TECNOLÓGICO UNIVERSITARIO DE MÉXICO

ESCUELA DE INFORMÁTICA

INCORPORADA A LA
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CLAVE 3079-48

"LOS PUNTOS DE SEGURIDAD MAS IMPORTANTES EN UN SISTEMA LOCAL EN UNIX"

T E S I S

QUE PARA OBTENER EL TITULO DE:
LICENCIADA EN INFORMÁTICA
P R E S E N T A:

MAYRA HERNÁNDEZ MARTÍNEZ

ASESOR DE TESIS:
L.I. JOSÉ FRANCISCO ÁGUILA PATIÑO

MÉXICO, DF. **2007**

AGRADECIMIENTOS

Agradezco a Dios principalmente por permitirme compartir con ustedes este momento, a mis padres, por darme fuerza de salir a delante, por su apoyo incondicional, que me han ayudado a salir adelante, por su cariño y cuidados de padres, que no han disminuido desde el momento de mi nacimiento, hasta el día de hoy, por que con su amor y ejemplo me ha enseñado mas de la vida que todas las escuelas juntas, por que es la roca en la cual me apoyo, muchas gracias mamá por quererme tanto y querer siempre lo mejor para mi, gracias por tu entrega y dedicación que me das día a día.

A mi director de Tesis: L.I. José Francisco Águila Patiño

Por compartir su inestimable experiencia conmigo, por su apoyo, por su paciencia con mis errores, por su guía y sus valiosos consejos, que me ayudaron a salir adelante en esta etapa tan importante de mi vida.

A mis familiares y amigos:

A mis familiares. Por que siempre creyeron en mí y siempre tuvieron una palabra de aliento hacia mi persona, por que siempre he contado con ellos, en las buenas y en las malas.

A mis amigos. Por todos los momentos que vivimos juntos, por que llenaron de felicidad una época muy importante de mi vida, por que compartieron conmigo los sueños del futuro y las alegrías del presente.

“LOS PUNTOS DE SEGURIDAD MAS IMPORTANTES EN UN SISTEMA LOCAL EN UNIX”

INTRODUCCIÓN.

No se puede dejar pasar por alto la seguridad en general y principalmente por las cosas de suma importancia, es por eso que me interesa dar a conocer un poco mas de la seguridad informática de los múltiples riesgos a los que a diario nos enfrentamos a nuevas cosas; entre ellos mi interés por ver la seguridad en el sistema operativo UNIX en un sistema se que este tema es muy extenso pero al menos podemos tocar los puntos mas importante de los que conlleva la seguridad en UNIX. Se preguntaran por que se da mi decisión repentina en realizar y poner mi empeño en esta tesis, todo esto surge durante mi servicio social en PEMEX Refinación donde el sistema operativo a utilizar es UNIX manejándolo y viendo muchos casos de seguridad entre las cuales están las cuentas de usuarios etc... Mi interés por ver como se maneja todo esto comenzó cada día a crecer, tratando de dar una respuesta a todas aquellas dudas en cuanto a seguridad y aquellos puntos débiles por los que este sistema operativo se ve vulnerable y muchos al igual que yo hasta hoy desconocíamos.

Es una realidad que día a día nos enfrentamos a nuevos ataques con los que a su vez nuestra seguridad se ve interrumpida. Es por que ello que debemos tomar medidas de seguridad, para evitar encontrarnos con tipos de problemas, con los famosos intrusos que pueden dañar, modificar, o incluso eliminar todo lo que tenemos, que pueden entrar en nuestro sistemas y realizar gran numero de operaciones por nosotros, que después nos puedan costar, tanto tiempo, dinero y esfuerzo. Es fundamental volver a recalcar, lo importante que es la seguridad y poco a poco ir conociendo diversas opciones que nos pueden ayudar a mejorarla.

Para ello espero que en este trabajo se vean reflejados algunos de los puntos mas importante para poder tener una mejor seguridad, desde como dejamos la máquina hasta en la manera en que debemos buscar una buena contraseña, los permisos que se les asigna a determinados usuarios para poder manipular y hacer uso de información, teniendo además un adecuado medio de seguridad para evitar que hackers y virus puedan entran y nos puedan afectar, y junto con ello evitar un gran caos.

1 SISTEMAS OPERATIVOS

1.1 ANTECEDENTES

Concepto de un Sistema Operativo.

-Un Sistema Operativo es un conjunto de programas que ordenadamente y relacionándose entre sí, contribuyen a que el ordenador lleve a efecto correctamente el trabajo encomendado **(1)**.

-Un Sistema Operativo es una parte importante de cualquier sistema de computación. Un sistema de computación puede dividirse en cuatro componentes: el hardware, el Sistema Operativo, los programas de aplicación y los usuarios. El hardware (Unidad Central de Procesamiento (UCP), memoria y dispositivos de entrada / salida (E/S) proporciona los recursos de computación básicos. Los programas de aplicación (compiladores, sistemas de bases de datos, juegos de video y programas para negocios) definen la forma en que estos recursos se emplean para resolver los problemas de computación de los usuarios **(2)**.

-Desde el punto de vista del usuario, el sistema operativo es una serie de programas y funciones que ocultan los detalles del hardware ofreciéndole una vía sencilla y flexible de trabajar con él.

-Un Sistema Operativo es un conjunto de programas que sirven como ***interfaz*** entre el usuario (Sirve como agente de intercambio de información entre la computadora y el usuario) y la computadora, además de que ***administran*** los recursos de la misma (entendiéndose como recursos: el hardware y el software, memoria, disco duro, procesador, monitor, etc.). En mi definición un Sistema Operativo es un conjunto de programas que controla y administra los recursos de la computadora.

Los sistemas operativos han venido evolucionando a través de los años, ya que los sistemas operativos se han apegado íntimamente a la arquitectura de las computadoras en las cuales se ejecutan. A continuación un breve resumen de sus generaciones.

(1) Libro teoría y diseño de los sistemas operativos, Juan M. Morena Pascual, Juan A. Pérez Campanero, Edit. Anaya Multimedia pagina 24 **.**

(2) De Internet SISTEMA OPERATIVO en el buscador google.com.mx.

Sus generaciones:

0º Generación: De acuerdo a lo investigado en la década de los 40 los sistemas informáticos no disponían de Sistema Operativo con lo que los usuarios de estos debían introducir las instrucciones en código binario lo que hacia su uso restringido a personas de mucho conocimiento en esa materia.

1º Generación: Por lo que en la década de los 50 aparece el primer Sistema Operativo para lograr la fluidez en la trasmisión de información, el primero que aparece es el **JCL (**lenguaje de control de trabajo**),** se usaban tarjetas perforadas y eran controladas por operadores (personas con cierto conocimiento). Posteriormente se pasaron de las tarjetas a las cintas perforadas y estas avanzaban mucho mas rápido.

El primer Sistema Operativo ocupaba en memoria 64 KB (bastante en función de la capacidad total de. la memoria en aquella época).ver figura 1.1

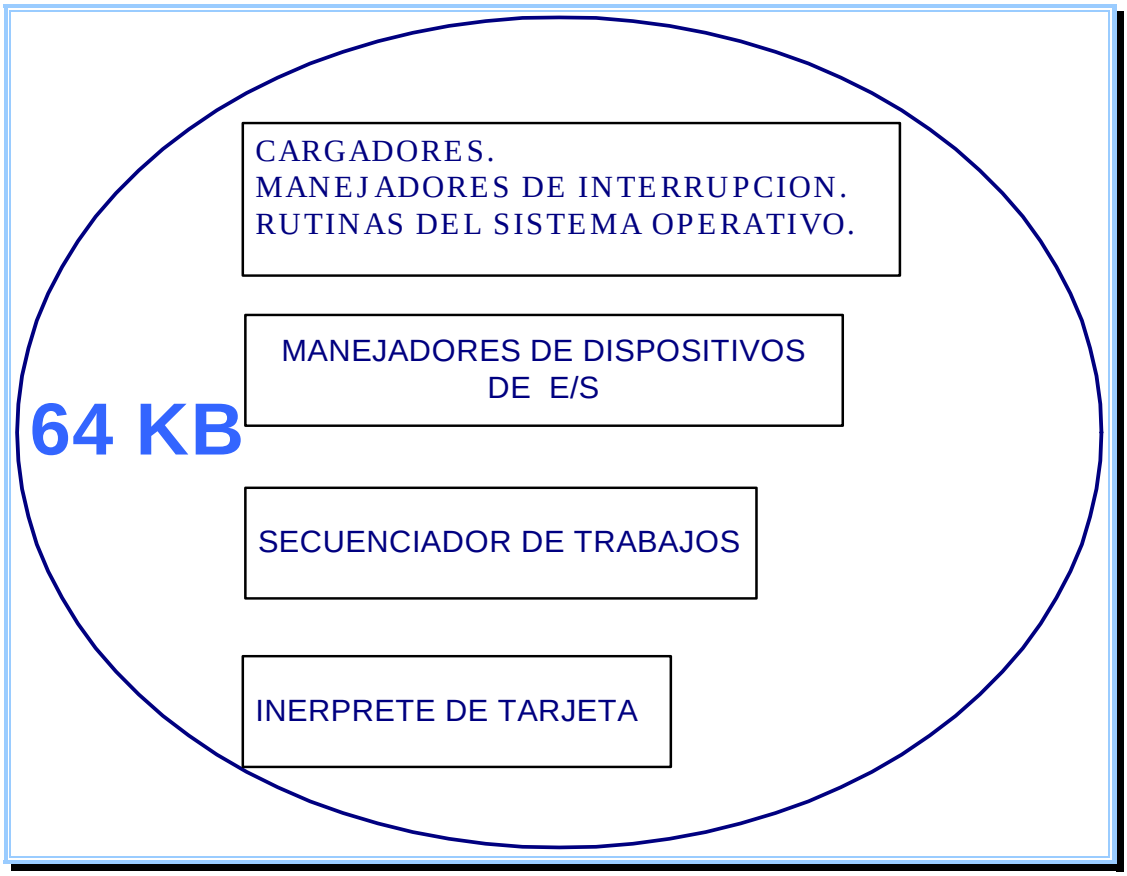


Figura 1.1 Grafico interno de un sistema operativo

2º Generación: Se dan los primeros pasos de la multiprogramación es decir varios programas de usuario. El Sistemas Operativos reparte tiempos del procesador. Aparece nueva tecnología que usa buffers entre terminales: impresora, etc.

