de datos de contraseñas protegidas.

- Control de terminales. Proporciona el acceso al sistema a través de las terminales. Registra la actividad de conexión de cada terminal: último usuario conectado, marcas de tiempo, etc. Esta base de datos permite que el administrador de autentificación establezca políticas diferentes en cada una de las terminales, en función de su ubicación física y de las necesidades administrativas.
- Subsistemas. Cada subsistema está representado por un archivo que almacena la lista de usuarios que poseen algún privilegio especial, bien como administrador, o bien para realizar funciones especiales dentro del subsistema protegido. Esta base de datos mejora la responsabilidad de las acciones administrativas, permitiendo que sólo los usuarios autorizados puedan ejecutar programas que mantienen al subsistema. La seguridad mejora, puesto que se controla quién tiene permiso para ejecutar programas que mantienen a los subsistemas, y se responsabiliza a los usuarios reales que asumen funciones administrativas.
- Control de archivos. Ayuda a mantener la integridad de la Base Informática de Cómputo registrando el contenido y los permisos de sus archivos más importantes. Esta base de datos es una herramienta efectiva para detectar modificaciones en la copia activa de la Base. El programa para la administración del sistema integrity comprueba los permisos de los archivos de la Base Informática de Cómputo contra esta base de datos.

7.5.2 Comprobación del sistema tras una caída.

Existen diversos programas para mantener la base de datos de autentificación, el área del sistema operativo dentro del sistema de archivos y, en conjunto, todos los sistemas de archivos. En cualquier caso, la regla básica que debe seguirse es utilizar estos programas de forma que primero se apliquen sobre los componentes mas básicos del sistema de archivos, y luego sobre los más complejos. Si no se hace así, puede suceder que las correcciones realizadas a alto nivel se destruyan cuando se ejecuten los programas que realizan el mantenimiento a bajo nivel. Por tanto, el orden en el que deben utilizarse estos programas en modo mantenimiento (monousuario) para comprobar el estado del sistema después de una caída es el siguiente:

1. Realizar una comprobación de la consistencia de los sistemas de ficheros mediante el comando de mantenimiento fsck . Esto se hace automáticamente tras rearrancar el sistema. Los sistemas de archivos que contengan información sensible deben considerarse como entidades sensibles. Por tanto, la integridad del sistema de archivos que alcanza el comando de mantenimiento fsck mejora la seguridad global del sistema.

El comando de mantenimiento fsck debe ejecutarse en modo monousuario después de cualquier caída o terminación anormal del sistema. Es posible que cuando el sistema se cayó estuvieran modificándose archivos de usuario, del sistema, o de auditoría. Aunque es probable que estos datos se hayan perdido, la ejecución del comando fsck es la única posibilidad de recuperarlos en el directorio /lost + found del sistema de archivos al que pertenezcan.

2. Comprobar la ausencia de archivos críticos. Cuando el sistema para súbitamente, bien por un fallo en el equipo físico/lógico, bien por una caída de tensión, pueden extraviarse archivos de la base de datos de seguridad, o es posible que se estuvieran modificando cuando se cayó el sistema, y se encuentren en un estado inconsistente.

3. Generar un informe de auditoría.

4. Comprobar la consistencia de la base de datos de autentificación para comprobar todas las bases de datos de seguridad.

5. Comprobar los permisos de los archivos del sistema. Se comparan las entradas de la base de datos de control de archivos contra los permisos actuales de los archivos del sistema. Cuando se encuentran discrepancias en un archivo, éste debe examinarse para determinar:

-¿Cuáles son los permisos de dueño/grupo/otros actuales del archivo?
-¿Cuáles deberían ser los permisos correctos?
-¿Cuáles son las fechas de último acceso y modificación del archivo?
-¿Quién estaba trabajando en el sistema en ese momento?

Una vez que se ha descubierto el motivo que ha producido la discrepancia, deben restablecerse los permisos correctos. Después de que se han corregido todas las discrepancias existentes, debe ejecutarse de nuevo un programa que valide los permisos.

7.5.3 Uso de la terminal de prevalecimiento.

Cuando todas las bases de datos de seguridad están corruptas, no se permiten las conexiones, pero existe una terminal de prevalecimiento en la cuál puede conectarse tan sólo la cuenta root. Esta terminal, que suele corresponder con el servidor, debería ubicarse en u lugar seguro, puesto que en él los bloqueos normales no se aplican a la cuenta del superusuario.

7.6. Auditoría.

El subsistema de auditoría registra los sucesos relacionados con la seguridad del sistema mediante un registro de auditoría, que puede examinarse con posterioridad. Esta pista permite detectar las intrusiones en el sistema y el uso incorrecto de los recursos.

El proceso de auditoría permite revisar los datos recopilados para examinar tipos de accesos a los objetos (archivos), y observar las acciones de usuarios específicos y sus procesos. También se auditan los intentos de violar los mecanismos de protección y autorización. El subsistema de auditoría proporciona un alto grado de confiabilidad, puesto que también se registran los intentos de soslayar los mecanismos de seguridad.

El subsistema de auditoría utiliza las llamadas al sistema operativo y las utilidades para clasificar las acciones de los usuarios en tipos de sucesos, que pueden usarse para la generación y reducción de la auditoría selectiva. Uno de estos tipos de sucesos es la denegación de control de acceso discrecional, que registra los intentos de usar un objeto de forma no permitida. Por ejemplo, si un proceso intenta escribir en un archivo de sólo lectura, se audita este suceso, lo cuál muestra que el proceso intentó escribir en un archivo sobre el que no tenía derechos. Cuando el administrador examine el registro de auditoría, será fácil observar los intentos sucesivos de acceso a archivos para los que no se concede el permiso apropiado.

Cuando los usuarios intentan conectarse al sistema, deben pasar a través del proceso de identificación y autentificación, antes de que se les conceda el acceso a la terminal. El mecanismo de seguridad marca en cada proceso creado por el usuario, un indicador invariable de su identidad: el identificador de usuario durante la conexión ("Login User IDentificator, LUID"). Este identificador se mantiene incluso cuando se cambia de cuenta de usuario a través del comando su . En cada registro de auditoría generado se almacena tanto el LUID como los identificadores, reales y efectivos, de usuario y grupo. Con esto, puede garantizarse la responsabilidad de las acciones de los usuarios.

El subsistema de auditoría posee su propio administrador, que tiene un control completo sobre los sucesos seleccionados que generan registros de auditoría, sobre los valores de los parámetros que controlan al subsistema, y sobre la posterior reducción y análisis de los datos auditados.

7.6.1 Componentes del subsistema de auditoría.

El subsistema de auditoria posee cinco componentes:

- Mecanismo de auditoría del núcleo.
- Dispositivo manejador de la auditoría.
- Proceso servidor para la compactación del registro de la auditoría.
- Programa para la interfaz de administración de la auditoría.
- Utilidad para el análisis y la reducción de datos.

También existen utilidades seguras que no se consideran una parte del subsistema de auditoria, tales como el comando de usuario login , que escriben registros en la pista de auditoría.

Mecanismo de auditoría del núcleo.

Este mecanismo genera los registros de auditoría basándose en la actividad de los procesos de los usuarios a través de las llamadas que realizan al núcleo del sistema operativo. Cada llamada del núcleo contiene una entrada en una tabla de subsistema que indica si aquella es relevante para la seguridad, y si es así, a qué tipo de suceso corresponde. Además, existe una tabla de códigos de error que clasifica las llamadas en sucesos concretos de seguridad. A continuación, este mecanismo realiza una llamada interna al dispositivo manejador de la auditoría para escribir un registro sobre la pista.

Por ejemplo, la llamada al sistema open ( ) está clasificada como un suceso que hace disponible a un objeto. Si el usuario jgomez realiza una llamada open ( ) sobre el archivo agenda , y tiene éxito, se genera un registro de auditoría que indica el suceso. Sin embargo, si la llamada falla porque el usuario jgomez realiza una llamada open ( ) para abrir en modo escritura el archivo agenda, y éste no posee dicho permiso, entonces la acción se clasifica como un suceso de denegación de control de acceso discrecional de jgomez sobre el objeto agenda.

Dispositivo manejador de la auditoría.

Es el responsable de las siguientes acciones:

- Acepta registros de auditoría del mecanismo del núcleo y de las utilidades seguras.
- Crea y escribe los archivos intermedios de la pista de auditoría.
- Proporciona datos de la pista al proceso servidor de auditoría para su compactación.
- Ofrece registros para la generación de auditoría selectiva basada en tipos de sucesos, identificadores de usuario y de grupo.

El dispositivo manejador de la auditoría mantiene el registro como una colección de archivos de auditoría. Cada vez que se habilita, comienza una nueva sesión de auditoría que, al arrancar, hace que el subsistema genere una colección de archivos en los que se escriben los registros de auditoría. Cuando la colección alcanza un determinado tamaño (configurable por el administrador), el subsistema crea una nueva colección , y empieza a escribir en ella.

Proceso servidor de compactación de la auditoría.

El objetivo principal del servidor es proporcionar un mecanismo de compactación y registro para la sesión de auditoría. También posee una función de soporte para los subsistemas protegidos, permitiéndoles escribir registros de auditoría. Dependiendo de los criterios para la generación de registros que se hayan establecido, puede generarse una gran cantidad de información de auditoría en el sistema. El servidor proporciona un mecanismo de compactación que comprime los datos de auditoría en un formato de registro empaquetado que se almacena en el archivo de compactación de auditoría. El algoritmo de compactación empleado suele reducir un 60% el tamaño de los archivos, lo cuál ahorra una gran cantidad del espacio necesario para almacenar los registros de auditoría.

La segunda función del servidor es proporcionar un archivo histórico en el que se describe la sesión de auditoría actual. En este archivo hay información sobre el número de registros disponibles en la salida del archivo compactado para la sesión, la hora de arranque y parada de la sesión, así como otros indicadores correspondientes a su estado.

La tercera función del servidor de auditoría es la de programa interfaz con el manejador del dispositivo de auditoría, para la escritura de registros procedentes de los subsistemas protegidos que no poseen el privilegio para escribir en la auditoría.

Interfaz del subsistema de auditoría.

Es una utilidad que proporciona un conjunto de opciones sencillas para configurar y mantener el subsistema de auditoría. Esto permite al administrador realizar la inicialización, modificación y mantenimiento del subsistema (copias de seguridad, recuperaciones), y la reducción tanto de datos de auditoría generales como selectivos.

Utilidad para la reducción y el análisis de datos.

El subsistema de auditoría incluye una utilidad para la reducción y el análisis de datos, que permite examinar tanto las pistas de las sesiones de auditoría anteriores como la de la actual. Mediante el fichero histórico que genera el servidor, la utilidad de reducción puede identificar los archivos de compactación necesarios para reducir una sesión de auditoría. Puesto que estos archivos se encuentran en un formato comprimido, el programa de reducción posee las funciones necesarias para descomprimir los datos.

Para proporcionar un análisis efectivo de los datos auditados, la utilidad de reducción permite especificar tipos de sucesos, identificadores de usuario, de grupo y nombres de objetos para reducir selectivamente los datos. Además, puede definirse un intervalo de tiempo aplicable para buscar los registros que encajen con el criterio especificado, y cuando un registro está fuera del intervalo indicado se descarta para esa reducción.

7.6.2 Método de auditoría.

En este subapartado se explica el funcionamiento del subsistema de auditoría, y cuáles son los criterios utilizados para recopilar datos.

Privilegios de auditoría.

Hay cuatro privilegios asociados al subsistema de auditoría:

- Privilegio para establecer los parámetros de auditoría a todos los usuarios del sistema.
- Privilegio para registrar información específica en el registro de auditoría.
- Privilegio para que los usuarios auditen sus propias actividades.
- Privilegio para prevenir cualquier auditoría.

Origen de los registros de auditoría.

La pista de auditoría contiene los sucesos relativos a la seguridad del sistema. Una auditoría eficaz no sólo involucra a las peticiones de los procesos de usuario realizadas a través de las llamadas al sistema operativo, sino que también existen otros sucesos, tales como la conexión, desconexión, e

intentos fallidos de conexión. Estos sucesos son críticos para determinar quién ha accedido al sistema, a qué hora, desde qué terminal, y cuáles han sido las acciones realizadas. Los fallos de conexión son imposibles de auditar a nivel del núcleo del sistema operativo, pues éste no tiene conocimiento sobre lo que está haciendo concretamente una aplicación; en este caso, el comando de usuario para la conexión al ordenador login. Por tanto, a determinadas aplicaciones críticas para la seguridad, tales como login, debe permitírseles generar registros de auditoría.
A continuación se describen las tres fuentes de origen de los registros de auditoría.

Mecanismo de auditoría del núcleo.