3º Generación: Segunda mitad de la década de los 60 y 1º mitad de los 70. Es entonces cuando se desarrollan los Sistemas Operativos tan importantes como el UNIX para la gestión de grandes mainframes.

Durante esta generación el usuario perdió el control del hardware. En los equipos informáticos venían incluido el Sistemas Operativos.

4º Generación: En la segunda mitad de los 70 y primera de los 80. Los Sistemas Operativos aumentan sus prestaciones y gestionan eficientemente los recursos del ordenador. Es en esta época donde más facilidad se le da al usuario para su manejo. IBM separa los costos de hardware y software, con esta estrategia de marketing se pensaba que facturarían el doble en ganancias. Pero no fue así. Los vendedores de software pasan a hacerse responsables de los bugs (o fallos de sus programas). Se incrementaron las empresas desarrolladoras de software por lo que esto perjudico seriamente a IBM ya que así perdió la exclusividad.

Posteriormente se abrió el mercado de computadoras compatibles con IBM, estos son los llamados clónicos donde varios dispositivos informáticos de distintas fabricas o procedencias intercomunicados y compatibles entre si constituyendo un único equipo informático. Este tipo de ordenadores es de precio mucho mas reducido que los IBM y sus prestaciones son las mismas.

5º Generación: Llegando a los 90 los entornos gráficos cobraron mucha importancia, evolucionaron las llamadas GUI (interfaces graficas del usuario). Los sistemas operativos tipo windows 9*, millenium o NT para empresas desplazaron de las empresas al ya consolidado UNÍX.

Figura 1.2. Grafico interno de la fusión entre el software y el hardware.

Entre una de las características e innovaciones mas importantes de un Sistema Operativo es la llamada **multi-tarea**. Un Sistema Operativo opera entre la CPU y los periféricos y con la tecnología multi-tarea reparte el tiempo entre ellos dos.

Concepto de Proceso

No podría entenderse como trabaja un Sistema Operativo sin la entidad básica llamada proceso. Un proceso es un programa en ejecución, junto con su entorno asociado (registros, variable, datos, etc).

El núcleo de Kernel, es la parte central del sistema operativo, y consiste en un conjunto de rutinas cuya misión es la de controlar el procesador, la memoria, la entrada y salida y el resto de recursos de la computadora. El Kernel reacciona ante cualquier interrupción de eventos y da

atención a los procesos, creándolos, terminándolos y respondiendo a cualquier petición de servicio por parte de los mismos.

La diferencia entre un programa (conjunto de instrucciones) y un proceso (instrucciones ejecutándose), es importante para entender los estados de un proceso.

Los estados de un proceso son la etapas o condiciones por las que pasa un programa desde que fue colocado por el usuario en la computadora para su ejecución, hasta que termina.

En la figura 1.3 siguiente, puede verse gráficamente la transición entre los estado de un proceso.

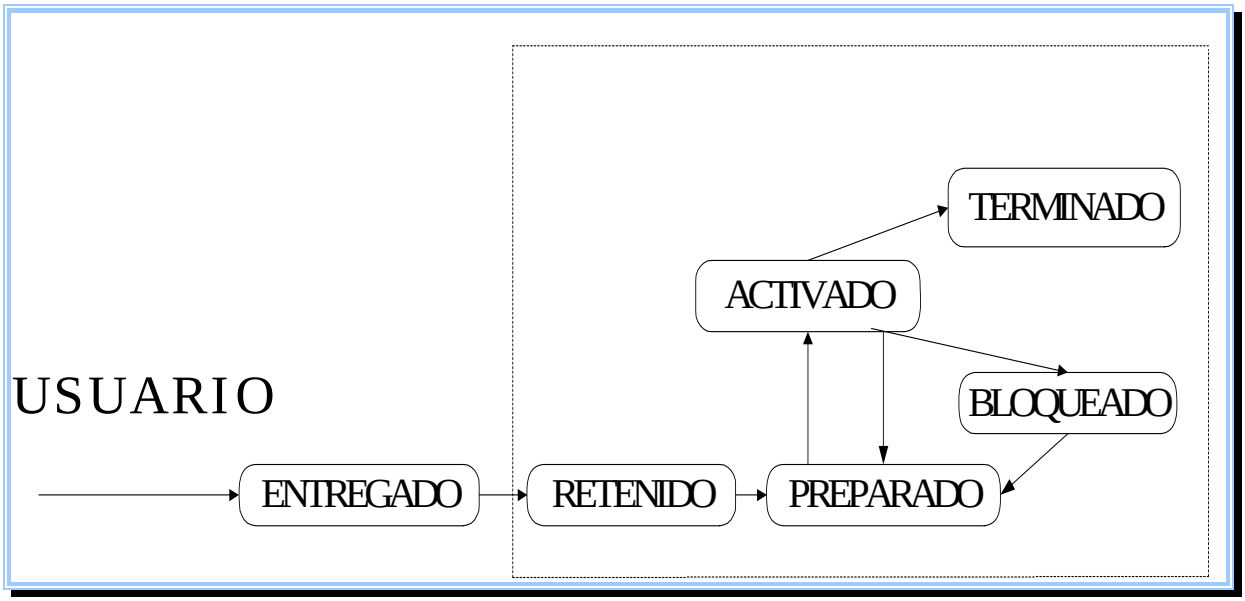


Figura 1.3 Estado de Proceso.

1. Entregado: El usuario entrega un trabajo al Sistema Operativo y éste lo coloca en la cola de trabajos en espera.
2. Retenido: El proceso está en la cola de trabajos en espera. El programa existe pero todavía no es conocido por el sistema operativo.
3. Preparado: El calendarizador de trabajos (Job Scheduler) del Sistema Operativo pasa al programa a la cola de trabajos preparados o listos en espera de ser ejecutado. Esto dependerá de las políticas de encolamiento como la prioridad del trabajo, su hora de

ejecución, etc. El proceso es cargado dentro de la memoria y listo para correr, en espera de tiempo del procesador.

4. Activo: El proceso empieza a correr tomando tiempo del procesador. El planificador de procesador del Sistema Operativo le asigna tiempo del procesador. Si se termina su tiempo el proceso regresa al estado preparado.
5. Bloqueado: El proceso está en espera de terminar una operación de entrada o salida, o de que se cumpla una condición.
6. Terminado: El proceso ha completado su trabajo y sale del sistema.

1.2 FUNCIONES

Las funciones de los sistemas operativos son diversas y han ido evolucionando con el paso del tiempo de acuerdo con los progresos de la técnica y la informática han experimentado. Las principales funciones de un sistema Operativo serían:

Administración de Trabajos. Cuando existen varios programas en espera de ser procesados, el sistema operativo debe decidir el orden de procesamiento de ellos, así como asignar los recursos necesarios para su proceso.

Administración de Recursos. Consiste que el sistema operativo esta en capacidad de distribuir en forma adecuada y en el momento oportuno los diferentes recursos (memoria, dispositivos, etc.,...) entre los diversos programas que se encuentran en proceso, para esto, lleva un registro que le permite conocer que recursos están disponibles y cuales están siendo utilizados, por cuanto tiempo y por quien, etc.

Control de Operaciones de Entrada y Salida. El sistema operativo decide que proceso hará uso del recurso, durante cuánto tiempo y en que momento.

Administración de la Memoria. Supervisa que áreas de memoria están en uso y cuales están libres, determina cuanta memoria asignará a un proceso y en que momento, además libera la memoria cuando ya no es requerida para el proceso.

Recuperación de Errores. El sistema operativo contiene rutinas que intentan evitar perder el control de una tarea cuando se producen errores en la trasferencia de información hacia y desde los dispositivos de entrada / salida.

Programas de Servicio. El sistema operativo contiene programas de servicios que apoyan al procesamiento de los trabajos, se conocen también como utilerías y se pueden clasificar en tres tipos:

A) Utilerías del Sistema. Se ejecutan bajo el control del sistema operativo y se utilizan para preparar algunos recursos usados por el sistema, que son de uso interno.

B) Utilerías para Archivos. Manejan información de los archivos tales como imprimir, clasificar, copiar, etc.

C)Utilerías Independientes. Realizar funciones que se relacionan con la iniciación de dispositivos de Entrada / salida, carga del sistema operativo, etc.