Un gran porcentaje de los registros de auditoría almacenados en el registro proceden de aquí. Esta parte del subsistema de auditoría genera registros como respuesta a las llamadas al sistema de los procesos de usuario que corresponden con sucesos relacionados con la seguridad. Llamadas como open ( ) pueden corresponder a varios sucesos, dependiendo tanto de los argumentos como del estado del archivo que se desea abrir. Si open ( ) recibe el atributo 0_CREAT, se crea el archivo siempre que no exista. Pero si el atributo que recibe es 0_TRUNC, entonces si el archivo existe, se vacía. Este ejemplo ilustra cómo la llamada open ( ) puede corresponder a tres sucesos distintos: hacer un objeto disponible, creación de un objeto, o modificación de un objeto.

El mecanismo de auditoría del núcleo determina al final de la llamada al sistema a qué tipo de suceso pertenece la llamada, y si ese suceso debe ser auditado a petición del administrador.

Aplicaciones seguras.

La base segura de cómputo (TCB) contiene un conjunto de aplicaciones seguras que son esenciales para proporcionar un entorno de confianza. Para reducir la cantidad de datos de auditoría escritos en la pista, y hacer que ésta resulte más significativa, se permite que estas aplicaciones seguras escriban directamente sobre el dispositivo de auditoría . Esto permite al comando login escribir un

registro de auditoria de conexión en la pista, evitando que una conexión al sistema se represente como una colección de llamadas al sistema operativo necesarias par completar el procedimiento de conexión.

Subsistemas autorizados.

El tercer método para la producción de registros de auditoría es a través de subsistemas autorizados. A veces, un subsistema detecta problemas o inconsistencias, que recomiendan la escritura de un registro de auditoria informativo. Sin embargo, los subsistemas no pueden escribir directamente registros de auditoria.

En su lugar, los subsistemas formatean los registros como si fueran una aplicación segura, y se los entregan al proceso servidor de auditoria a través de una interfaz segura. El servidor, que es una aplicación segura, realiza la tarea de escribir el registro en el dispositivo de auditoría. Esto permite que los procesos de los subsistemas protegidos generen registros de auditoría concisos e informativos, sin tener que distribuir el privilegio para escribir en la auditoria entre estos subsistemas.

Responsabilidad de la auditoría.

El subsistema de auditoria registra los sucesos relacionados con la seguridad del sistema asociándoles con un usuario específico. Los usuarios se conectan al sistema a través del comando login . Este programa realiza la autentificación del usuario para determinar si se le permite el acceso.

El procedimiento de conexión ha sido mejorado para proporcionar soporte a la auditoria tanto para conexiones satisfactorias como insatisfactorias. Cuando un usuario se conecta correctamente, el comando login marca en el proceso el identificador de usuario durante la conexión (LUID), que a diferencia de los otros identificadores efectivos y reales de usuario y grupo, no puede ambiarse. Por

tanto, se mantiene una responsabilidad estricta tanto para el proceso como para el usuario. Un proceso de usuario puede continuar haciendo llamadas a setuid ( ) y setgid ( ), que son auditadas. Los registros de auditoría indican el LUID del proceso junto con los identificadores reales y efectivos de usuario y grupo.

Tipos de sucesos de auditoría.

Cada registro de auditoría, independientemente de su creador, se marca con un tipo de suceso. Para las llamadas al sistema de los procesos de usuario, el tipo de suceso lo determina el mecanismo de auditoría del núcleo, basándose en cuál es la llamada al sistema y en su resultado. Para las aplicaciones o el subsistema de auditoría, es el proceso que escribe el registro de auditoría quien establece el tipo de suceso, que no es cambiado ni por el dispositivo de auditoría, ni por su servidor.

Los tipos de sucesos son importantes porque clasifican los eventos de seguridad del sistema. Tanto la generación como la reducción de registros de auditoría se controlan mediante los tipos de sucesos.

El subsistema de auditoría proporciona un amplio rango de tipos de sucesos, que son un equilibrio entre la granularidad y las clases de seguridad relevantes. En la tabla adjunta se resumen los tipos de sucesos:

Tipo de suceso de auditoría
Arranque/parada de la computadora
Conexión/desconexión a una cuenta
Creación/borrado de proceso
Hacer un objeto disponible
Correspondencia de un objeto con un sujeto
Modificación de un objeto
Hacer que un objeto no esté disponible
Creación de un objeto
Borrado de un objeto
Cambios de control de accesos discrecional
Denegación en el control de accesos discrecional
Acciones del administrador/operador
Autorización insuficiente
Denegación de recurso
Funciones para la comunicación entre procesos
Modificaciones de proceso
Sucesos del subsistema de auditoría
Sucesos de base de datos
Sucesos de subsistemas
Uso de autorización

Tabla 3. Tipos de sucesos auditables en un sistema C2

El subsistema de auditoría utiliza los tipos de sucesos para determinar si un registro debe escribirse en la pista de auditoría. El administrador de auditoría tiene pleno control sobre cuáles son los sucesos que deben auditarse.

Auditoria obligatoria.

Con objeto de mantener una información precisa sobre todos los procesos de usuario que sirva para disponer de una auditoria significativa, el mecanismo de auditoría del núcleo siempre audita ciertas llamadas al sistema. Esto significa que cuando se habilita la auditoría, algunos sucesos serán auditados aunque el administrador no haya seleccionado ninguno. A estas llamadas al sistema se les denomina obligatorias, y son esenciales para el mantenimiento del estado del proceso.

La auditoría obligatoria no se limita tan sólo al grupo de llamadas al sistema indicadas en la tabla anterior. El suceso de conexión es el único registro de auditoría obligatorio definido para una aplicación segura. Cuando un usuario se conecta a la terminal, este registro contiene un indicador de la terminal en la que se ha producido la conexión. Si el mismo usuario está conectado en varias terminales del sistema, es posible seguir sus acciones en cada una de las terminales.

Guía para una auditoría eficiente.

El subsistema de auditoría está diseñado para ofrecer un rendimiento y una confianza flexible, y permite recopilar los datos de auditoría que se deseen. La generación de registros de auditoria soporta la pre-selección de sucesos, identificadores de usuario y de grupo. La pre-selección es un instrumento de gran valor cuando es necesario concentrarse por alguna razón en un usuario o grupo de usuarios. Los tipos de sucesos también pueden utilizarse para la pre-selección, como por ejemplo, auditar tan sólo los sucesos de conexión y desconexión. La pre-selección proporciona grandes ahorros en el espacio del disco porque se reduce la cantidad de registros de auditoría escritos por el subsistema en las colecciones de archivos. Sin embargo, la pre-selección tiene un inconveniente. Si ha habido una violación de seguridad en el sistema, y ese suceso, o el usuario que lo ha perpetrado, no ha sido seleccionado para la auditoría, entonces se ha perdido el registro de la acción.

Por este motivo, es más conservador no pre-seleccionar los sucesos de auditoría ni los usuarios o grupos, y en su lugar, realizar una auditoría completa. El beneficio que se obtiene es que cualquier suceso relacionado con la seguridad que haya ocurrido está registrado en la pista de auditoría. La desventaja es que consume una gran cantidad de espacio en disco y un porcentaje notable de la UCP.

Es posible combinar la auditoría completa con la post-selección para examinar tan sólo los registros que resulten interesantes, realizando un examen selectivo de la pista de auditoría basándose en tipos de sucesos, e identificadores de usuario y de grupo, nombres de objetos, así como fecha y hora de los registros generados. Con todo esto, el subsistema de auditoría combinado con la utilidad para la reducción y el análisis de datos proporciona la flexibilidad para negociar entre el rendimiento del sistema y la capacidad del disco mediante la pre-selección, y la conveniencia de la auditoría completa combinada con la post-selección.

La administración del subsistema de auditoría es la clave para la auditoría eficiente. A través de un cuidadoso establecimiento y uso del subsistema de auditoría, se dispone de una herramienta poderosa que ayuda a mantener seguro el sistema, e identificar los problemas cuando aparecen. El subsistema está diseñado para ser muy completo desde el punto de vista de la cobertura de los sucesos auditados, tanto por parte de las acciones del núcleo, como por el uso de las utilidades del sistema. También está diseñado para ser seguro y minimizar el impacto sobre el rendimiento global del sistema.

7.7. Recomendaciones para el mantenimiento seguro de un sistema.

A continuación, se describe un conjunto de reglas prácticas para facilitar el mantenimiento seguro de un sistema.

7.7.1. Mantenimiento seguro de las cuentas administrativas.

Todas las recomendaciones par el mantenimiento seguro de las cuentas de los usuarios se aplican con especial énfasis a las administrativas. Cuando un pirata se apropia de una cuenta de usuario,

sólo afecta a sus archivos, y quizás a los de su grupo, pero si una cuenta administrativa como root se ve afectada, entonces todo el sistema está en peligro.

Estas son algunas de las recomendaciones aplicables sobre las cuentas administrativas:

- No deben ejecutarse programas de otros usuarios desde una cuenta administrativa. Previamente hay que conectarse a esas cuentas a través del comando su, y comprobar el efecto del programa.
- Poner el directorio actual como último camino de la variable de entorno PATH. De esta manera se ejecutarán en primer lugar los programas del sistema, evitando la activación de caballos de Troya depositados en la cuenta.
- Al llamar a un comando, debe utilizarse su camino completo, y no su nombre terminal. Por ejemplo, hay que escribir /bin/su , en vez de su.
- No dejar la terminal abandonada con una sesión de trabajo abierta.
- Hay que obligar a que la cuenta root sólo pueda utilizarse en el servidor. Si esto no fuera posible, debe cambiarse el nombre de la cuenta root por otro que resulte difícil de adivinar a los piratas.
- Cambiar una vez al mes las contraseñas de las cuentas administrativas de uso frecuente.
- Asegurarse de que todas las llamadas al comando su se registren en el archivo /usr/adm/sulog , y que éste sólo sea accesible por las cuentas administrativas.

- Impedir que personas esporádicas actúen en funciones administrativas, ni siquiera aunque se les vigile.

7.7.2 Mantenimiento seguro del sistema.

A continuación se describen las acciones que deben realizarse para el mantenimiento seguro del sistema:

- Pensar en cuáles pueden ser las vulnerabilidades del sistema.
- Mantener la integridad de los archivos del sistema. Es conveniente crear una base de datos con los permisos correctos de los archivos, y comprobar que los actuales son adecuados. Cualquier programa de root con el atributo SUID es candidato para que un pirata lo sustituya por un caballo de Troya.
- Ser muy cuidadoso con los permisos de los dispositivos, sobre todo, con los de los discos.
- Sospechar de los archivos existentes en cuentas de usuario cuyo dueño sean cuentas administrativas, y que posean los atributos SUID/SGID.
- No montar un sistema de archivos exterior, sin comprobar previamente los programas con atributo SUID/SGID, y los permisos de los dispositivos.
- Proteger todas las copias de seguridad que se realicen.
- Activar el mecanismo de envejecimiento de las contraseñas, estableciendo una vigencia mínima y otra máxima. Mover la contraseña cifrada hacia un archivo (/etc/shadow) que tan sólo puedan manipular las cuentas administrativas.
- Mantener un informe de privilegios de accesos al sistema, que indique los usuarios autorizados a entrar en la computadora. Debe existir un mecanismo instantáneo para impedir el acceso al sistema a una persona a la que acabe de retirársele el privilegio.
- Activar el comando acct de contabilidad del sistema o, si se dispone de ella, la auditoría, al menos para registrar los sucesos de conexión.
- Si dispone de ello, permitir que una terminal o una cuenta se bloqueen al alcanzar un cierto número de intentos fallidos.
- Programar el intérprete de comandos para que concluya su ejecución después de un cierto periodo de inactividad, por ejemplo, treinta minutos.
- Tan sólo debería poder ejecutar el comando su el superusuario.
- Si un programa de seguridad debe ser ejecutable a nivel de grupo u otros, debe comprobarse que sólo son ejecutables, no legibles ni modificables.
- Asignar el valor 077 a la máscara de usuario (umask 077) en el archivo /etc/profile de configuración de las cuentas de usuario.

CAPITULO 8

Instituciones y documentos sobre la seguridad de UNIX en internet

8.1 Computer Emergency Response Team.
8.2 Otros foros telemáticos de interés.
8.3 Preguntas frecuentes sobre seguridad.

n el presente capitulo se relacionarán varias instituciones de diferente naturaleza que han declarado su inquietud por la seguridad informática, y muy en especial, en los sistemas abiertos, entre los que naturalmente se cuenta Unix, produciendo un número siempre creciente de documentos electrónicos, de interés.

La proliferación y la ágil estructura de esas organizaciones no podría explicarse sin contar con la suspicacia no sólo de los particulares , sino también , y sobre todo , de empresas, instituciones y gobiernos de varios paises del mundo , que respaldan sus sensibles actividades.

Como no podría ser de otra manera, el centro de gravedad de todas ellas está en los Estados Unidos , en donde la Agencia de Seguridad Nacional (National Security Agency) sigue muy de cerca todo lo que tiene que ver con seguridad informática. En todo caso sólo se tratará de referencias lo más técnicamente posible algunas organizaciones, con domicilio, prestigiosos representantes, actividad y política informativa conocida, que puedan ser útiles para el administrador de sistemas Unix responsable en instalaciones sensibles, que siempre necesitará fuentes de información y documentación dinámicas, y tal vez, alguna ayuda o respaldo técnico, administrativo o jurídico, ante dificultades excepcionales.

8.1 Computer Emergency Response Team (CERT)

El Equipo de Respuesta ante Emergencias Informáticas es, muy probablemente, la más activa y seria organización que debe ser considerada por los administradores de Unix de todo el mundo, y muy en especial, por los que tienen sistemas conectados a Internet.

Fue creado por la Agencia de Proyectos de Investigación Avanzados para la Defensa (Defense Advanced Research Projects Agency , DARPA) en Noviembre de 1988, es decir tiene una raíz militar. El hecho que catalizó esta iniciativa fue la aparición del gusano (worm) de Internet que invadió miles de computadoras conectadas a esta red, muchas de ellas de gran importancia militar y estratégica. Por ello, la División de Sistemas Electrónicos del Comando de Sistemas de la Fuerza Aéreas (Air Force System

Command -Electronic System Division) utiliza el CERT como agente ejecutivo , acelerando la transición de la tecnología del software a los sistemas para la defensa. El CERT tiene a su vez un centro de coordinación y un órgano llamado FIRST (Forum of Incident Response and Security Teams, Foro de Equipos de Seguridad y Respuesta ante Incidentes), mediante el cual se relacionan expertos en seguridad de todo el mundo para encontrar las soluciones idóneas a cada problema concreto. Uno de los mayores honores que puede existir actualmente para un experto en seguridad Unix es el ser citado en una de las recomendaciones del CERT (CERT Advisories).

Este centro depende del Instituto de Ingeniería del Software (Software Engineering Institute, SEI) de la Universidad de Carnegie Mellon, lo que le garantiza cierta independencia de los fabricantes de equipos y aplicaciones, con los que, por otra parte, mantiene excelentes relaciones.

Las recomendaciones del CERT cubren la inmensa mayoría de las incidencias que afectan a la seguridad de sistemas informáticos abiertos y normalizados, diagnosticando el problema (que se trata generalmente de un agujero en la seguridad del software o de su configuración), una relación de las peores consecuencias que podrían derivarse de él, y las medidas que el CERT recomienda en cada caso. Generalmente, las recomendaciones del CERT se distribuyen por correo electrónico . o se recogen mediante ftp anónimo a través de una conexión Internet cuyas instrucciones figuran en todos y cada uno de los documentos electrónicos que emite.

También existe un grupo de noticias (news), comp.security.announce, que distribuye a sus suscriptores, o a quien pueda leer este grupo directamente , cada una de las recomendaciones del CERT según se van publicando , aunque, en principio, no se proporcionan por esta vía parches a nivel de código fuente ya que ésta es un potestad exclusiva del fabricante de los sistemas.

Las secuencias de cientos de recomendaciones del CERT describe mejor que ninguna otra referencia la constante evolución de la seguridad de los sistemas abiertos, desde los primeros efectos, casi obvios, hasta los más recientes detectados en los sofisticados World Wide Webs con Mosaic.

Evidentemente, resultan más delicadas, por razones comerciales, la elaboración y distribución de recomendaciones del CERT que hacen referencias a sistemas concretos. Incluso puede sospecharse una posible intencionalidad del fabricante deseoso de reservarse puerta traseras , a través de las cuales poder acceder a los sistemas instalados y administrados por sus clientes.

En cualquier caso, cuando las recomendaciones del CERT hacen referencia a un producto comercial en particular, sólo son publicados si ya se ha encontrado una solución para él, que por lo general es suministrada por el propio fabricante del equipo o sistema en cuestión, en forma de parche o de una revisión nueva del software, o bien puede obtenerse mediante ftp anónimo de uno o varios servidores de archivos conectados a Internet. El servidor del propio CERT tiene la dirección IP cert.org (192.88.209.5), y el resto de los datos de esta organización son:

Internet E-mail: cert@cert.org
Tel: (1) 412-268-7090 (servicio de atención durante las 24 horas)

Lista de correo electrónico para la distribución de información de virus multiplataforma :

listserv@lehigh.edu

En caso de tener un incidente de seguridad que se considere necesario comunicar formalmente, se ha de proporcionar, en ingles, la siguiente información, lo más completa que sea posible :

1) Names of host (s) compromised at your site.
   Nombres de los servidores locales comprometidos.
2) Architecture and OS (operating system and revision) of compromised host(s).
   Arquitectura y sistema operativo, con su versión y revisión, de los servidores comprometidos.
3) Wheter or not security patches have been applied to the compromised host(s); if so, were patches applied before or after the intrusion.
   Si se han aplicado parches o no de seguridad a los servidores comprometidos; si es así, indicar si los parches se aplicaron antes o después de la intrusión.
4) Account name(s) compromised.
   Nombre de la cuenta comprometida.
5) Others host(s)/site(s) involved in the intrusion and whether or not you have alredy contacted those site(s) abaut the intrusion.
   Otros servidores/instalaciones implicados por la intrusión y si ya se ha contactado o no con dichas instalaciones en relación con la intrusión.
6) If other site(s) have been contacted, the contact information used for contacting the site(s) involved.

Si se ha contactado ya con otras instalaciones, cuál fue la información utilizada para contactar con las mismas.

7) If CERT is to contact the other site(s) , can we give the other sites your contact information (i.e., your name, e-mail, address, and peone number

Si el CERT ha de contactar con otras instalaciones, puede darles su información de contacto (por ejemplo, su nombre, dirección de correo electrónico y número de teléfono).

8) Whether or not any law enforcement agencies have been contacted.

¿ Se ha contactado o no alguna instancia legal ?

9) Appropriate log extracts (including timestamps).

Extractos de registros apropiados (incluyendo horas y fechas).

10) What assistance you would like from the CERT Coordination Center.

¿ Qué ayuda desearía del Centro de Coordinación CERT?

8.2 Otros foros telemáticos de interés

Existen algunos otros grupos de noticias muy útiles para la discusión telemática de seguridad informática en Internet. Usuarios de todo el mundo participan en ellos publicando sus mensajes y ofreciendo su dirección de correo electrónico para relaciones posteriores.

Uno de los más activos es comp.security que contiene las recomendaciones del CERT antes descritas, y el boletín comp.security.unix , especialmente dedicado al sistema operativo Unix. Otros, como el boletín comp.security.misc, que antes de existir algunos posteriores más específicos, reflejó la actualidad de la seguridad en Unix, o el boletín alt.security, que pese ha estar relacionado con el anterior, puede llegar a tratar incluso de temas tan alejados como las cerraduras de automóviles, o los sistemas de alarmas acústica, junto a los derechos civiles en informática y comunicaciones.

Entre estos foros de carácter más general, destaca comp.risks, que sistemáticamente sigue cualquier riesgo o incidente que tenga una explicación tecnológica. Está moderado por la asociación informática ACM (Association for Computer Machinery), y periódicamente edita la revista " Software Engineering Notes ", recopilando los mensajes más interesantes recibidos en la dirección electrónica del

Grupo de Interés Especial en Ingeniería del Software (Special Interest Group on Software Engeenering, SIGSOFT). Su responsable es Peter G: Neumann (Neumann@csl.sri.com).

Periódicamente también, los mensajes más relevantes e ilustrativos procedentes de la mayoria de estos foros telemáticos para componer un gran archivo orientativo sobre las preguntas formuladas con más frecuencia. Así es como surgen los archivos de " Frequently Asked Question " (FAQs). Lo que sigue se basa en el boletín alt-security-faq que mantuvo Alec David Muffett .

Como no podría ser de otra manera , no todo el que envía un ,mensaje a través de Internet está los suficientemente informado, o mínimamente formado, para que tal mensaje resulte del interés de otros usuarios, por lo que, en algunas ocasiones , se escriben mensajes fuera de lugar , improcedentes e ingenuos.

Los más llamativos, que suelen provocar sonrisa, o incluso situaciones ridículas, son los que vienen a preguntar algo como " ¿ Puede alguien explicarme como funciona el agujero de seguridad de tal sistema y cuál es el procedimiento para conseguir permisos de administrador con él ? "

8.3 Preguntas frecuentes sobre seguridad

Siguiendo una versión adaptada del popular archivo de Alec Muffett, se harán aquí algunas preguntas que se pretende que sirvan para centrar la utilidad de los conocimientos sobre seguridad en Unix.

8.3.1 ¿ Cuál es la diferencia entre un "hacker" y un "cracker" ?

Para ser exactos, estos calificativos se aplican a distintos tipos de talentos (en ocasiones delincuentes) informáticos, aunque en español puedan perderse los matices y los dos términos se confundan con el de "pirata".

Un "cracker" presume de ser capaz de acceder a los sistemas a los que no tiene autorización . A veces lo hace por algún tipo de débil justificación, con la excusa de que " así se demuestra que es posible ", y en la mayor parte de las casos, simplemente, por el placer de hacer algo ilegal o ilícito, y por lo tanto, ser considerado como un miembro más de un selecto grupo, con la marginación de los escogidos.

Existen ciertos tipos de "crackers" completamente antisociales que cometen actos vandálicos, que muchas veces llegan a ser completamente irreparables o irreversibles, como borrar o contaminar información. Pero incluso en las actividades aparentemente más serias, como la diplomacia, la administración de justicia o las estrictamente militares, también hay gamberros, novatadas, y pura y simple destrucción innecesaria y gratuita.

Otros incontrolados, entre los que se encuentran la inmensa mayoría de los que se tiene público conocimiento, presumen de acumular méritos saltándose las protecciones de las aplicaciones comerciales o profesionales. Su propósito suele ser el de disfrutar, y luego distribuir gratuitamente, algo por lo que se debería pagar , pero que en la práctica resulta muy difícil de controlar y cobrar como propiedad intelectual.

Por otra parte el término " hacker " califica a los genuinos talentos informáticos, que no sólo acumulan conocimientos enciclopédicos conocimientos, sino que también son capaces de improvisar las soluciones más ingeniosas. Tampoco dudan en elegir el camino más corto, ni en utilizar todo tipo de recursos, incluyendo código ya escrito que resuelve una parte , o tal vez todo el problema de forma práctica y brillante, por lo general.

En Internet, cuando se trata de explicar la terminología de los " hackers ", es casi obligado citar " Jargon File ", que es un enorme archivo ASCII editado por Eric Raymond en el que se relacionan , a modo de glosario, las palabras que distinguen a los adictos a la informática. Este fichero ha sido editado por el Instituto de Tecnología de Massachussets en un magnífico libro titulado " The New Hackers Dictionary ", que se ha convertido en una auténtica referencia de terminología ultramoderna, permanentemente actualizada.

8.3.2 Seguridad y secretismo

Leyendo mensajes y literatura sobre seguridad informática se acaba por distinguir dos filosofías claramente diferenciadas:

- La más peligrosa es la que parte de la convicción de que la seguridad sólo se consigue mientras no exista ni una sola persona fuera del grupo de máxima confianza, que conozca los mecanismos internos de la seguridad. En la actualidad se puede afirmar que esta filosofía proporciona solo una pseudoseguridad institucional.
- Pero el éxito de los sistemas abiertos, la interconectividad y la mejor comprensión de las técnicas de programación, así como el notable aumento de la potencia y las posibilidades de las computadoras domésticas , han puesto en peligro al secretismo informático de forma prácticamente irrebersible.

La base técnica de estos sistemas consiste en la necesidad de saber cómo hacer algo, para poder llevarlo a cabo, sean cuales fueren los motivos por los que se desee hacerlo. Si no se sabe como hacerlo, el sistema, aparentemente, no está en peligro.

En teoría, puede admitirse que estos mecanismos funcionan tan disuasoriamente como se espera, pero en la práctica, hacen depender al sistema de información de un " tal vez no tan selecto " grupo de especialista que en cualquier momento pueden cambiar de trabajo, ideas o intereses. Una vez que se pierde el control del secreto, hay que volver ha empezar todo de nuevo.

Por lo tanto, es evidente que cada vez existen más argumentos para crear y utilizar sistemas que son seguros no por razones filosóficas, sociales, laborales, políticas o, simplemente, empresariales o autoritarias, sino por otra mucho más sólidas y demostrables, como sólo pueden serlo las matemáticas. ¿ Y qué es lo que hace un sistema inseguro ? ¡ Pues encenderlo ! Simplemente , el que funcione, lo convierte en inseguro.

El experto en seguridad Gene Spafford, lo dice así:

> " El único sistema auténticamente seguro es el que está desconectado, desenchufado, y empaquetado en un recipiente hermético de titanio, guardado en un búnker de hormigón, rodeado de gas nervioso y custodiado por guardias muy bien armados y magníficamente pagados. E incluso así, yo no me jugaría la vida por él."

Es decir, que un sistema es tan seguro como lo sea la gente que accede a él. Incluso puede llegar a ser completamente seguro sin ningún tipo de protección especial durante tanto tiempo como continúe en funcionamiento entre personas responsables, y se hagan copias de seguridad regularmente en previsión de posibles fallos en el equipo físico.