D) Seguridad. Protege las memorias, los archivos.

Haciendo breve resumen sus funciones mas importantes son:

- Gestiona los recursos de la computadora en sus niveles mas bajos.
- Dispone de una interface (elemento que hace posible la fácil comunicación usuario maquina) liberando al usuario del conocimiento del hardware. El Sistema Operativo windows se basa en una interface graficas, "GUI" (Interface Grafica de Usuario), permitiendo al usuario interactuar con el hardware de una forma sencilla y rápida.
- Sobre el Sistema Operativo funcionan el resto de programas y aplicaciones del software.

1.3 TIPOS DE SISTEMAS OPERATIVOS.

Actualmente los sistemas operativos se clasifican en tres grandes grupos: sistemas operativos por sus servicios (visión externa), sistemas operativos por su estructura (visión interna), sistemas operativos de red.

Un ejemplo si abordamos su estudio desde la perspectiva del número de usuarios a los que pueden dar servicio, o la forma de utilizar los recurso del sistema, dicha clasificación puede variar, según la utilización de recursos, esta se basa en las características con que se hayan diseñado dichos sistemas para hacer uso de los recursos existentes.

Sistemas operativos por visión externa.

Esta clasificación es la más comúnmente usada y conocida desde el punto de vista del usuario final.

Esta clasificación se comprende fácilmente con el cuadro sinóptico que a continuación se muestra en la figura 1.4.

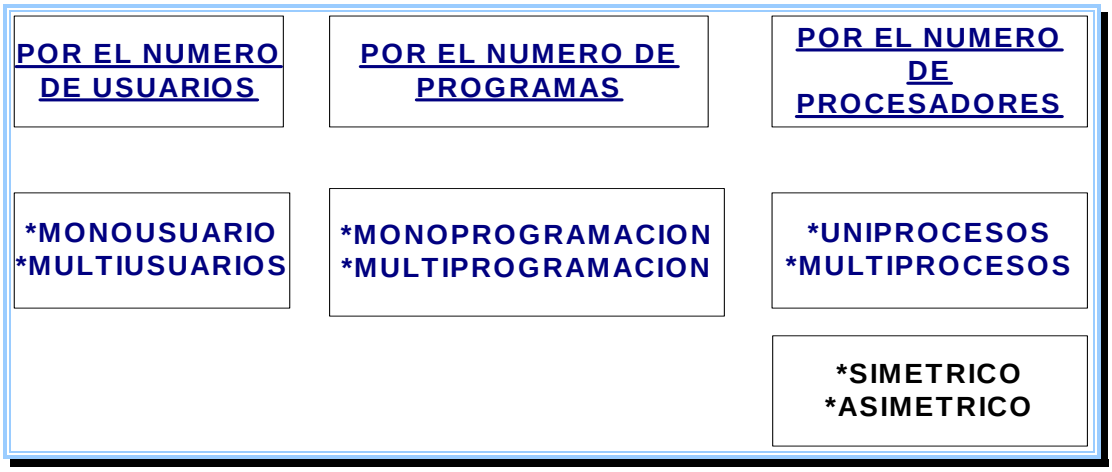


Figura 1.4 Sistemas Operativos por servicios.

Por el número de usuarios:

Sistema Operativo Monousuario. Son aquellos que soportan un usuario a la vez, sin importar el número de procesadores que tenga la computadora o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo.

En otras palabras los sistemas monousuarios son aquellos que nada más puede atender a un solo usuario, gracias a las limitaciones creadas por el hardware, los programas o el tipo de aplicación que se este ejecutando.

Sistema Operativo Multiusuario. Son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.

En esta categoría se encuentran todos los sistemas que cumplen simultáneamente las necesidades de dos o más usuarios, que comparten mismos recursos.

Este tipo de sistemas se emplean especialmente en redes. En otras palabras consiste en el fraccionamiento del tiempo (timesharing).

Por el número de programas:

Sistema Operativo Monoprogramación. Son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monoprogramación, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo solo una tarea a la vez.

Los sistemas operativos monoprogramación son más primitivos y, solo pueden manejar un proceso en cada momento o que solo puede ejecutar las tareas de una en una.

Sistema Operativo Multiprogramación. Un sistema operativo multiprograma es aquél que le permite al usuario estar realizando varios programas al mismo tiempo.

Es el modo de funcionamiento disponible en algunos sistemas operativos, mediante el cual una computadora procesa varias tareas al mismo tiempo. Existen varios tipos de multiprogramas. La conmutación de contextos (context Switching) es un tipo muy simple de multiprograma en el que dos o más aplicaciones se cargan al mismo tiempo, pero en el que solo se esta procesando la aplicación que se encuentra en primer plano (la que ve el usuario). En la multiprogramación cooperativa, la que se utiliza en el sistema operativo Macintosh, las tareas en segundo plano reciben tiempo de procesado durante los tiempos muertos de la tarea que se encuentra en primer plano (por ejemplo, cuando esta aplicación esta esperando información del usuario), y siempre que esta aplicación lo permita. En los sistemas multiprogramas de tiempo compartido, cada tarea recibe la atención del microprocesador durante una fracción de segundo.

Un sistema operativo multitarea puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background. Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.

Un sistema operativo multiprogramas se distingue por su capacidad para soportar la ejecución concurrente de dos o más procesos activos. La multiprogramación se implementa generalmente manteniendo el código y los datos de varios procesos simultáneamente en memoria y multiplexando el procesador y los dispositivos de entrada y salida entre ellos.

La multiprogramación suele asociarse con soporte hardware y software para protección de memoria con el fin de evitar que procesos corrompan el espacio de direcciones y el comportamiento de otros procesos residentes.

Por el número de procesadores

Sistema Operativo de Uniproceso. Un sistema operativo uniproceso es aquél que es capaz de manejar solamente un procesador de la computadora, de manera que si la computadora tuviese más de uno le sería inútil.

Sistema Operativo de Multiproceso. Un sistema operativo multiproceso se refiere al número de procesadores del sistema, que es más de uno y éste es capaz de usarlos todos para distribuir su carga de trabajo. Generalmente estos sistemas trabajan de dos formas: simétrica o asimétricamente.

- **Asimétrica:** Cuando se trabaja de manera asimétrica, el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que reciben el nombre de esclavos.
- **Simétrica:** Cuando se trabaja de manera simétrica, los procesos o partes de ellos (threads) son enviados indistintamente a cual quiera de los procesadores disponibles, teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema. Se dice que un thread es la parte activa en memoria y corriendo de un proceso, lo cual puede consistir de un área de memoria, un conjunto de registros con valores específicos, la pila y otros valores de contexto. Un aspecto importante a considerar en estos sistemas es la forma de crear aplicaciones para aprovechar los varios procesadores. Existen aplicaciones que fueron hechas para correr en sistemas monoproceso que no toman ninguna ventaja a menos que el sistema operativo o el compilador detecte secciones de código paralelo, los cuales son ejecutados al mismo tiempo en procesadores diferentes.

Por otro lado, el programador puede modificar sus algoritmos y aprovechar por sí mismo esta facilidad, pero esta última opción suele ser costosa y difícil, obligando al programador a ocupar tanto o más tiempo a la paralelización que va elaborar el algoritmo inicial.

Sistemas operativos por su estructura (visión interna).

Según, se deben observar dos tipos de requisitos cuando se construye un sistema operativo, los cuales son:

- Requisitos **de usuario**: Sistema fácil de usar y de aprender, seguro, rápido y adecuado al uso al que se le quiere destinar.
- Requisitos **del software**: Donde se engloban aspectos como el mantenimiento, forma de operación, restricciones de uso, eficiencia, tolerancia frente a los errores y flexibilidad. A continuación se describen las distintas estructuras que presentan los actuales sistemas operativos para satisfacer las necesidades que de ellos se quieren obtener.

Estructura Monolítica. Es la estructura de los primeros sistemas operativos constituidos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra. Las características fundamentales de este tipo de estructura son:

- Construcción del programa final a base de módulos compilados separadamente que se unen a través del ligador (linker).
- Buena definición de parámetros de enlace entre las distintas rutinas existentes, que puede provocar mucho acoplamiento.
- Carecen de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos de la computadora, como memoria, disco, etc.