Los problemas comienzan cuando existe una necesidad, como la confidencialidad más o menos compartida, que tiene que ser razonablemente satisfecha: Una vez que se comienza a poner contraseñas y otras medidas específicas para la seguridad de un sistema, la experiencia demuestra que es muy difícil terminar de hacerlo alguna vez.

Según las preguntas frecuentes sobre seguridad del FAQ, los agujeros pueden ser de cualquiera de los siguientes cuatro tipos, o bien fruto de la combinación de varios o todos ellos:

1. Físicos, debidos al acceso no autorizado de personas a una máquina, pudiendo manipularla como no debiera permitirse.
   El mejor ejemplo de ello , es la facilidad que ofrecen algunos sistemas para ser eventualmente arrancados en modo monousuario con permisos de administrador de forma que se puede acceder a cualquier archivo o programa si se tiene la ocasión de llegar a estar discretamente a solas con la computadora. Ni que decir tiene que las copias de seguridad con información confidencial siempre deben ser custodiadas, y es recomendable utilizar algoritmos de cifrado, si no se consigue garantizar que personas no autorizadas jamás accederán físicamente a ellas.

2. Del propio equipo lógico , por cuyos fallos de seguridad queda muy comprometida, generalmente, por la indebida escritura de privilegios objetos de software (procesos - servidores -"daemons" , ejecución temporizada de programas "cronjobs")capaces de provocar acciones que supuestamente nunca deberían realizarse.

El ejemplo más famoso es la vulnerabilidad del comando debug dentro de la utilidad para el envío de correo electrónico sendmail , que permite iniciar un intérprete de comandos con privilegios de superusuario, y desde él, hacer lo que desee prácticamente sin ningún tipo de restricción. Pese a lo que creen la mayor parte de los usuarios, los ataque que utilizan estos agujeros no se limitan a los que permitieron la difusión del gusano de Internet, ya que cualquier "cracker" puede aprovecharlos utilizando el puerto 25 del comando para conexión remota telnet .

Permanentemente aparecen nuevas vulnerabilidades en los comandos del sistema, y la única esperanza cierta sólo la proporciona:

a)Tratar de estructurar el sistema de forma que el menor número de programas posible sean capaces de ejecutarse con privilegios de los usuarios del sistema root, daemon, o bin, y el que lo necesite, tenga comprobada su seguridad.

b)Suscribirse a una lista de correos electrónico que permita conseguir detalles de los problemas y/o parches adecuados tan pronto como sea posible, y entonces actuar nada más recibir la información.

3. Incompatibilidad de los mecanismos de seguridad. En ocasiones, debido a la falta de experiencia , o por otros motivos, el administrador del sistema integran una combinación de equipo físico y lógico que cuando se utiliza conjuntamente es auténticamente peligroso desde el punto de vista de la seguridad. Es la incompatibilidad de tratar de hacer dos cosas útiles pero no conectadas lo que crea la vulnerabilidad del sistema.

Problemas como éstos dificultan el encontrar un sistema que funcione, por lo que suele ser recomendable construirlo uno mismo con las directrices que se están exponiendo aquí.

4. Elegir una filosofía adecuada, y mantenerla. Según Spafford, suele ser una cuestión de percepción y entendimiento, ya que ni el equipo lógico, ni el equipo físico, ni la perfecta compatibilidad de los componentes funcionan a menos a menos que se hayan seleccionado una adecuada política de seguridad y se utilicen los elementos

disponibles para reforzarla El mejor mecanismo de contraseña es inútil si los usuario creen que su nombre al revés es una buena contraseña La seguridad depende de la política y la operación del sistema en consonancia con esa política

8.3.3 ¿ Cuáles son las herramientas que ayudan a la seguridad ?

1. COPS, que es un conjunto de procedimientos de comandos adecuados, elaborados por Dan Farmer y que forman una batería de pruebas de seguridad. Tiene un rudimentario averiguador de contraseñas y procedimientos que comprueban el almacenamiento de archivos en disco . en busca de cambios sospechosos en el bit s de programas, otros comprueban los permisos de los ficheros esenciales del sistema y del usuario . e incluso miran si algún programa del sistema puede comportarse de forma que cree problemas.

   Este paquete se suministra en dos versiones, una escrita en perl y otra, sustancialmente equivalente, en procedimientos de comandos. Esta última está bastante actualizada según se van

   conociendo nuevos agujeros en la seguridad de Unix.

2. Crack, escrito por el citado autor del archivo de preguntas frecuentes, Alec David Muffet es un programa para averiguar contraseñas.
3. npasswd, de Clyde Hoovery fpasswd+, de matt Bishop.

Estos programas están escritos para ajustar el balance en la guerra de averiguación de contraseñas. Proporcionan un sustituto del comando estándar passwd , pero previendo que el usuario seleccione algunas que pueden ser fácilmente averiguadas por programas como el citado Crack.

Existen varias versiones de estos programas disponibles en Internet, adaptados en diversos grados para proporcionar compatibilidad con los sistemas basados en System V, NIS/YP, y esquemas de contraseñas gemelas . Asimismo, proporciona un control de acceso a las

terminales, usuarios y grupos para la administración El termino más usual para determinar este tipo de programas, es el de " fascista"

4 " shadow-a Shadow Password Suite ". Escrito por John F Haugh II, es un conjunto de programas y funciones reemplazables , compatibles con la mayoría de los sistemas Unix, que implementa contraseñas gemelas, es decir, un sistema que reemplaza el archivo de contraseñas /etc/passwd, haciendo que su contenido se oculte para todos los usuarios excepto para root, consiguiendo así hacer imposible de raíz el éxito en el intento de averiguarlas Si se utiliza en combinación con una contraseña " fascista ", proporciona un alto grado de seguridad de las contraseñas.

5 TCP Wrappers , por Wietse Venema. Son algunos programas que proporcionan un filtro cortafuegos para muchos de los servicios de red que Unix proporciona por defecto. Si se instala, pueden contener acceso que de otra manera no estarían restringidos, como entradas desde ftp, tfpt y telnet, proporcionando más información sobre el programa de conexión login, lo cual puede ser útil si se observa que alguien trata de entrar al sistema.

6 SecureLib, por Williams LeFebvre (phil@pex.eecs.nwu.edu). Contiene unas funciones con las que sustituir tres llamadas al sistema operativo accept() recvfrom() recvmsg(). Estas sustituciones resultan compatibles con las originales, con la funcionalidad adicional de que comprueban la dirección Internet de la máquina que inicia la conexión para asegurarse de que está permitido conectarse. Un fichero de configuración define cuáles son las computadoras autorizados para utilizar determinados programas. Una vez que estas llamadas al sistema operativo han sido reemplazadas, se compilan y pueden ser utilizadas para una nueva librería compartida del sistema libc Por otro lado, la nueva librería segura del sistema libc.so puede ubicarse en un directorio especial. Cualquier programa que deba ser protegido, puede iniciarse a través de un valor alternativo en la variable del entorno LD_LIBRARY_PATH del programa montador ld de Unix que utilizan todos los compiladores.

7 SPI A este programa, en principio, sólo tienen acceso las computadoras de organizaciones conectadas con el Departamento de Energía estadounidense y algunas organizaciones militares.

Para obtenerlo, hay que contactar con el Laboratorio Nacional Lawrence Livenmore .

SPI es una herramienta para el administrador que está basada en pantallas mediante las cuales

puede comprobar las opciones de configuración, incluyendo los cambios en ficheros (integridad)

buscando puertas traseras y virus, y otras comprobaciones de seguridad. Desde las versiones

iniciales se contemplaba la posibilidad de integrarlo con COPS, y también existe una versión para VMS.

8.3.4 ¿ No es peligroso proporcionar herramientas averiguadoras a cualquiera?

Esto depende del punto de vista. Algunas personas consideran que dar acceso público no restringido a programas como COP y Crack es una irresponsabilidad, puesto que los malhechores pueden obtenerlo fácilmente.

Pero también es cierto que los auténticamente se pueden considerar delincuentes ya disponen de estos programas desde hace años, por lo que puede ser una estupenda idea proporcionárselo también a los administradores de sistemas, y que de esta forma puedan comprobar la seguridad de sus sistemas antes de que lo hagan los delincuentes.

8.3.5 ¿ Dónde se pueden conseguir?

- COPS v 1.04, por ftp del servidor cert.sei.cmu.edu en el directorio pub/cops y en el servidor archive.cis.ohio-state.edu dentro del directorio pub/cops.
- Crack v 4.1f y UFC Patchlevel 1. Disponible en cualquier gran servidor de USENET, como por ejemplo ftp.uu.net y en el volumen 28 del grupo de noticias comp.sources.misc.
- npasswd. Actualmente parece estar en casi permanentes revisión por mucha gente, por lo que conviene consultar a los autores.
- passwd+: en el servidor dartmouth.edu como archivo comprimido pub/passwd+.tar.Z

- shadow: al igual que Crack , en el directorio comp.source.misc y, en cualquier gran servidor de archivos USENET.
- TCP Wrappers : mediante ftp anónimo en cert.sei.cmu.edu: pub/network_tools/tcp_wrapper.shar y en ftp.win.tue.ni: pub/security/log_tcp.shar.Z
- Securelib: en el servidor eecs.nwu.edu, como archivo pub/securelib.tar

8.3.6 ¿ Cómo se consigue entrar en los sistemas ?

Es difícil dar una respuesta definitiva a esta pregunta. Muchos sistemas accedidos sólo lo han sido para ser utilizados en la entrada a otros sistemas. Intentándolo en muchas máquinas antes de hacerlo en una nueva, el pirata espera conseguir cualquier posible perseguidor y despistarlo por completo. Puede conseguirse una ventaja intentando entrar desde tantos sitios diferentes como sea posible, con l propósito de blanquear las conexiones.

Otra razón pueda ser psicológica. A algunas personas les encanta jugar con servidores buscando sus limites y sus posibilidades. Algunos piratas pueden llegar a pensar que es auténticamente excitante estar en más de seis máquinas Internet, dos pasarelas y una red de X.25 simplemente para llamar a la puerta de alguna famosa empresa o institución (NASA, CERN, AT&T,UCB). Hay que completarlo como una introspección en la red. Ésta es una fuerte atracción para algunos piratas, y ciertamente lleva a la adicción y a la autoperpetuación de la piratería.

Y el " cómo " de la cuestión vuelve a ser una materia muy delicada. En las universidades es una práctica extremadamente corriente el que las cuentas se presten entre estudiantes.

Esta clase de situaciones ocurren con frecuencia, y no sólo en las universidades. Una solución , tal vez la única, esta en la formación. No se puede permitir a los usuarios actitudes como ésta:

> " No importa que contraseña utilizo para mi cuenta, puesto que ,
> después de todo, yo sólo la utilizo para imprimir con láser "

Hay que enseñar que el uso de un servidor es una responsabilidad de grupo. Hay que estar seguros de que se comprende que una cadena es tan fuerte como su eslabón más débil.

Finalmente, si los demás comprenden los problemas del administrador del sistema, y simpatizan totalmente con él, deberá configurar el sistema de forma que nadie pueda equivocarse. Hay que creer en la educación del usuario, pero no confiar sólo en ella.

8.3.7 ¿ Con quien se debe contactar si alguien ha entrado ya?

Si la máquina está conectada a Internet , es recomendable ponerlo en conocimiento del CERT, de la forma descrita en la sección anterior.

8.3.8 ¿ Qué es un cortafuegos ?

En Internet , se entiende por cortafuegos a una máquina que normalmente está conectada entre una instalación e Internet. De esta forma se consigue un filtro controlador del tráfico de la red, permitiendo el acceso a ciertos puertos de Internet. Por ejemplo, para disponer de forma más restrictiva de los mismos servicios de la máquina hubiera proporcionado a toda la red, y acceso a bloques específicos , y sólo a ellos. Hay máquinas similares disponibles para tipos de redes basada en sistemas diferentes a Unix.

Los contrafuegos proporcionan una aproximación " todo o nada " muy efectiva, que se está haciendo cada vez más popular para conectarse con seguridad.

8.3.9 ¿ Por qué no se debe utilizar procedimientos de comandos con el bit s a nivel dueño ?

Existen múltiples razones, la mayoría relacionadas con defectos detectados en el núcleo de Unix. Estos son algunos de los problemas mejor conocidos, que las versiones del sistema operativo más reciente han intentado corregir:

1. Si el archivo de comandos empieza por el mandato #!/bin/sh y pude establecerse un sinónimo simbólico o de naturaleza, a través del comando ln, con el nombre -i, puede mantenerse inmediatamente un archivo de comandos con el bit s activo a nivel de dueño, puesto que invocándole así, #!/bin/sh -i, se consigue un interprete de comandos interactivos.
2. Muchos núcleos de sistema operativo sufren condiciones de carrera, es decir, accesos concurrentes no ordenados, que pueden permitir cambiar un archivo de comandos por otro ejecutable a elección del usuario entre el momento en el que el nuevo proceso lanzado a través de una llamada al sistema operativo exec() va establecer el bit s, y el instante en el que el intérprete de comandos arranca. Si se es suficientemente persistente , en teoría puede utilizarse el núcleo para ejecutar cualquier programa que se desee.
3. El defecto de IFS. La variable del entorno del intérprete de comandos IFS contienen una lista de caracteres para ser tratados como espacios por un programa de dicho tipo cuando se analizan los nombres de los comandos. Cuando la variable IFS para que contenga el carácter /,el comando /bin/true se convierte en bin true. Todo lo que se precisa es exportar la variable IFS modificada , instalar un comando llamado bin en un camino ejecutable, y lanzar un fichero de comandos con bit s que se llamará /bin/true. De esta forma se conseguirá que el programa bin se ejecute como si tuviera bit s.

Si realmente se necesita escribir algún fichero de comandos con el bit s, es necesario:

a) Escribir un programa en lenguaje C con el atributo s que ejecute el archivo de comandos. Este programa inicializará las variables IFS y PATH con valores seguros antes de que se ejecute, a través de la llamada al sistema operativo exec(), el archivo

de comandos. Si el sistema tiene un soporte en tiempo de ejecución con librerías enlazadas, también debe considerarse la variable del montador LD_LIBRARY_PATH.

a) Utilizar un lenguaje de comandos como Perl, que tiene una utilidad segura para el manejo del bit s, y que es, en general, muy celoso de la seguridad.

Pero en realidad, lo más seguro es no utilizar ningún fichero de comandos con el atributo s a nivel de dueño.

8.3.10 ¿ Por qué no se debe dejar la cuenta root permanentemente conectada en la consola ?

Utilizando un terminal "inteligente" como consola, y si se ha dejado su dispositivo correspondiente /dev/console con permisos de escritura para todo el mundo, mientras la cuenta del administrador está conectada, existe un agujero potencial. El terminal puede ser vulnerable por control remoto a través de secuencias de escape, y puede ser utilizado para " teclear " muchas cosas dentro del intérprete de comandos del usuario privilegiado. El tipo de terminal puede ser conseguido usualmente a través del comando de usuario ps.

Se han previsto varias soluciones para este problema, y la más general es sólo dar permisos de escritura en la consola al propietario y al grupo, y utilizar el mecanismo de establecer el bit s a nivel de grupo sobre cualquier programa que tenga la necesidad de mostrar su salida por la consola, como por ejemplo,write.

8.3.11 ¿ Por qué no se deben crear cuentas Unix con palabras nulas ?

Crear cuentas sin contraseñas para cualquier propósito es potencialmente peligroso, no por ninguna razón directa, sino sólo porque pueden mostrar a un intruso un talón de Aquiles.

Por ejemplo, en muchas instalaciones puede encontrarse un usuario sin contraseña llamado sync, que permite al operador del sistema sincronizar los discos sin tener que conectarse al servidor. Esto es seguro.

El problema aparece si el sistema es uno de los muchos que no comprueba al usuario antes de utilizarlo, por ejemplo, al solicitar ftp. Un pirata podría ser capaz de conectarse a la máquina mediante una amplia variedad de métodos basados en ftp, pretendiendo ser el usuario sync sin contraseña, y así copiar el archivo de contraseñas para realizar el ataque al mismo en su propio servidor.

Pese a que existen mecanismos para evitar que ocurra esto en la mayor parte de las modernas versiones de Unix, el estar completamente seguro obliga a tener un conocimiento muy profundo de cada paquete del sistema, y de cómo se produce la verificación de usuarios. Si no es posible asegurarse de ello por completo, es mucho mejor no dejar vulnerabilidades como ésta.

Otra vulnerabilidad se produce cuando s dejan cuentas sin contraseñas abiertas y existe la posibilidad, en sistemas con soporte de ejecución de librerías enlazadas, de engañar a los programas del sistema cambiando la variable LD_LIBRARY_PATH a una librería a conveniencia del pirata, y ejecutando el comando de conexión login -p o el comando de cambio de identidad para suplantar a dicho usuario.

8.3.12 ¿ Cuáles son los agujeros de la seguridad relacionados con X - Windows (y con otros sistemas de ventanas)?

Muchísimos, algunos de los cuales sólo afectan a X. y otros tienen impacto en toda la seguridad del sistema.
Puede encontrarse información abundante en algunos grupos de noticias, como comp.windows.

Pero sí es de reseñar que X es uno de esos paquetes que frecuentemente generan " incompatibilidad de uso" en los problemas de seguridad, por ejemplo, la facilidad que ofrece a los piratas para ejecutar sesiones X en servidores bajo cuentas sin contraseñas, como por ejemplo la de sync, si no se ha configurado correctamente. Debe considerarse todo lo ya expuesto el el apartado sobre cuentas sin contraseña.

8.3.13 ¿ Cómo se puede generar contraseñas segura ?

En rigor, no se puede. La palabra aquí es generar. Una vez que se ha especificado un algoritmo para generar contraseñas y se utiliza con un método sistemático, se convierte en una simple cuestión de análisis el encontrar todas y cada una de las contraseñas del sistema.

A menos que el algoritmo sea muy sofisticado, probablemente presente un período muy corto, es decir, que pronto comience a repetirse, de manera que :

a) Un pirata puede intentar entrar con cada posible salida del generador de contraseñas de cada usuario del sistema, o
b) El pirata puede analizar la salida del programa de contraseñas, determinar el algoritmo que se ha utilizado, y aplicarlo a otros usuarios para determinar sus contraseñas.

Un buen ejemplo de ello, en el que se ha asumido el imposible de que un generador de números aleatorios pueda generar un número infinito de contraseñas, se ha detallado en el clásico artículos de Morris y Thompson " Password Securrity, A Cas History ".

La única forma de conseguir una razonable originalidad en nuestras contraseñas, en opinión de Alec
David Muffett, es no intentar ser demasiado original. Lo mejor es trabajar con un método flexible propio, que no esté basado en:

1) Modificar cualquier parte del nombre del usuario más sus iniciales.
2) Modificar una palabra del diccionario.
3) Acrónimos.
4) Cualquier sistemática, dependiente de un algoritmo o similar.

8.3.14 ¿ Por qué son tan importantes las contraseñas?

Pues porque son la primera linea de defensa respecto a cualquier ataque interactivo contra el sistema. Es asi de simple: si un pirata no puede entrar en el sistema, no tiene forma de leer o escribir en el archivo de contraseñas, y por lo tanto, no tiene ninguna forma de atacarlo.

Ésta es también la razón por la que si un pirata al menos puede leer el archivo de contraseñas, y en la mayoria de los sistema Unix esto es difícil de evitar, es muy importante que el pirata no sea capaz de descifrar las contraseñas que contenga. Si puede hacerlo, también es fácil asumir que pueda entrar en el sistema y convertirse en su administrador a través de algunos de las vulnerabilidades de la seguridad.

8.3.15 ¿ Cuantas contraseñas posibles hay ?

La mayor parte de los usuario se preocupan porque programas como Crack puedan aumentar su potencia para realizar búsquedas exhaustivas sobre todas las contraseñas posibles, hasta adivinar la de una cuenta concreta, generalmente la del usuario privilegiado.

Si se asumen las siguientes hipótesis:

1) Las contraseñas válidas se crean con un juego de 62 caracteres `[A-Z a-z 0-9].
2) Las contraseñas válidas tienen entre 5 y 8 caracteres de longitud, el conjunto de posibles contraseñas es de 100000+1000000+10000000+100000000= 111100000 (base 62). Es decir, un número demasiado grande para ser abordado por las tecnologías actualmente disponible. No se puede olvidar , sin embargo, que las contraseñas pueden estar constituidas por más caracteres todavía, y que se pueden utilizar espacios, y todos los caracteres de puntuación y símbolos como (Ç,><,}#%) también. Si es posible utilizar algunos de los 95 caracteres que no son de control en las contraseñas, esto aumentará más aún las dificultades a las que tendrá que enfrentarse el pirata.

Sin embargo, todavía es mucho más eficiente para un pirata conseguir una copia del Crack para romper cualquiera de las cuentas del sistema, ya que sólo se necesita una para entrar en él, y conseguir los privilegios del administrador aprovechando alguna de las vulnerabilidades del sistema operativo.

CAPITULO 9

Seguridad Futura

9.1 Novell/USL Unix SVR4 Enhanced Security.
9.2 Kerberos: seguridad en sistemas distribuidos.

9.1 Novell/USL Unix SVR4 Enhanced Security

Novell/Unix System Laboratories (Novell/USL) Unix System V Release 4 Enhanced Security, a partir de ahora Enhanced Security , ha sido diseñado para cumplir la creciente necesidad de seguridad que demanda la comunidad Unix. Al estar certificándose en el nivel de seguridad B2 respecto al Libro Naranja y en F4/Q4 a través de la Agencia de Seguridad de la Información Alemana satisface la mayoría de los requisitos que solicitan los usuarios comerciales y gubernamentales: control de acceso obligatorio, vías fiables, utilidades para la administración y lista de control de accesos.

Los niveles de seguridad superiores (B3/A1) proporcionan pocas funcionalidades adicionales, centrándose fundamentalmente en asegurar las ya existentes. Asimismo, B2 es el máximo nivel de seguridad que puede alcanzar un sistema Unix manteniendo su apariencia clásica y la compatibilidad con las aplicaciones existentes.

9.1.1 Privilegio mínimo

Originalmente los sistemas Unix eran pequeños y no funcionaban en red. El concepto de privilegio único era adecuado porque los servidores se administraban y mantenían por una persona. Según aumentó su complejidad, las actividades de operación, administración y mantenimiento se dividieron entre varias personas realizando funciones especializadas:

- Un operador del sistema es responsable de las acciones habituales, mientras que el administrador se preocupa del mantenimiento.
- Las modificaciones del sistema operativo puede realizarlas un programador de sistemas.
- En un centro de proceso de datos, la facturación por el uso de recursos informáticos puede realizarla un auditor del sistema.
- Un responsable de seguridad puede ocuparse de revisar los registros de auditoría y contabilidad para determinar si se ha realizado algunas actividades sospechosas.

Las tareas realizadas en cada una de estas funciones son diferentes, y el nivel de confianza que debe depositarse en cada persona es distinto . Por ejemplo, quién examine la contabilidad del sistema no debe estar autorizado para detener el servidor o visualizar los archivos de otros usuarios. El modelo de privilegio único basado en un superusuario no encaja correctamente en un esquema de privilegios basado en las acciones que deben realizar distintos usuarios.

La alternativa es el concepto de privilegio mínimo, o la habilitación para realizar una acción con la cantidad mínima de privilegios que permitan realizarla satisfactoriamente. Esto beneficia a aquellas situaciones en las que una persona debe realizar una actividad concreta sin permitírsele realizar por defecto un amplio conjunto de tareas. Por ejemplo, a un operador puede permitírsele establecer la fecha y hora del servidor, pero no acceder a archivos sensibles del sistema. Tan sólo se permite que el operador realice un conjunto bien definido de comandos privilegiados. Las acciones que sean ajenas a la función del operador no puede ejecutarlas, o si lo hace, es sin privilegio alguno. Históricamente Unix tan sólo ha proporcionado una entidad privilegiada, el superusuario, y todas las actividades que requerían algún tipo de privilegio debían realizarse a través de esta identidad. Por este motivo era difícil la delimitación de responsabilidades para cada función desempeñada en el sistema.

A través del privilegio mínimo se dividen las funciones del superusuario en un conjunto de funciones bien definidas y menos potentes: operador del sistema, de seguridad, responsable de seguridad y auditor. Cada instalación puede añadir nuevas funciones o ampliar las existentes. Cada función tiene un conjunto bien definido de operaciones privilegiadas que pueden realizarse.

9.1.2 Utilidad para la administración de la seguridad.

Esta utilidad permite redefinir la forma en la que funciona el mecanismo de asignación de privilegios y funciones. En los sistemas Unix actuales del administrador asume todos los privilegios, por tanto, los procesos que ejecuta también los adquieren. En esta situación existen puntos vulnerables.

Supóngase que teniendo el administrador privilegio de superusuario llama a los programas:

$ date 030993	cambiar la fecha del servidor.
$ mail	visualizar el correo electrónico

El administrador cree que se están ejecutando los comandos de usuario /bin/date y /bin/mail. Sin embargo, puesto que no se ha especificado el camino completo del archivo ejecutable, se utilizan los caminos actuales de la variable de entorno PATH para encontrar los ejecutables. Si en esta variable hay un camino $HOME/bin, el administrador podría estar ejecutando un caballo de Troya de estos comandos que obtienen los privilegios de superusuario.

Hay otro problema debido al mecanismo de herencia asociado al nacimiento de un proceso. En Unix C1 todos los atributos asociados a la identidad del superusuario pasan al proceso hijo, por lo que los programas a los que llama el administrador tienen estos privilegios, tanto si los necesita como si no. Como consecuencia, se ejecuta un código con unos privilegios no previstos que carece de consideraciones de seguridad. Esta situación es especialmente peligrosa con programas que permiten ejecutar a otros, o incluso tienen una salida hacia el intérprete de comandos. Por ejemplo, si un administrador que ejecuta el correo electrónico , sale temporalmente al intérprete de comandos y llama a un programa para visualizar archivos, éste tiene privilegios de superusuario y, si fuera un caballo de Troya, adquiriría privilegios no previstos.