Generalmente están hechos a la medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

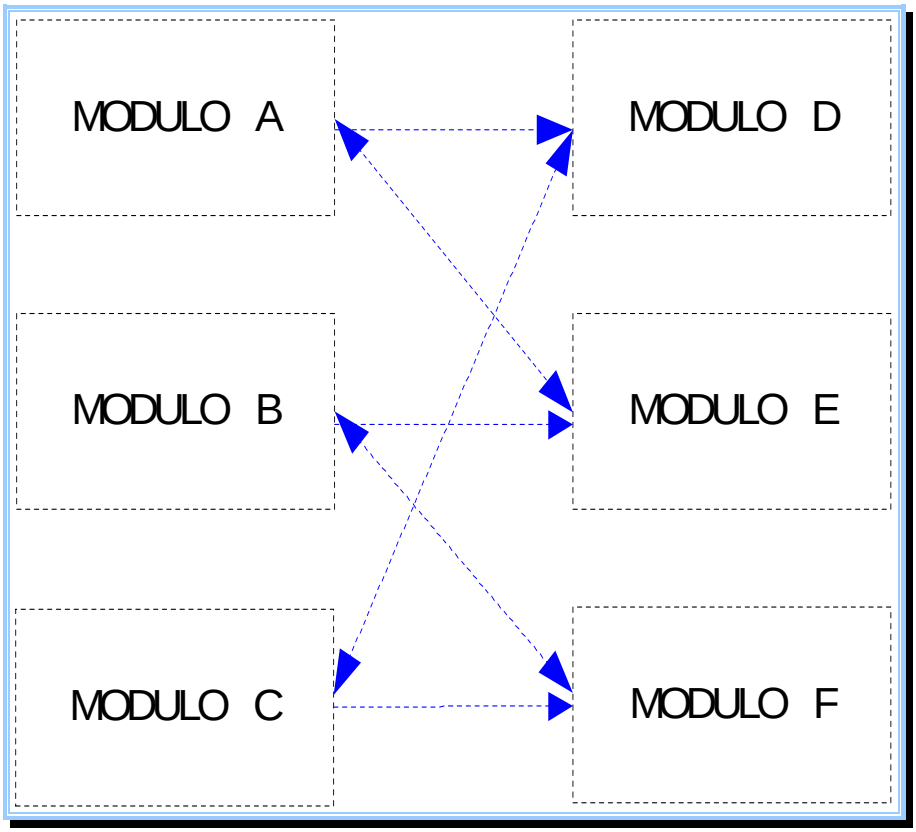


Figura 1.5 Estructura monolítica **(3)**.

Estructura Jerárquica. Con el paso del tiempo y las necesidades y las demandas de los usuarios, se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software, del sistema operativo, donde una parte del sistema contenía subpartes y esto organizado en forma de niveles. Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de elementos. Se constituyó una estructura jerárquica o de niveles en los sistemas operativos, el primero fue denominado **THE (Technische Hogeschool, Eindhoven),** de Dijkstra, que se utilizó con fines didácticos.

(3) Figura 1.5: consulta teoría y diseño de los Sistemas Operativos, de Juan M. Morena Pascual, Juan a. Pérez Campanero, Edit. Anaya Multimedia pagina 73

Figura 1.6 Sistema Jerárquico THE .

En la estructura anterior se basan prácticamente la mayoría de los sistemas operativos actuales. Otra forma de ver este tipo de sistema es la denominada de anillos concéntricos "rings".

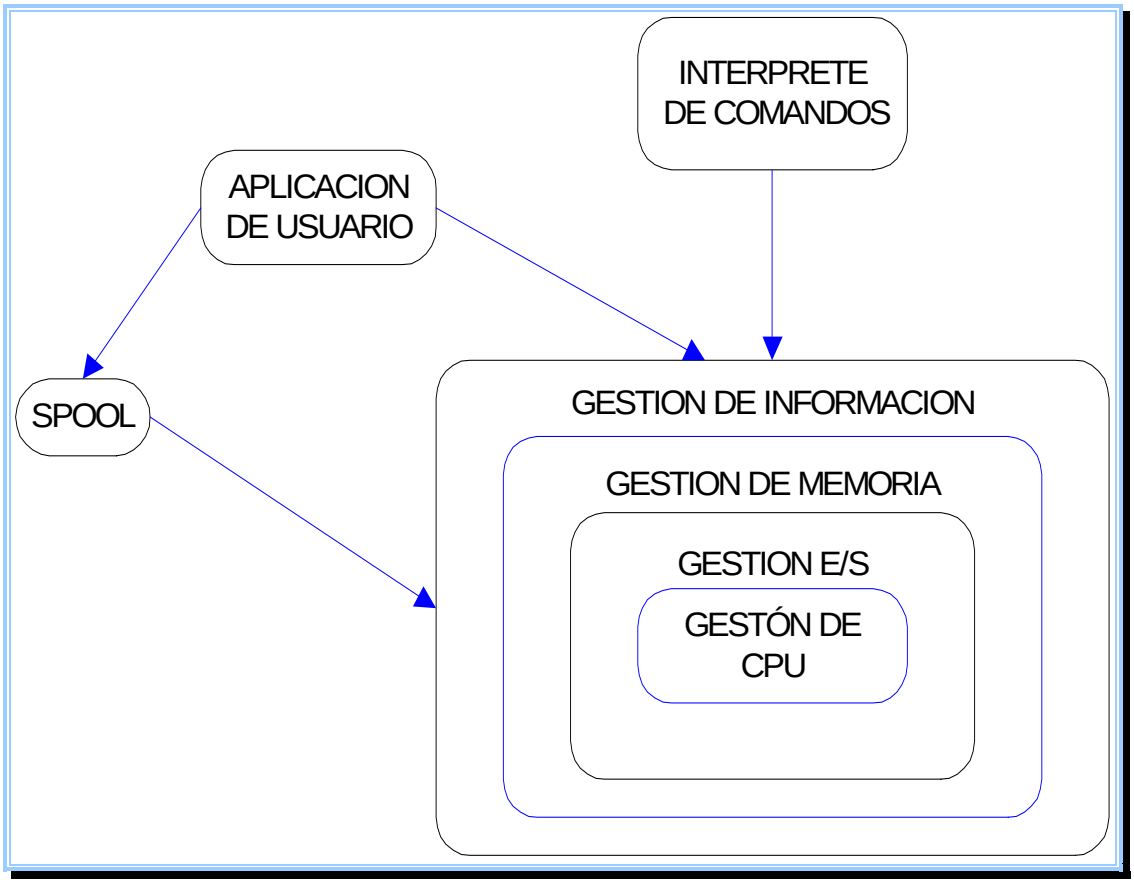


Figura 1.7 Organización de Anillos Jerárquicamente.

En el sistema de anillos, cada uno tiene una apertura, conocida como puerta o trampa (trap), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas.

Sistemas operativos de red.

Esta clasificación también se refiere a una visión externa, que en este caso se refiere a la del usuario, el cómo acceda a los servicios. Bajo esta clasificación se pueden detectar dos tipos principales: *sistemas operativos de red y sistemas operativos distribuidos*.

Son aquellos sistemas que mantienen a dos o más computadoras unidas a través de algún medio de comunicación (físico o no), con el objetivo primordial de poder compartir los diferentes recursos y la información del sistema.

El primer Sistema Operativo de red estaba enfocado a equipos con un procesador Motorola 68000, pasando posteriormente a procesadores Intel con Novell Netware.

Los Sistemas Operativos de red mas ampliamente usados son: Novell Netware, Personal Netware, Windows NT Server, UNIX, LANtastic, Windos 2005 server.

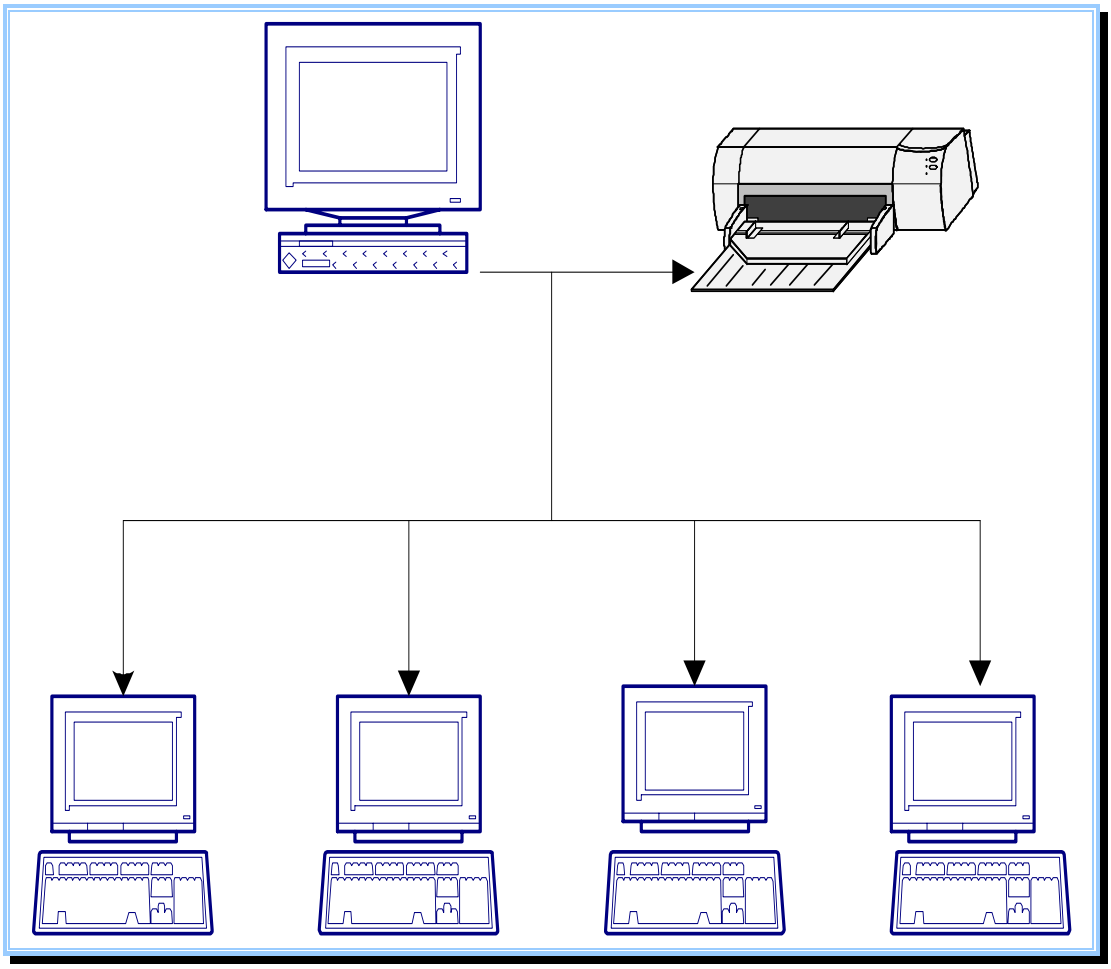


Figura 1.8, Un ejemplo de Sistema Operativo de red figura.

Sistemas operativos distribuidos.

Permiten distribuir trabajos, tareas o procesos, entre un conjunto de procesadores. Puede ser que este conjunto de procesadores esté en un equipo o en diferentes, en este caso es transparente para el usuario. Existen dos esquemas básicos de éstos. Un sistema fuertemente acoplado es aquel que comparte la memoria y un reloj global, cuyos tiempos de acceso son similares para todos los procesadores. En un sistema débilmente acoplado los procesadores no comparten ni memoria ni reloj, ya que cada uno cuenta con su memoria local.

Los sistemas distribuidos deben ser muy confiables, ya que si un componente del sistema se descompone otro componente debe de ser capaz de reemplazarlo.

Entre los diferentes Sistemas Operativos distribuidos que existen tenemos los siguientes: Sprite, Solaris-Mc, Mach, Chorus, Spring, Amoeba, Taos, etc.

Características de los Sistemas Operativos distribuidos:

- Colección de sistema autónomos capaces de comunicación y cooperación mediante interconexiones hardware y software.
- Objeto clave es la transparencia de comunicación.
- Generalmente proporcionan medios para compartir recursos.
- Servicios añadidos: denominación global, sistemas de archivos distribuidos, facilidades para distribución de cálculo (a través de comunicación de procesos internodos, llamadas a procedimientos remotos, etc.)

2. LA SEGURIDAD EN UNIX

2.1 LAS PARTES DE UN SISTEMA UNIX

Originalmente UNIX fue desarrollado para servir a pequeños grupos de personas que compartían la máquina totalmente, no existían limitaciones al acceso de un usuario a los archivos y órdenes de otros usuarios, incluso ni a los datos más sensibles destinados a mantener en funcionamiento sistema UNIX. Cualquier usuario podía tener acceso fácilmente, eliminar o, incluso suspender el sistema o, modificar archivos. **(4)**

Hay algunas versiones de UNIX en el que se ha llegado a incluir nuevas mejoras a la seguridad, poniendo un obstáculo para los usuarios no autorizados, haciendo más difícil el acceso y los agujeros de seguridad que han sido identificados y se han corregido. Esta versión de UNIX valida a los usuarios cuando se identifican mediante nombres de presentación, contraseñas que restringen los permisos de los archivos y acceso a recursos particulares.

Pero mas bien la seguridad que brinda UNIX es mucho mas que una protección de memoria, ya que cuenta con un sistema sofisticado de seguridad que controla la forma en que los usuarios acceden a los archivos, modifican las bases de datos y el aprovechamiento de los recursos. Nada de esto daría buen resultado si un sistema está mal configurado o se le da un uso inadecuado.

Las gran cantidad de fallas que se llegan a encontrar se dan por este tipo de problemas y no defectos del sistema en si (en cuanto al diseño).

- Distintos usuarios se acostumbran a que UNIX se configure de cierta manera.
- Tener acceso a la mayor parte de archivos y comandos .
- Están acostumbrados a construir sus archivos para que los pueda leer todo el mundo de forma predeterminada.
- Se acostumbran a instalar sus propios programas, lo que requiere privilegios especiales.
- La tendencia a versiones gratuitas de UNIX.

(4) Libro Introducción a UNIX, George Meghabghab, Prentice –Hall, Hispano Americana, Pagina 12.

Es por eso que estas acciones van en contra de la seguridad. Los administradores de sistemas con frecuencia tienen que restringir el acceso a los archivos y comandos que no son estrictamente necesarios para que los usuarios realicen su trabajo.

Pero por el momento el aspecto más importante es la mejora de la seguridad de un sistema UNIX tratando de dar una educación y motivación, conociendo de la mano peligros que se pueden prevenir y una buena actitud por parte de los usuarios que puede hacer cambiar la situación por que es importante que se empleen juntos para poder obtener mejores resultados.

Por otro lado los programadores ignoran o pasan por desapercibido, no hacen caso de sus experiencias con UNIX, a sobrepasar la memoria temporal es un problema importante (debido a que casi siempre al uso de memoria de longitud fija y de programas que no sirvan sus argumentos),

Pero bueno no todo es malo, UNIX con el paso del tiempo ha ido mejorando y se irán eliminando todos esos errores.

Aunque no todo es bueno. Se dice con frecuencia que UNIX es flexible y permite reutilizar programas. Por lo que se le agregan funciones a las plataformas UNIX y eventualmente se integran nuevas versiones. Desafortunadamente las nuevas características se implementan sin una compresión de las hipótesis que se hicieron en los mecanismos sin pensar en lo difícil que se ha añadido para los operadores y mantenedores. El aplicar estas características y programas en ambientes de cómputo heterogéneos de diferente naturaleza, también conduce a problemas.

Otro problema son la mejoras realizadas por cada proveedor. En lugar de tratar de presentar una interfaz simple y común para la administración de sistemas en todas las plataformas, cada vendedor ha creado un nuevo conjunto de comandos y funciones. En muchos casos estas mejoras al conjunto de comandos ayudan al administrador. Pero ahora cientos o miles de comandos nuevos, opciones, intérpretes de comandos permisos y valores que los administradores de interprete de comandos de cómputo deben entender y recordar. Además muchos comandos y opciones son similares, pero tienen significados diferentes dependiendo del intérprete de comandos en el que se usen. El resultado de esto puede conducir a desastres si el administrador se confunde momentáneamente. Esta complejidad dificulta el desarrollo de herramientas que tratan

de ofrecer soporte y control entre distintas plataformas. De los sistemas operativos estándar, UNIX es el menos estándar desde el punto de vista de la administración.

Puede ser a la vez una ventaja y desventaja de UNIX su naturaleza robusta le permite aceptar y soportar nuevas aplicaciones a partir de las viejas. Pero los mecanismos existentes a veces son completamente inapropiados para las tareas que se le asignan.

Por que lo que resta ver como va caminando UNIX si no se puede cambiar y el intérprete de comandos en el que se usa, lo que se debe hacer es aprender a proteger los sistemas lo mejor posible. Como bien sabemos basta con un solo error en un programa del sistema UNIX que puede comprometer la seguridad de todo el Sistema Operativo, es por ello que la vigilancia y atención son indispensable para que un sistema funcione en forma segura. Si por alguna razón se encuentra un problema, es necesario cubrirlo.

Antes de entrar en la seguridad en UNIX, hablare a cerca de las principales partes de que se compone el sistema operativo UNIX.

Cada sistema de cómputo consiste en un hardware (parte física del sistema, monitores, impresora, etcétera.) y software (son programas que utiliza el hardware). Como bien sabemos el sistema operativo UNIX es software que trabaja estrechamente relacionado con el hardware. El hardware varia de sistema en sistema y de fabricante, pero por lo general incluye un procesador, una o mas terminales y cualquier cantidad de periféricos.

La unidad de procesamiento es la caja que se encarga del cómputo. Esta unidad actúa como el cerebro de la computadora, almacenando y procesando información. El tamaño del procesador puede variar desde unos cuantos centímetros a microchips. En el gabinete están incluidos los siguientes elementos.

- CPU Unidad central de procesamiento, la cual procesa datos de acuerdo con las instrucciones dadas por el software. Algunas computadoras tienen más de un cpu.
- RAM Memoria de acceso aleatorio, que proporciona almacenamiento temporal de datos para la ejecución de programas y sus datos asociados.
- Tarjetas de expansión, disco duro, etc.

Por lo regular casi todas las unidades de procesamiento incluyen una unidad de disco o algún otro medio de almacenamiento, pero (en sentido estricto) estos elementos son periféricos.

Los usuarios se comunican con la PC por medio de una terminal, una pantalla y un monitor. Un sistema UNIX puede tener una sola terminal o miles de ellas ubicadas alrededor del mundo. Aunque las terminales en sí parecen computadoras en su exterior, muchas de ellas carece de cualquier tipo de procesador. Si bien la mayor parte de las terminales sirve sólo como conducto para introducir y desplegar caracteres, algunas pueden llevar a cabo funciones más complejas.

La consola es una terminal que se conecta directamente a una conexión especial llamada puerto en una PC. El puerto y no la terminal es el que distingue la consola: ésta es parte de la estación de control del administrador del sistema. Si ocurre un error grave, UNIX presenta el mensaje adecuado en la consola; algunas actividades, como la desactivación y el mantenimiento del sistema, sólo pueden llevarse a cabo desde la consola. Además de efectuar estas funciones especiales, la consola puede usarse para las mismas tareas que cualquier otra terminal.

Entre los periféricos se incluye otros equipos que pueden conectarse a la computadora. La lista de periféricos crece a diario, unidades de disco, de disquetes, impresoras, CD-ROM, módems, sintetizadores de voz, tarjetas de fax y docenas de otros tipos de equipo.