Mediante el uso de comandos de usuario *tfadmin* no hay privilegios asociados a un usuario, sino que vinculan a una función, y los usuarios se asocian a ella. Para adquirir un privilegio hay que ejecutar el comando de usuario *tfadmin* , que posee un base de datos con la siguiente estructura:

función: seudónimo:*comando*:*privilegio(s)*

Por ejemplo:

secadmin:date:/bin/date:p_sysops

Por otro lado, se asignan funciones a los usuarios, por ejemplo, *antonio* puede ser el administrador de seguridad (secadmin), y cuando en esta situación ejecuta el comando de usuario /bin/date, que puede escribir como date, lo hace con el privilegio p_sysops.

Si se considera de nuevo el ejemplo anterior:

$ tfadmin date 030993
$ mail

En primer lugar, tfadmin busca en su base de datos una entrada para date. Si la encuentra, se ejecuta el comando indicado en la base de datos con los privilegios que en ella figuran, en este caso *p sysops*, que es el necesario para modificar la fecha y hora del servidor . Éste es el único privilegio que se proporciona al proceso que ejecuta date. Como la ejecución del mail no requiere privilegio alguno, es innecesario su ejecución vía *tfadmin*.

9.1.3 Control de acceso obligatorio.

En Enhanced Security se han añadido etiquetas para garantizar la integridad y confidencialidad de los datos . Todos los procesos, archivos y mecanismos para la comunicación entre procesos (tuberías, colas de mensajes, semáforos, memoria compartida) poseen una etiqueta de confidencialidad. Con un mecanismo de control de acceso discrecional los permisos se asignan según los criterios del dueño del objeto, por tanto, el propietario de un archivo puede decidir quién accede al mismo.

Por el contrario, cuando el administrador del servidor establece un mecanismos de control de accesos obligatorio, el sistema garantiza su aplicación, y el dueño de un archivo no puede cambiar su etiqueta de confidencialidad.

La política de control de acceso obligatorio que implementa Enhanced Security es una modificación del modelo de Bell-LaPadula , y puede resumirse como " leer igual o inferior " y " escribir igual ". Considérese una situación en la que un documento ALTO SECRETO domina a SECRETO, y éste a DESCLASIFICADO. Cada nivel tan sólo puede escribir en sus archivos, pero un proceso en el nivel ALTO SECRETO puede leer los archivos de su clase y de los que domina. Por ejemplo, un proceso en el nivel SECRETO puede leer archivos de los niveles SECRETOS y DESCLASIFICADOS.

Los administradores determinan y establecen el conjunto discreto de etiquetas a las que un usuario puede acceder, así como el rango de niveles de conexión de las líneas para las terminales, lo cual garantiza la conexión al servidor de los usuarios cuya etiqueta domina a la de la terminal. Por ejemplo, una línea SECRETA es inaccesible por un usuario DESCLASIFICADO.

Por defecto Enhanced Security soporta 256 clasificaciones (niveles de confidencialidad de la información, por ejemplo: secreto, reservado, confidencial, difusión limitada) y 1024 categorías

(agrupaciones de la información, por ejemplo: nóminas , almacén, producción, diseño), aunque el sistema puede configurarse hasta 65,535 clasificaciones y 2099152 categorías. A cada etiqueta (clasificación más categoría) se le asocia un identificador de nivel que se utiliza como apuntador al valor de la etiqueta almacenada en el nodo índice de cada archivo. Para conveniencia de los usuarios a cada etiqueta puede asignársele un seudónimo. Por ejemplo para la etiqueta:

ALTO SECRETO: proyectoA, proyectoB
puede ser
AS

El núcleo del sistema operativo utiliza el identificador como método primario para referencias a las etiquetas . Cuando se solicita al núcleo comprobar un acceso de escritura se comparan los dos identificadores, ya que el sistema garantiza una política de " escritura igual ", y los identificadores son únicos.

Por ejemplo, si un proceso cuyo identificador es el 10045 solicita escribir en un archivo cuyo identificador es 10045 se concede el acceso puesto que ambos valores son iguales, por el contrario si el proceso con identificador 10046 solicita el acceso en lectura, no es suficiente con la comparación de los identificadores. Puesto que el sistema soporta una política de " lectura inferior ", la comprobación de este tipo de acceso requiere una etapa adicional. Mediante la comprobación de dos identificadores no puede desprenderse una relación jerárquica , por ejemplo, no puede garantizarse que el identificador 10046 domine al 10045. Esto implica la necesidad de comprobar las dos etiquetas completas, y no sus identificadores. Por razones de rendimiento la representación binaria de las etiquetas se almacena en una " cache " cuyo tamaño puede ajustarse según los requisitos de cada sistema. Por ejemplo, si el proceso con identificador 10046 solicita una operación de lectura sobre un archivo cuyo identificador es el 10045 ocurre lo siguiente:

- Se comprueba si las representaciones binarias de las etiquetas por comparar están en el "cache".

- Si no lo están, el sistema las busca en la base de datos de identificadores y carga su representación binaria en la " cache ".
- Se comparan las representaciones de las etiquetas para determinar si existe una relación de dominancia que permita el acceso de lectura. Si existe se otorga, si no , se impide.

9.1.4 Aislamiento de accesos.

Existe una forma adicional de integridad de los datos mediante el aislamiento de acceso realizando a través del uso de niveles en el control de acceso obligatorio. Se definen dos jerarquías independientes de etiquetas : una para el usuario, y otra para el sistema. De esta manera, se prohibe a los usuarios la lectura, modificación o ejecución de archivos sensibles del sistema, y se protege a los administradores respecto a la ejecución de aplicaciones inseguras.

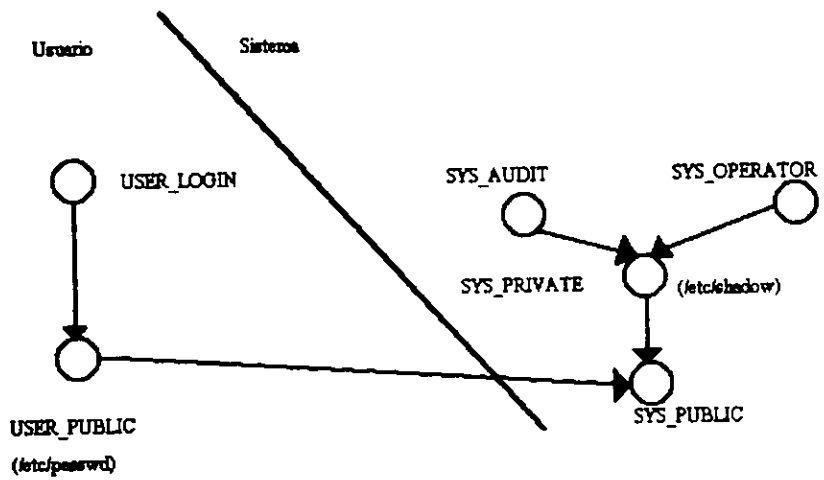


Figura 9.1 Mecanismos para el aislamiento de acceso.

Tal y como se muestra en la figura 9.1, el nivel USER_PUBLIC está definido para archivos de usuario y comandos no administrativos. Todos los usuarios no administrativos cuando se conectan al servidor lo hacen en el nivel USER_LOGIN. Los niveles SYS_PUBLIC, SYS_PRIVATE, SYS_OPERATOR y SYS_AUDIT se definen para uso administrativo y del sistema. En SYS_PUBLIC están los archivos y comandos accesibles tanto a los administradores como a los usuarios. El nivel SYS_PRIVATE se define para el acceso administrativo y no es accesible para los usuarios normales. En el nivel SYS_AUDIT, se almacena el registro de la auditoría del sistema.

Comandos de usuario como date y mail podrían etiquetarse como SYS_PUBLIC. Puesto que tanto los usuarios como el sistema tienen acceso de lectura a estos archivos, todos ellos pueden ejecutar estos comandos. Pero como los usuarios no tienen permiso de escritura en este nivel , tampoco pueden depositar un caballo de Troya. Un administrador en el nivel SYS_PRIVATE no domina ni a USER_PUBLIC ni a USER_LOGIN, y puesto que no obtiene los privilegios necesarios para burlar el control de acceso obligatorio, está protegido contra la ejecución de caballos de Troya ubicados en esos niveles por los usuarios.

9.1.5 Control de acceso discrecional

Enhanced Security proporciona dos mecanismos complementarios para el control de acceso discrecional: los permisos que proporcionan Unix a los archivos, y listas de control de acceso, que están diseñadas para satisfacer el nivel B3 del Libro Naranja. Éstas permiten que el dueño de un objeto conceda o deniegue el acceso a cada usuario o grupo del servidor. Las listas pueden ser tan largas como se desee, aunque el administrador puede establecer un máximo a través de un parámetro configurable; cuanto mayor es la lista , más lento es su tratamiento, por lo que hay un límite práctico a su tamaño. A cada archivo y mecanismo para comunicar procesos se le asocia una lista que comienza en su nodo índice.

Por compatibilidad con las versiones anteriores de Unix System V, los permisos de los archivos se muestran y manipulan de la forma usual. Sin embargo, los permisos se almacenan como entradas de la lista para el control de acceso. Así, los permisos correspondientes al dueño del archivo y otros

usuarios constituyen las entradas base de la lista. La concesión o revocación de nuevos permisos se realizan mediante la inclusión de entradas adicionales simbolizadas a través del carácter " + ". Históricamente en Unix han existido tres permisos para representar lo que el grupo dueño del archivo podía hacer con él. En Enhanced Security estos permisos son la clase del grupo del archivo, y se utilizan para representar los máximos permisos que proporcionan las entradas adicionales de la lista para el control de acceso. A continuación se describen estos conceptos mediante un ejemplo: los números entre paréntesis indican asociaciones , y no aparecen en la lista real.

```
(4)   (5)   (6)   (2)   (3)   (1)
rwx   r-x   r-x+  pepe  demo  ejemplo

# archivo:ejemplo                (1)
# dueño:pepe                     (2)
# grupo:demo                     (3)
usuario::rwx                     (4)

usuario:ana—x
grupo::r-x                       (5)
grupo:sys:—

clase:r-x
otros:r-x                        (6)
```

Figura 9.2 Ejemplo de la lista para el control de acceso

En la lista de control de acceso de la figura anterior, las tres primeras entradas indican el nombre del archivo (ejemplo), el dueño del archivo (pepe) y el grupo dueño del archivo (demo). Las entradas de dueño (4) y de grupo (5) sin nombre de usuario representan los permisos del dueño y del grupo dueño del archivo respectivamente. La entrada clase se calcula mediante la operación "o" de las entradas adicionales de la lista ana, el grupo dueño del archivo, y sys, que están agrupadas en la figura mediante un rectángulo.

Respecto al ejemplo anterior, obsérvese que las entradas de dueño, clase y otros se modifican para reflejar los cambios de permisos indicados mediante el comando de usuarios chmod . No se modifica ninguna entrada adicional. Obsérvese que las entradas de clase se ha modificado y usado como una máscara, mientras que la entrada del grupo dueño no se ha visto alterada, puesto que en Enhanced Security se considera que es adicional.

9.1.6 Vía fiable

Proporciona un camino que garantiza al usuario que todo lo que escribe llega sin alteración al sistema. Para ello, antes de conectarse al servidor, el usuario pulsa la tecla de atención asegurada. Cuando el sistema la reconoce interrumpe todos los procesos asociados a la terminal e inicia una secuencia de conexión. La vía fiable ofrece protección contra ataques de accesos al sistema y frente a caballos de Troya.

Una forma habitual de atacar a los accesos de un sistema es mediante un programa que marca números de teléfonos, y espera hasta que el servidor le envía el mensaje de conexión al sistema. A partir de este momento, el programa ensaya combinaciones de nombres de conexión y contraseñas hasta que consigue acceder al servidor. Pero si para enviar el mensaje de conexión se solicita una secuencia especial de caracteres de control configurable, la mayoría de estos programas dejan de ser efectivos.

Un caballo de Troya habitual es el que simula el programa de conexión del sistema, y se ejecuta por un usuario autorizado desde una terminal del servidor. Cuando otro usuario llega a esa terminal, el programa suplantador se queda con su nombre y contraseña sin que sea consciente de ello. Por el contrario, si el usuario pulsa la tecla de atención asegurada muere el suplantador y el usuario entra sin peligro alguno en el ordenador.

En Enhanced Security la conexión fiable funciona de la siguiente manera:

1. Un usuario que desea acceder al sistema pulsa la tecla de atención asegurada.
2. El sistema identifica la tecla antes de aplicar cualquier modalidad de control sobre la línea de comunicaciones.
3. Tras detectar la tecla de atención, el mecanismo de seguridad concluye la sesión de trabajo actual, y coloca la línea en un estado en el que no puede utilizarse para realizar operaciones de entrada/salida. A continuación lanza una secuencia de conexión.
4. Si no se completa la conexión en un plazo predeterminado, entonces es necesario volver a pulsar la tecla de atención para reiniciar la conexión.