Por sí mismo el hardware no lleva a cabo ninguna actividad. Para realizar una tarea, el hardware debe recibir instrucciones, y aquí es en donde entra el software. Para visualizado el software de una computadora UNIX, consiste en varias partes que trabajan juntas: Kernel, shell, herramientas y aplicaciones. La relación entre estas partes se muestra a continuación en la siguiente figura 2.1.

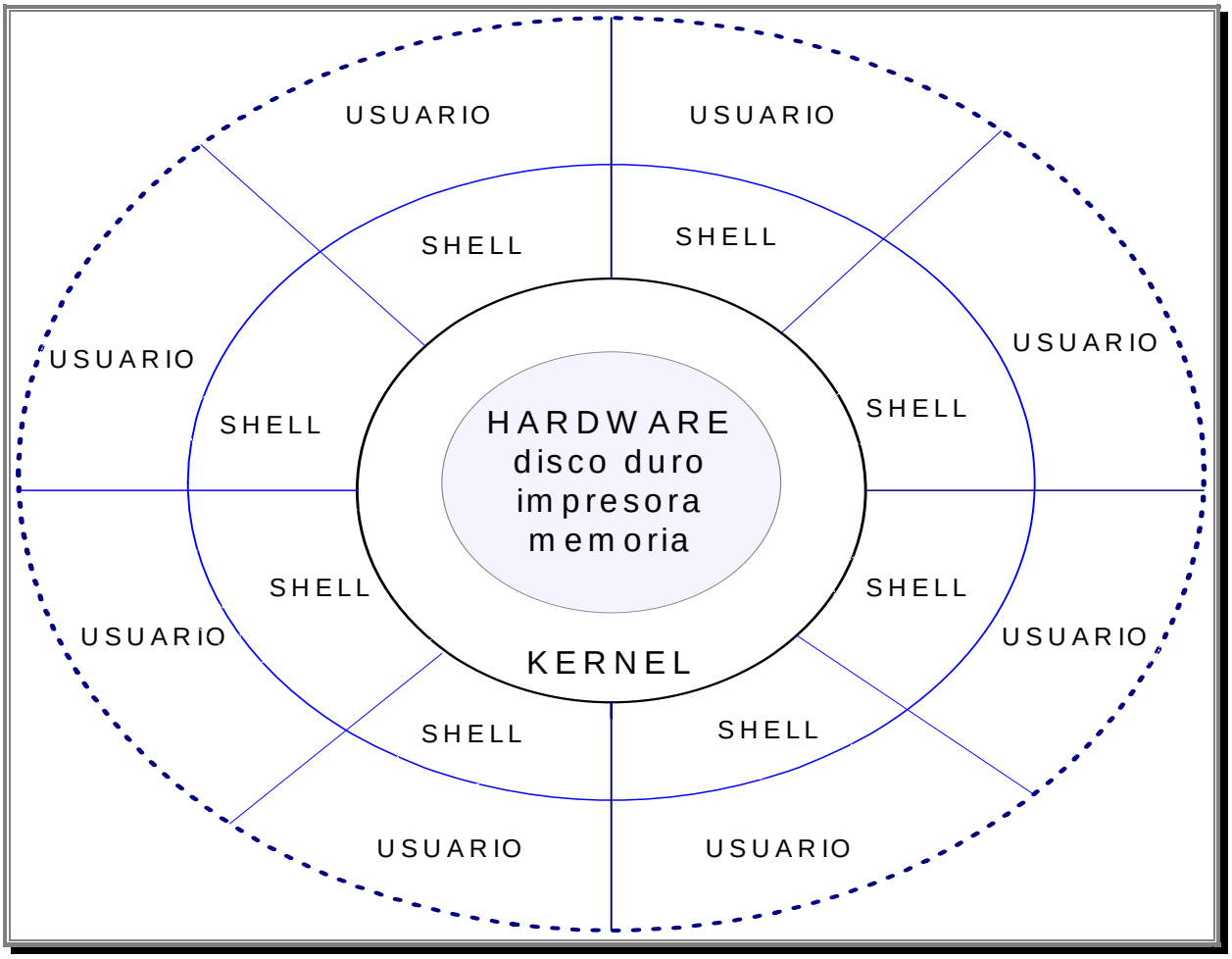


Figura 2.1. partes de un Sistema UNIX.

1. **KERNEL** O NÚCLEO de UNIX. Donde se controla y coordinan las actividades de la PC. Tras el arranque de una PC, la principal tarea consiste en cargar al Kernel. Este permanece en la memoria de la máquina hasta que es apagada. El Kernel controla todas las facetas de la operación del hardware y actúa como una capa de protección que rodea al hardware. Los programas se pueden comunicar con el hardware sólo sirviéndose del Kernel .

El Kernel funciona al mismo tiempo como patrón y obediente de los programas que se ejecutan en la máquina, los cuales reciben el nombre de procesos. El Kernel supervisa el proceso, pero también lleva a acabo tareas solicitadas por él.

Estos programas se comunican con el Kernel a través de llamadas de sistemas. Cuando el programa necesita crear un archivo, escribir utilizando una impresora, comunicarse con otro programa o efectuar muchas otras acciones posibles, utiliza la llamada de sistema solicitando al Kernel que realice la tarea adecuada. Ya que todos los UNIX utilizan las mismas llamadas de sistema, los programas UNIX son muy portátiles. (Esto de portátil no significa que el programa pueda copiarse a otra máquina y ser ejecutado; quiere decir que un programador puede hacer que el programa sea ejecutado en otra máquina con solo hacer unos cuantos cambios o no).

2. **SHELL** Es una programa que actúa como interfaz de usuario para el Kernel es la parte que vemos y utilizamos. Estos Shells han adoptado la forma de interfaces de línea de comando, pero versiones más recientes de UNIX también ofrecen shells de interfaz gráfica de usuario (**GUI**, Graphical User Interfaces), X - Windows.
3. **USUARIO** Es la parte fundamental que junto con los programas sirven como interfaz entre el usuario para el intercambio de información que hay en la computadora y el usuario, a demás de que administran los recursos de la misma, como la memoria, disco duro, procesador, monitor, etc.

Los sistemas UNIX también incluyen una amplia variedad de programas de utilería llamados herramientas o comandos. La mayor parte de las herramientas son pequeñas y relativamente sencillas; cada una efectúa una sola tarea o un grupo de tareas relacionadas. Como copiar archivos o elaborar respaldos, son secundarias en relación al trabajo principal de la computadora.

Por lo regular la gente compra PC para ejecutar aplicaciones y no para comunicarse con el sistema, pero casi ningún sistema UNIX viene con programas de aplicación. En realidad estas deben adquirirse con el proveedor de la computadora o de otros vendedores.

2.2 LA IMPORTANCIA DE LA SEGURIDAD

Concepto de seguridad: Es un conjunto de normas políticas, procedimientos y estándares que protegen y / o resguardan los bienes tangibles e intangibles de factores internos o externos. Un sistema siempre es seguro si se comporta como los usuarios esperan que lo hagan.

El punto de vista sobre seguridad otorga mayor atención a los aspectos de seguridad, como son: aspectos físicos y seguridad de los datos.

La seguridad es una inversión exitosa de los elementos de manera metódica, control de políticas, procedimientos de seguridad física y lógica de la organización así como también de los recursos involucrados.

La importancia de la seguridad.

Hoy en día en un sistema informático es uno de los principales problemas que podemos encontrar en todos los sistemas de computo. Haciendo referencia a lo que normalmente entendemos cuando nos mencionan seguridad casi siempre se piensa en el término privacidad de la información, en el que podemos incluir las contraseñas de acceso al sistema, los permisos de acceso a la información, mensajes cifrados ...etc., todo lo relacionado con la protección y confidencialidad de nuestros datos, pero echamos de menos otros aspectos importantes relacionados también con la seguridad como son los siguientes.

- **Privacidad** Se trata de evitar que la información contenida en un sistemas informático sea accesible a personas no autorizadas.
- **Disponibilidad** Nuestro sistema debe estar siempre disponible cuándo el usuario lo necesite. Evitar paradas e interrupciones en el sistema, así como la degradación de dicha información.
- **Integridad** Se debe proteger la información (datos o programas), de posibles modificaciones e incluso del borrado por parte de usuarios que no sean los propietarios de dicha información.
- **Aislamiento** En el caso de que se detecten accesos no autorizados o la aparición de programas extraños en el sistema, habrá que ver cómo se produjeron o cómo llegaron hasta allí, qué problemas han causado o pueden causar y detectar a la persona (as) responsable (s) en el suceso.
- **Consistencia** El sistema debe funcionar como se espera que lo haga. Habrá que evitar fallas en el software y hardware, en caso de que se produzcan, habrá que reaccionar lo antes posible para erradicar el problema y evitar un posible desastre.

- **Auditoría** No sólo se llegan a cometer fallos o actos maliciosos los usuarios no autorizados por ello hay que mantener un registro interno del sistema, con idea de determinar qué se ha realizado, quién lo ha realizado y qué se ha visto.

2.3 TIPOS DE SEGURIDAD

Es importante saber que no solo en los sistemas informáticos hay que proteger la información que en él se encuentra almacenada. Imagine qué pasaría si la máquina que almacena los datos se viese afectada por cualquier tipo de problema. Con esto nos estamos refiriendo a que aparte de proteger nuestros datos también existen otros tipos de seguridad.