9.2 Kerberos: seguridad en sistemas distribuidos.

En un sistema basado en computadoras personales no interconectadas la información puede protegerse asegurando físicamente a las computadoras. En un entorno multiusuario el sistema operativo controla los recursos, y protege a los usuarios mediante su identificación y autenticación durante la conexión al servidor. Por el contrario, un sistema distribuido se caracteriza porque los usuarios conectados a un servidor (cliente) solicitan servicios ubicados en otras computadoras (servidores). En esta situación, ¿cómo puede controlar sus recursos un servidor que recibe una petición procedente de un usuario que está conectada en otra computadora ? Existen tres posibilidades:

1. No hacer nada. Se asume que el servidor en el que trabaja el cliente es suficientemente seguro para prevenir el acceso no autorizado.
2. Solicitar que se identifique la computadora cliente.
3. Exigir que el usuario cliente se identifique siempre que solicite un servicio.

La primera alternativa puede utilizarse en un entorno cerrado en el que todos las computadoras se encuentran bajo control. La segunda es aplicada sólo cuando es suficiente la comprobación de la identidad de las computadoras de una empresa. La tercera es necesaria en un sistema distribuido abierto en el que existen computadoras donde los usuarios pueden configurar la información relevante para la seguridad.

Un mecanismo de identificación para un sistema distribuido abierto debe cumplir cuatro requisitos:

1. Inevitable: para que un atacante le resulte difícil de soslayar.
2. Fiable: todos los servicios de un sistema distribuido se basan en el de autenticación. Si éste falla también lo harán los demás.
3. Transparente: el usuario no debe saber que se está realizando su autenticación.
4. Incremental: debe funcionar aunque existan aplicaciones que no utilicen este mecanismo.

9.2.1 Definición.

Kerberos es el servicio de autenticación desarrollado por el Instituto de Tecnología de Massachusetts disponible bajo OSF/DCE para garantizar la seguridad en sistemas distribuidos abiertos.

Para probar la identidad de un usuario durante la sesión de trabajo a todos los servicios del sistema distribuido es suficiente con el nombre y la contraseña que se introduce al conectarse. La seguridad de Kerberos se basa en la de unos servidores de autenticación, y no en la de las computadoras a través de los cuales se conectan los usuarios, ni en la de los que proporcionan los servicios.

La autenticación es una pieza fundamental en un sistema distribuido seguro. Si un servidor conoce la identidad de un cliente, puede decidir sobre la concesión del servicio, la asignación de privilegios especiales, o a quien enviar la factura por su uso. Los sistemas de autorización y contabilidad pueden construirse sobre el servicio de autenticación proporcionando una seguridad equivalente a la que existe en una computadora multiusuario.

9.2.2 Funcionamiento

Kerberos mantiene una base de datos de sus clientes (usuarios o programas) con sus respectivas claves privadas (números conocidos tan sólo por Kerberos y por el cliente a quien pertenece).

Kerberos proporciona tres niveles de protección a las aplicaciones, y cada programador elige el adecuado en función de los requisitos:

1. Autenticación al comenzar una conexión en la red que soporta al sistema distribuido. Se asume que los mensajes posteriores proceden de la parte autenticada.
2. Mensajes seguros: autenticación de todos los mensajes, sin importar si se revela su contenido.
3. Mensajes privados: autenticación y cifrado de todos los mensajes.

9.2.3 Componentes software

En la figura 9.3 se muestra la arquitectura de Kerberos.

Biblioteca de aplicaciones de Kerberos
Biblioteca de cifrado
Biblioteca base de datos
Programa para la administración de base de datos
Servidor de administración
Servidor de autenticación
Software para la propagación de la base de datos
Programas de usuario
Aplicaciones

Figura 9.3 Componentes de Kerberos

La biblioteca de aplicaciones de Kerberos proporciona una interfaz a las aplicaciones clientes y servidores; contiene funciones para leer o crear peticiones de autenticación, y para generar mensajes seguros y privados. En la biblioteca de cifrado existen funciones para manipular la información mediante varios métodos. Esta biblioteca es un módulo independiente, y puede sustituirse por cualquier otro que se desee. Otro módulo reemplazable es la base de datos, en la cual Kerberos almacena un registro por cada cliente con su nombre, clave privada, y fecha de expiración. Los servidores de Kerberos utilizan la biblioteca y las herramientas para administrar esta base de datos. El servidor de administración de la base de datos de Kerberos proporciona una interfaz de lectura-escritura entre la red y la base de datos. El servidor de autenticación realiza lecturas sobre la base de datos para autenticar a los clientes de Kerberos en varios computadoras. Por último, existen programas de usuario para conectarse con Kerberos, que permiten modificar contraseñas y manipular credenciales.

9.2.4 Credenciales.

En este apartado se describe el protocolo de autenticación que utiliza Kerberos. Para ello se utilizan las abreviaturas indicadas en la tabla 9.2.

c	Cliente
s	Servidor
dir	Dirección del cliente en la red
vida	Tiempo de vida del pase
scp, SCP	Servidores de Concesión de Pases
Kerberos	Servidor de autenticación
KXXXXxZZZ	Clave privada de x
KXXXXx,yZZZ	Clave de sesión para x e y
{abc}KXXXXxZZZ	abc cifrado con la clave de x
PXXXx,yZZZ	Pase de x para usarlo y
AXXXxZZZ	Autenticador para x

Tabla 9.2 Abreviaturas para representar los protocolos de Kerberos

La autenticación mediante Kerberos se realiza en tres fases:

1. El usuario obtiene las credenciales necesarias para acceder a otros servicios.
2. El usuario solicita autenticación para un servicio específico.
3. El usuario presenta la credencial autenticada ante el servidor final que presta el servicio específico.

Kerberos utiliza dos tipos de credenciales: pases y autenticadores. El pase se utiliza para asegurar la identidad del usuario, a quien se entregó, entre el servidor de autenticación y el servidor final. También sirve para asegurar que la persona que lo utiliza es la misma a quien se le entregó. El autenticador contiene información adicional que al compararse con la existente en el pase, demuestra que el cliente que presenta un pase es el mismo a quien se le entregó.

El pase (figura 9.4) contiene: nombre del servidor, del cliente y su dirección dentro del sistema distribuido, marca de tiempo, vida y una clave aleatoria de sesión de trabajo. Toda esta información se cifra mediante la clave del servidor para el cual se utilizará el pase. Una vez entregado, el cliente indicado puede utilizar el pase cuanto desee hasta su expiración para acceder al servidor nombrado.

{s, c, dir, marca de tiempo, vida, KXXXs,cZZZ}KXXXsZZZ

Figura 9.4 Pase de Kerberos

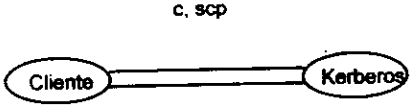
El autenticador sólo puede utilizarse una vez, por lo que el cliente debe producir uno cada vez que desee utilizar un servicio. El autenticador (figura 9.5) contiene: nombre del cliente, dirección del puesto de trabajo dentro del sistema distribuido, junto con su hora actual. El autenticador se cifra con la clave de la sesión de la cual es parte el pase.

{c, dir, marca de tiempo}KXXXs,cZZZ

Figura 9.5 Autenticador de Kerberos.

9.2.5 Obtención del pase inicial.

El proceso de conexión a un puesto de trabajo bajo Kerberos tiene una apariencia idéntica al de un sistema multiusuario, aunque su trasfondo es totalmente distinto, tal y como se muestra en la figura 9.6.



DDD{KXXXc,scpZZZ,{PXXXc,scpZZZÇKXXXscpZZZ}KXXXcZZZUUU

Figura 9.6 Obtención del pase inicial

Una vez que el usuario ha introducido su nombre, se envía una petición al servidor de autenticación, que contienen el nombre del usuario junto con el nombre de un servicio especial denominado servicio de concesión de pases. El servidor de autenticación comprueba si conoce al cliente; si es así, genera una clave aleatoria de sesión, que se utilizará posteriormente entre el cliente y el servidor de concesión de pases. A continuación, crea un pase para el servidor de concesión de pases que contiene: nombre del cliente, nombre del servidor de concesión de pases, hora actual, tiempo de vida del pase, dirección del cliente en el sistema distribuido y la clave aleatoria de sesión que acaba de crearse. Todo esto se cifra mediante una clave que tan sólo conocen el servidor de concesión de claves y de autenticación. A continuación, éste devuelve todo al cliente, cifrando la respuesta mediante la clave

privada de éste, que tan sólo es conocida por Kerberos y el propio cliente, puesto que está basada en la contraseña del usuario. Una vez que el cliente recibe la respuesta, se solicita la contraseña al usuario, que se convierte en una clave, y se utiliza para descifrar la respuesta procedente del servidor de autenticación. El pase, y la clave de la sesión se almacenan para su uso futuro, mientras que la contraseña del usuario y la clave se borran de la memoria.

9.2.6 Solicitud de un servicio

Supóngase que el usuario ya tiene un pase para el servicio que desea utilizar. Para acceder al servidor la aplicación construye un autenticador que contiene el nombre, la dirección, y la hora actual del cliente. El autenticador se cifra mediante la clave de sesión recibida en el pase para el servidor. El cliente envía al servidor (figura 9.7) el autenticador junto con el pase, en la manera que la aplicación haya establecido. Una vez que el servidor ha recibido ambos, utiliza la clave de sesión incluida en el pase para descifrar el autenticador y compara la información que contiene. Si son iguales, procede a realizar el servicio solicitado.

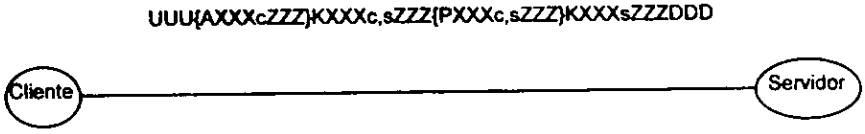
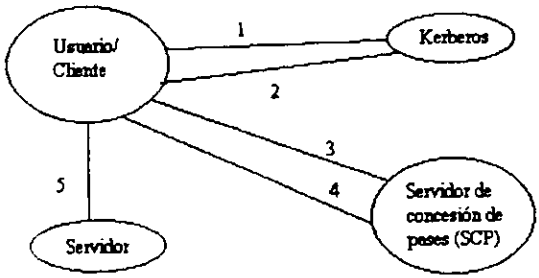


Figura 9.7 Petición de un servidor

9.2.7 Obtención de pases para servidores

Cada servicio que desee utilizar el cliente necesita un pase. Cuando un programa solicita un pase por primera vez, envía una petición al servidor de concesión de pases. Éste comprueba el autenticador el pase para la concesión; si es válido produce una clave de sesión para su uso entre cliente y servidor. A continuación se construye un pase que es enviado al cliente para que lo entregue al servidor, tal y como se muestra en la figura 9.8.



1. Solicitud de pase para el servidor de concesión de pases.
2. Pase para el servidor de concesión de pases.
3. Solicitud de pases para el servidor.
4. Pases para el servidor.
5. Petición de servicio.

Figura 9.8 Protocolo de autenticación Kerberos.

CAPITULO 10

Programación de aplicaciones seguras en lenguaje C.

10.1 Archivos de cabecera.
10.2 Funciones de biblioteca.
10.3 Recomendaciones para una programación segura.

n este capítulo se describen las llamadas al sistema y funciones de biblioteca de un sistema seguro. Obsérvese que usarlas puede hacer que las aplicaciones no sean transportables a otros sistemas, seguros o no.

El programa de ejemplo expuesto en este capítulo muestra cómo utilizar los mecanismos de protección de un sistema seguro para la programación de aplicaciones seguras, lo cual exige que desde su especificación existan requisitos de seguridad implementables mediante las funciones de seguridad existentes.

Para utilizar estos recursos siempre hay que compilar con la biblioteca de seguridad, -lprot. Para las funciones de contraseña además debe compilarse con la biblioteca matemática -lm, mientras que para la auditoria debe compilarse con -laudit.