Como la **seguridad física**, por la que nuestra máquina puede pasar por cualquier factor medio de almacenamiento físico de los datos (cintas, discos, etc.). Por lo tanto, es necesario proteger no sólo datos, sino todo tipo de elementos hardware y software que forman parte del equipo informático. Entre la multitud de factores a los que hay que hacer frente podemos enumerar los factores atmosféricos (humedad, calor, frío, etc.), que pueden ocasionar desperfectos en los medios de almacenamiento; factores naturales (terremotos, huracanes, inundaciones, etc.), que pueden causar todo tipo de destrozos intencionados, y así otros factores mas, como incendios, accidentes, descargas eléctricas, etc.

En cuanto a la **seguridad externa** hay que asegurarse de todos los accesos al sistema. Normalmente, un equipo informático no se encuentra aislado, sino que ésta conectado a otras muchas máquinas desde las cuales se puede accederá él. Esté tipo de seguridad puede ser el más complicado de controlar. Suponiendo que tenemos un sistema al cual se han conectado varias PC. Sólo las personas que tienen acceso podrían ocultar un programa que imite la ejecución de otros programas y así obtendrá contraseñas de los usuarios e incluso podría dejar residentes programas capturadores de teclas que registren todas las pulsaciones de teclado, y entre ellas las de contraseñas de los usuarios del sistemas. Algo parecido se puede hacer con la información que interconectan a las distintas máquinas. Si un usuario conoce el formato de las tramas que viajan a través de la red, podría averiguar el contenido de los paquetes que circulan de una máquina a otra e incluso podría llegar a modificar o eliminar dicha información **(5)**.

(5) Libro Seguridad en UNIX, Manuel Mediavilla, Edit. Ra-ma, página 16, 17**.**

No cabe duda que el tema de seguridad es bastante amplio, complejo pero sobre todo de gran importancia para el bien de nuestro sistema y de los datos almacenado en él.

Es por ello que no podemos perder de vista y mantener siempre presente que hay que estar alertas para enfrentarnos a los riesgos.

- La primera es la seguridad de derecho al sistema
- La segunda es la protección de los archivos y datos privados de un usuario respecto del resto de usuarios. La segunda es la protección de los archivos claves de los sistemas operativos intencionados o accidentales.
- La tercera es la seguridad física de la máquina.
- La cuarta es la protección del sistema contra determinados ataques por parte de piratas hackers experimentados que pueden dañar o destruir al sistema **(6).**

2.4 RIESGOS Y VULNERABILIDADES

Al conectarse en redes de comunicación se provoca que las vías de acceso a estos sistemas se multipliquen y, como consecuencia, el enfrentarnos a nuevos riesgos relativos a la seguridad. Por otra parte, la creciente aparición de nuevos usuarios que quieren acceder a la información a través de Internet también supone un riesgo añadido, ya que cuantos más usuarios haya, más posibilidades hay de que aparezcan nuevos usuarios maliciosos e incluso hackers con ganas de probar sus habilidades con los sistemas conectados a la red.

Un **HACKER**, o pirata informático, lo podemos considerar como una persona que disfruta intentado romper las protecciones y mecanismo de seguridad de los sistemas informáticos **(7).**

La meta de todo pirata es entrar al sistema informático, hacerse con todos sus privilegios y tenerlo al alcance. Un pirata siempre hace todo lo posible para no ser descubierto, ya que ello ya que al ser descubierto le podía causar hasta la cárcel. Pero hay de todo, desde piratas que se dedican entrar en los sistemas informáticos y provocar el mayor número posible de daños que les sea posible, hasta otros que se dedicaran a robar información confidencial para otras empresas, e incluso hay quienes sobornar a las empresas a cambio de no provocar un desastre en el sistema.

(6), (7) Libro Seguridad en UNIX, Manuel Mediavilla, Edit. Ra-ma, pagina 16, 17, 18.

A este último grupo de usuarios no se les debería llamar piratas, sino vándalos o terroristas informáticos.

Estos usuarios, que normalmente dejan sus cuentas desprotegidas, escogen una contraseña sumamente fácil de adivinar y no se preocupan demasiado por la seguridad de sus datos, suponen añadimos un administrador despreocupado por la seguridad de su sistema y que adolece de los mismos problemas que algunos de sus usuarios, el sistema se puede convertir en un auténtico problema de piratas.

Por lo tanto es, de suma importancia la existencia de un administrador en el sistema que se preocupe y mantenga una política de seguridad acorde con la importancia del sistema que administra, que sea capaz de transmitir a todos sus usuarios la importancia que la seguridad tiene en el sistema y el peligro que supone, tanto para ellos como para el resto de los usuarios y para el sistema global, inexistencia de unas normas de seguridad.

2.5 POLÍTICA DE SEGURIDAD

Políticas dentro de una máquina o red, el administrador o grupo de usuarios en conjunto deberían establecer ciertas políticas de seguridad para controlar totalmente el acceso al sistema, su protección de los mismos que deberán ser acatadas debidamente por los usuarios y administradores.

Es por eso la importancia de una política de seguridad. Dentro de las máquinas o red de máquinas, el administrador de usuarios en conjunto debería establecer una Política de seguridad para regular la asignación de nuevos **ids** de usuario, la cantidad de protección por contraseña requerida dentro del sistema y conectividad que la máquina permite a las LANS y el mundo externo. La política debería ser divulgada a los nuevos usuarios, y deberían hacerse órdenes regulares del sistema de archivos para asegurar su conformidad con está política. En caso de que el sistema sea relativamente aislado y tenga un pequeño grupo de usuarios con la misma comunidad de interés , la política de seguridad puede ser relativamente mejor. Por otra parte, si el sistema es grande tiene varios grupos de usuarios, un elevado perfil público, o contiene datos especialmente sensibles o privativos, la política de seguridad debe ser más restrictiva. La responsabilidad principal del cumplimiento de las normas de seguridad corresponden a cada

usuario individual, aunque los administradores del sistema pueden desarrollar un procedimiento de auditorias regulares con realimentación a la comunidad de usuarios.

Más allá de la política de seguridad, la regla más importante es la de conocer el sistema. Si el administrador y los usuarios del sistema utilizan frecuentemente las órdenes **ps, who, ls** y otras órdenes de información del sistema, se familiarizarán con la actividad normal día a día de la máquina y estarán alerta al estado del sistema en todo momento. Entonces las desviaciones de la norma serán rápidamente advertidas, de modo que el administrador del sistema podrá tomar acciones adecuadas para corregir errores, como las que en siguientes puntos mencionare.
Por lo que a continuación mencionare algunos ejemplo de políticas.

- **Protección de los datos frente a los otros usuarios.**

La protección frente a los otros usuarios. Cuándo un usuario comparte una máquina que los otros usuarios comparten sus datos como ya antes mencionamos, los archivos se dividen en tres niveles de permiso: los de usuarios individuales, los del grupo al que el usuario pertenece y los de todos los demás de usuarios de la máquina. Normalmente, en una máquina pequeña donde los usuarios comparten una fuerte comunidad de intereses, al administrador del sistema establecerá un único grupo para todos los usuarios, de modo que todos los usuarios puedan proteger archivos para sus usos propios. En instalaciones mayores donde existen varias comunidades no relacionadas puede haber muchos grupos diferentes.

Como por ejemplo la orden ls –l muestra los permisos de un archivo o directorio, como se muestra aquí:**(8)**.

```
$ ls -l /etc/inittab
-r- -r- - r- -  l root sys  2007 May 22 19: 49/etc/inittab
```

Aquí en este ejemplo: el propietario es root y el grupo es sys. El archivo es legible por todos, pero no es modificable ni ejecutable por ninguno.

Los archivos tienen tres grupos de permisos de cada uno de los tres niveles de seguridad: es decir, tiene acceso de lectura, escritura y ejecución para el propietario, para el grupo y para todos los usuarios. Cada archivo es propiedad de un id de presentación y pertenece a un grupo.

(8) Libro UNIX sistema versión 4, Manual de Referencia, Stephen Coffin, Edit. Osborne McGraw-Hill, página 575.

Cuando se crea un nuevo archivo, el usuario que lo crea es el propietario del archivo, y su grupo se asigna como **id** de grupo. Se puede ceder la propiedad del archivo con la orden **chown**, y se puede cambiar el grupo del archivo con **chgrp**, pero sólo si es propietario del archivo. Normalmente no se puede reclamar la propiedad una vez, que se ha cedido, aunque si un archivo es legible se puede crear una nueva copia de él con la propiedad restaurada. Sólo el superusuario puede cambiar los permisos de cualquier archivo del sistema.

- **Permisos implícitos para creación de archivos.**

Después de crear un archivo se deberían verificar sus permisos con **ls –l** para comprobar que son los que se desean. ¿Es aceptable que todos los usuarios pertenecientes al grupo tengan acceso a los datos?. ¿Debería alguien mas tener permitido la lectura o escritura del archivo?. El usuario debe preguntarse estas cuestiones y establecer los permisos para cada archivo que este vaya a crear.