Puede utilizarse el programa ccprot para ayudar a realizar la compilación.

```
# Fichero :  ccprot
#
#Compila con la biblioteca de funciones seguras, monta y ejecuta
#
#Las funciones de biblioteca para manejar contraseñas tienen que
#compilarse con la biblioteca matemática: - lm
#
#Las funciones de biblioteca de auditoria tienen que compilarse con
#la biblioteca de auditoría : - laudit

for     Programa_Fuente                 in $*
do
        ERRORES=" ";
        Nombre_Sin_Extension=`basename $Program_fuente   .c`;
#       echo $programa_Fuente $Nombre_Sin_Extension   ;
        cc $Programa_Fuente     - o $Nombre_Sin_Extension   - lprot    \
                        2> $Nombre_Sin_Extension.err ;
```

```
        Errores=`grep error $Nombre_Sin_Extension.err`;
        echo "$Errores" ;
if [-n "$Errores "]
        then
                pg $Nombre_Sin_Extension.err ;
        else
                rm $Nombre_Sin_Extension.err ;
                > $Nombre_Sin_Extension.res 2> /dev/null ;

                $Nombre_Sin_Extension >> $Nombre_sin_Extension.res \
                                                2>> $Nombre_Sin_Extension.res ;
                pg $Nombre_Sin_Extension.res ;
        fi
done
```

10.1 Archivos de cabecera

Los archivos de cabecera del sistema seguro contienen definiciones y estructuras de datos necesarias para sus características de seguridad. Los programas de usuario no suelen acceder a estos ficheros, que son:

- <sys/security.h>. Contiene definiciones que utilizan las llamadas de seguridad al sistema operativo.
- <prot.h> Posee definiciones correspondientes a la base de datos de autenticación.
- <sys/audit.h> Contiene las definiciones del registro de auditoría, así como otras estructuras de datos y definiciones asociadas con el subsistema de auditoría.

Todos los programas que utilicen las funciones de seguridad tienen que definir una constante simbólica llamada SecureWare para acceder a ciertas definiciones imprescindibles para su funcionamiento:

```
# defineSecureWare
```

10.2 Funciones de biblioteca.

La mayor parte de la interfaz de programación de un sistema seguro se realiza mediante funciones de biblioteca : esto flexibiliza su construcción, al estar fuera del núcleo del sistema operativo frente a las llamadas al sistema que forman parte del mismo.

Las funciones de biblioteca de un sistema seguro se agrupan en las siguientes categorías:

- Gestión de contraseñas robustas.
- Auditoría.

10.2.1Contraseñas

La misión de las funciones correspondientes es garantizar la robustez de la contraseña . Para utilizar estas funciones es necesario compilar con la biblioteca matemática, dando la opción - lm, y llamar previamente a las funciones set_auth_parameters().

Pueden agruparse en las siguientes categorías:

- Cálculo de la longitud mínima de la contraseña.
- generación aleatoria de contraseña pronunciables.
- Aplicación de pruebas para determinar la validez de una contraseña propuesta por un usuario .

Longitud mínima de la contraseña.

Según el tiempo de vida permitido a la contraseña, el número de caracteres del alfabeto con el que se construye, y el tiempo estimado entre intentos consecutivos de conexión, se puede valuar la longitud mínima que debería tener la contraseña para minimizar la probabilidad de éxito de un ataque que intente conjeturar la misma.

Sintaxis de la función para calcular la longitud mínima de la *contraseña*

```
int     passlen(Vida, Intervalo_Conexión, Tamaño_alfabeto)
        time_t      Vida;
        time_t      Intervalo_Conexión;
        int         Tamaño_Alfabeto;
```

Descripción de la función para calcular la longitud mínima de la contraseña.

La función PASSLEN() devuelve la longitud mínima de la contraseña calculada mediante el algoritmo . Para ello utiliza el tiempo de vida de la contraseña (Vida), el intervalo entre dos intentos de conexión (Intervalo_Conexión) y el número de caracteres distintos del alfabeto (Tamaño del alfabeto).

Generación aleatoria de contraseñas pronunciables.

Si el administrador de seguridad del sistema no permite que los usuarios elijan su contraseña, debe proporcionarles una fácil de memorizar, lo que usualmente se consigue si la misma es pronunciable, es decir, sus caracteres alfabéticos constituyen una cadena de vocales y consonantes que puede articularse.

Sintaxis de la función para generar contraseñas pronunciables.

```
Int     randomword( Palabra, Palabra_Silabizada,
                    Minimo, Maximo, Restriccion,
                    Semilla )
            char     *Palabra, *palabra_Silabizada ;
            usigned short int     Minimo, Maximo;
            int Restriccion ;     long  Semilla;
```

Descripción de la función para generar contraseñas pronunciables

La función randomword() devuelve una contraseña en la cadena palabra, y su versión silabizada en Palabra_Silabizada. Minimo y Maximo indican las cortas respectivas en la longitud de las contraseñas. No hay restricciones en la contraseña cuando Restriccion vale cero, de lo contrario, se aplican las pruebas de robustez de la función acceptable_password(). Semilla es un valor inicial que utiliza el generador aleatorio de la contraseña.

Aceptabilidad de la contraseña

Sintaxis de la función para valorar la aceptabilidad de una contraseña.

```
Int             acceptable_passwdord ( Palabra, Fichero )
                        char    *Palabra ;
                        FILE    *Fichero;
```

Descripción de la función para valorar la aceptabilidad de una contraseña.

Esta función determina si la contraseña que hay en Palabra es lo suficientemente robusta como para soportar el ataque de un programa averiguador de contraseñas. En el segundo argumento se especifica un Fichero en el que se almacenará el informe de validez. Si la función devuelve 1, la contraseña ha superado todas las pruebas: palíndromo , nombre de conexión, nombre de grupo, palabra de diccionario, etc.

Esta función tan sólo puede aplicarse si el programa principal se ha llamado previamente a la función set_auth_parameters().

Programa para comprobar las funciones de contraseña.

```
/*
*       Archivo: passwdc2.c
*               cc paswdc2.c - o passwdc2 - lprot - lm
*
```

```
 * Manipulación de contraseñas
 *
 */

#define     SecureWare /* Necesario para acceder a ciertos valores de UNIX C2 */
#include    <stdio.h>
#include    <sys/types.h>
#include    <sys/security.h>
#include    <sys/audit.h>
#include    <prot.h>

#include    <errno.h>

#define     ERROR                           (-1)
#define     INACEPTABLE                 0
#define     CERO                        0
#define     SIETE_CARACTERES            7
#define     OCHO_CARACTERES             8
#define     VEINTE_CARACTERES                   20
#define     SESENTA_CARACTERES          60
#define     SIN_RESTRICCIONES           0
#define     CON_RESTRICCIONES                   1
#define     TAMAÑO_MAXIMO_DE_LA_CONTRASEÑA  100
#define     VEINTE_DIAS                 20*40*60*60
#define     DOS_DIAS                    2*24*60*60
#define     NUMERO_DE_LETRAS_DISTINTAS      26*02

main (argc, argv )
        int     argc;
        char    *argv[ ];
{
        char    contraseña[ TAMAÑO_MAXIMO_DE_LA_CONTRASEÑA ];
```

```
char    contraseña_Silabizada[
                        TAMAÑO_MAXIMO_DE_LA_CONTRASEÑA];
int     Longitud_Minima_De_La_Contraseña = SIETE_CARACTERES;
int     Longitud_Maxima_De_La_Contraseña = OCHO_CARACTERES;
int     Tiempo_De_Vida_De_La_Contraseña = VEINTE_DIAS;
int     Restricciones_Sobre_La_Contraseña =SIN_RESTRICCIONES;
long    Semilla_Para_Generar_La_Contraseña = CERO;
int     Intervalo_Entre_Conexiones = DOS DIAS;
int     Tamaño_Del_Alfabeto = NUMERO_DE_LETRAS_DISTINTAS;
FILE    *Archivo_De_Mensajes_De-Error = NULL ;


set_auth_parameters( argc, argv );

Longitud_Minima_De_La_Contraseña = passlen (
                        Tiempo_De_Vida_De_La_Contraseña,
                        Intervalo_Entre_Conexiones,
                        Tamaño_Del_Alfadbeto );

printf( " La longitud mínima de la contraseña es:  %d \n",Longitud_Minima_De_La_Contraseña );
Longitud_Minima_De_La_Contraseña = OCHO_CARACTERES;
Longitud_Maxima_De_La_Contraseña = VEINTE_CARACTRES;
Restricciones_Sobre_La_Contraseña = SIN_RESTRICCIONES;
time( &Semilla_Para_Generar_La_Contraseña );

randomword( Contraseña, Contraseña_Silabizada, Longitud_Minima_De_La_Contraseña,
                Longitud_Maxima_De_La_Contraseña, Restriccione_Sobre_La_Contraseña,
                Semilla_Para_Generar_La_Contraseña );

printf( "Contraseña: %s\n", Contraseña);
printf( " Contraseña_Silabizada :%s \n", Contraseña_Silabizada);
```

```
Longitud_Minima_De_La_Contraseña = VEINTE_CARACTERES;
Longitud_Maxima_De_La_Contraseña = SESENTA_CARACTERES;
Restricciones_Sobre_La_Contraseña = CON_RESTRICCIONES;
time(&Semilla_Para_Generar_La_Contraseña );

randomword ( Contraseña, Contraseña_Silabizada, Longitud_Minima_De_La_Contraseña,
             Longitud_Maxima_De_La_Contraseña, Restricciones_Sobre_La_Contraseña
             Semilla_Para_Generar_La_Contraseña );

printf( " Contraseña: \n %s\n ",Contraseña );
printf( "Contraseña silabizada: \n %s \n", Contraseña_Silabizada );

system( "> fmerr" ) ;
Archivos_De_Mensajes_De_Error = fopen ("fmerr", "a+" );

if ( acceptable_password ( "aaa", Archivo_De_Mensajes_De_Error ) == INACEPTABLE )
{
        printf( "Ha detectado un polindromo \n ");
}

if ( acceptable_password( "root8", Archivo_De_Mensajes_De_Error )== INACEPTABLE )
{
        printf( " Ha detectado un nombre de conexión \n");
}

if ( acceptable_password( "graup", Archivo_De_Mensajes_De_Error ) == INACEPTABLE )
{
        printf( " Ha detectado un nombre de grupo \n ) ;
}
if ( acceptable_password("hellos", Archivo_De_Mensajes_De_Error ) == INACEPTABLE)
{
printf( "Ha detectado una palabra de diccionario \n " );
```

```
        }

        return( 0 );
}
```

Ejecución del programa sobre contraseñas

```
$ passwdc2
La longitud mínima de la contraseña es : 5
Contraseña: asbeccitnomvoolixeu
Contraseña silabizada: as-bec-cit-nom-vool-ix-eu
Contraseña:
        vowdiatibsatackabvukgowgesjun
Contraseña silabizada:
        vowd-iat-ibs-at-ack-ab-vuk-gowg-es-jun
Ha detectado un palindromo
Ha detectado una palabra de diccionario
$
```

10.3 Recomendaciones para una programación segura

A continuación se proporcionan algunas recomendaciones útiles para que los programadores construyan aplicaciones seguras:

- Consultar al administrador de seguridad antes de ejecutar programas que escriban en archivos sensibles del sistema, necesiten privilegios de superusuarios o manipulen estructuras de datos del núcleo del sistema operativo.

- Establecer un entorno para los programas y el manejo de señales. Esto es fundamental para programas que ejecutan otros programas utilizando las llamadas exec(), system() o popen(). Los programas con el atributo SUID deberían tener ajustadas las variables IFS y PATH.
- Restringir umask() para los programas que crean archivos privados.
- Utilizar caminos completos para ejecutar programas.
- Proteger a los programas de lectura por parte de cualquier usuario.
- los programas con el atributo SUID que ejecutan otros programas pueden penetrarse fácilmente. Cuando se pase el control a un subprograma utilizar las llamadas setuid () y setgid () para restaurar los permisos originales. Asegurarse que los archivos necesarios tan sólo en el proceso padre se cierran antes de llamar al proceso hijo.
- Observar que las llamadas al sistema access() y eaccess() son diferentes. Usar la apropiada.
- Capturar y manejar todas las señales que puedan recibirse. Es importante deshacer cualquier acción parcial que pueda dejar archivos sensibles en un estado inconsistente.

CONCLUSIONES

Muchos de los problemas de seguridad que ocurren pasan desapercibidos hasta que generan un problema grave, debido a que nadie se preocupa por estar al pendiente de lo que sucede en el sistema.

La seguridad en cómputo es un tema difícil, sobre todo para la mayoría de la gente que lo ve como un hecho lejano y no como un problema real al que es necesario enfrentarse todos los días.

A pesar de todo, la seguridad no es tan lejena como parece; aplicando las técnicas y herramientas con un criterio bien definido, pueden permitir a un sistema Unix contar con un nivel de seguridad apropiado para las necesidades de sus usuarios y sus administradores.

Es importante recordar que las herramientas nunca podrán reemplazar a un buen administrador, consciente de sus tareas e informado de los problemas a los que puede llegar a enfrentarse y sus posibles soluciones.

BIBLIOGRAFÍA

1. Sánchez Sebastián. " Unix guía del usuario ". Editorial RA-MA 1996.

2. Todino Grace, John Strang and Jerry Peek. " Unix Práctico ". Editorial Prentice Hall. 1995

3. Ribagorda Garnacho A., Calvo Orra A. " Seguridad en Unix ". Editorial Paraninfo. 1996.

4. Christian Kaare . " Unix " Editorial Prentice Hall. 1996

5. Calvo Orra Alfonso. " Auditorías software ", Editorial Grupo OPL. 1993.

6. Lechner, Mikel. " Primeros pasos en el mejoramiento de la seguridad en Unix ". EDPACS. 1992.

7. Morant J.L.. " Seguridad y protección de la Información ". Editorial CEURA. 1994.