El mismo sistema unix proporciona automáticamente al creador de un archivo la propiedad del mismo, y asigna al archivo el grupo de su creador. Aunque esto no puede ser modificado, se puede declarar una variable del sistema asociada con el **id** de presentación que establecerá los *permisos* de un archivo sin acción explícita por parte del usuario***. Esta variable del sistema*** se denomina **umask** (por mascara de usuario [user mask] *para creación de archivos*), y es accedida con la orden **umask**. Se puede determinar el valor actual de **umask** ejecutando la orden sin argumentos del modo siguiente:

```
$umask
000
$
```

Aquí el resultado son tres octales que se refieren a los permisos del propietario, el grupo y los otros, de izquierda a derecha. A este número se le denomina **máscara** ya que cada dígito se resta de un permiso implícito global del sistema que todos los nuevos archivos obtienen. Normalmente este permiso global es **–rw-rw-rw-,** pero sistemas y programas individuales pueden diferir de este valor. Puesto que la **umask** del usuario se resta de este valor implícito, no se pueden activar permisos con **umask** que estén normalmente desactivados, pero se pueden desactivar permisos

que están normalmente activados. Naturalmente se pueden activar explícitamente permisos con **chmod** si el usuario tiene la propiedad del archivo.

Cada dígito octal de la **umask** contiene un bit binario que borra a un permiso: un 1 borrará el permiso de ejecución, un 2 borrará el permiso de escritura y 4 borrará el permiso de lectura. Por tanto, si un dígito es cero, se utiliza el permiso implícito; por ejemplo, la **umask** anterior, 000, significa que no se alteran ninguno de los valores implícitos. El valor de **umask** 022 crearía archivos sin permisos de escritura para el grupo o para el resto. Por ejemplo.

```
$umask
000
$> def.perm
$ ls –1 def.perm
-rw-rw-rw-     1 daniel   other      0 May  10   14 : 57  def . perm
$ umask 022
$ umask
022
$ > no.escr
-rw-r—r--          1  daniel   other      0 May   10   14 : 58  no.escr
$umask 777
& > no.perm
$ls –1 no.perm
----------         1 daniel    other      0 May   10   14 : 58  no.perm
```

La **umask** implícita de 000 crea los archivos con los permisos implícitos. Cuando se redefine **umask** a 22, se crean archivos sin permiso de escritura para el resto de usuarios. la **umask** 777 desactiva todos los permisos para todos los usuarios, de modo que el último archivo del ejemplo anterior no es accesible para nadie.

Para declarar la **umask** se utiliza la orden **umask** con el código octal como argumento. Esta declaración no sobrevive a una despedida, de modo que si desea modificar, permanentemente la umask se hace uso del archivo **.profile**.

- **Cifrado de archivos.**

Editores como **ed, vi,** y **emacs** proporcionan la capacidad de crear y editar archivos cifrados. Se puede decir al editor que descifre un archivo cuando lo cargue y lo cifre de nuevo cuando lo escriba al disco. La opción **–x** especifica del modo siguiente:

```
$ vi –x fich.cifr
key:
```

Aquí el editor está solicitando la contraseña de cifrado o clave [Key]. Introduzca la clave del mismo modo que introduce una contraseña durante la presentación. Cualquier contraseña es aceptable cuando se cifra un archivo, pero naturalmente debe utilizarse la misma para descifrarlo. Como es habitual, la contraseña no produce eco. Si se introduce la contraseña correctamente, el archivo de vuelta durante o después de la sesión de edición, será cifrado de nuevo. Se puede utilizar este procedimiento para sesión de edición, será cifrado de nuevo. Se puede utilizar este procedimiento para crear un nuevo archivo o para editar un archivo existente.

Las versiones del sistema UNIX facilita un filtro para efectuar cifrado y descifrado. La orden **crypt** lee la entrada estándar y escribe en la salida estándar. Si la entrada está en texto llano, la salida será cifrada. Si la entrada está cifrada, la salida será descifrada. Al igual que el procedimiento de edición, **crypt** solicita una contraseña, tal como se muestra aquí:

```
$ cat texto.Llano | crypt
Enter Key:
```

Como es habitual la contraseña no produce eco.

Desgraciadamente, el algoritmo por el cual los archivos son cifrados en el sistema UNIX es poco conocido y hay disponibles programas que pueden reventar el algoritmo de cifrado. Por tanto, no es seguro poner demasiada confianza en los archivos cifrados, especialmente en un entorno opuesto. Los programas transgresores de cifrado funcionan analizando las frecuencias de los caracteres en los archivos cifrados y comparándolos con la frecuencias de caracteres en el

texto inglés normal. Para vencerlos se puede modificar la frecuencia de caracteres del texto normal antes del cifrado con otro filtro tal como **pack** o **compress**. Por ejemplo:

```
$compres texto.Llano
$ crypt < fich.Llano .Z >  fich.sal
```

El archivo comprimido no puede ser analizado por ningún transgresor de crypt conocido. Naturalmente, cuando se descifre el archivo hay que recordar desempaquetarlo del modo siguiente:

```
$ crypt < fich.sal > texto.Llano.Z
$uncompress texto.Llano.Z
```

Recuerde que debe comprimir antes de cifrar y descomprimir después de descifrar, naturalmente, el esquema de protección de archivos de más éxito es el que implica escribir el archivo a un disco o cinta magnética, suprimir el archivo de la máquina y poner a buen el medio magnético.

- **Ids de presentacion y contraseñas.**

El corazón del esquema de seguridad del sistema UNIX es el id presentación y la contraseña del usuario individual. Si los atacantes pueden ser mantenidos completamente fuera del sistema, no podrán causar daños. Desgraciadamente, en muchas máquinas la seguridad de la contraseña es tan pobre que incluso un infractor sin experiencia puede obtener un shell. Es responsabilidad de cada usuario defender su propia contraseña y cambiarla regularmente, de modo que con el tiempo las contraseñas pasan a ser detectables. Toda contraseña debería ser envejecida de modo que el usuario estuviese forzado a cambiarla con regularidad. Puesto que la contraseña se almacena en forma cifrada, ni siquiera el administrador del sistema puede determinar cuál es. Afortunadamente, el ataque por frecuencia de letras mencionando anteriormente no es posible con una muestra de texto tan breve como una contraseña.

La herramienta que permite modificar la contraseña es la orden **password**. Que permite modificar la contraseña, luego requiere al usuario que introduzca la nueva contraseña dos veces antes de que tenga efecto.

Es la mayoría de los sistemas UNIX existen reglas que describen una contraseña aceptable, e incluso si estas reglas no son forzados por el software del sistema proporcionan buenas líneas de guía para crear una contraseña propia. Una buena contraseña tiene como poco seis caracteres, de los cuales al menos uno (preferiblemente dos) es un carácter numérico no alfabético. Una mezcla de caracteres mayúscula y minúscula es buena, y cualquier secuencia inusual o no intuitiva de caracteres también es útil. Algunos ejemplos de contraseñas no aceptables son id de presentación, el nombre del usuario, el nombre de su hijo, su número de habitación o teléfono, su signo astrológico, su dirección, etc.

- **Historial de presentaciones.**

Algunas versiones de UNIX proporcionan una visualización de la última vez que un **id** de presentación fue utilizado. La visualización cuando el usuario se presenta ante la máquina, del modo siguiente:

```
Login: mayra
Password

Login  Last used: Wed May 22  15:11:02 2007
$
```

Esta visualización pretende alertar al usuario por si alguien más ha estado utilizando su **id** de presentación está siendo mal utilizado y debería inmediatamente tomar medidas para cambiar la contraseña.

Esta característica está mantenida por el programa **login** cuando éste verifica la contraseña. Mantiene un archivo de longitud cero denominado **.lastlogin** en el directorio propio. La fecha y hora de la última presentación del sistema, no del usuario individual, y sus permisos lo hacen difícil. El archivo lo hacen difícil de modificar, como puede verse aquí:

```
$ ls  -l $HOME /. LastLogin
-r - - - - - - - -   1 root    sys                    0 Oct   07  09:11 . LastLogin
```

Esta no es una característica de seguridad enérgica, pero avisar al usuario de si su id de presentación ha sido comprometido.

2.6 SEGURIDAD EN EL SISTEMA OPERATIVO UNIX

Como los aspectos relativos a la seguridad varían de un sistema a otro, fundamentalmente por las diferencias existentes entre los distintos sistemas operativos, nos centramos básicamente en la seguridad de los sistemas informáticos con el sistema operativo UNIX.

El sistema operativo UNIX es un sistema multiusuario y multitarea. El propio sistema operativo nos proporciona varios mecanismos de seguridad, como la protección de memoria, el control de acceso a los archivos, la protección del uso de los recursos del sistema, entre otros. Estos mecanismos son mecanismos indispensables, ya que sin ellos cualquier usuario podría acceder al espacio de memoria asignado a otro, tendría acceso a los archivos de los demás de memoria asignado a otro usuario, tendrá acceso a los archivos de los demás usuarios del sistema, etc.

El acceso en UNIX se realiza a través de una cuenta. Cada cuenta tiene asignado un nombre de usuario y una cuenta, cada cuenta pertenece a un único usuario, aunque pueden estar compartidas por varios. Cada vez que intentamos acceder al sistema, éste nos pedirá en primer lugar un nombre de usuario y a continuación la contraseña asociada a ese usuario. Si ambos son correctos, podremos acceder al sistema.

Las contraseñas son la clave que nos permitirá acceder al sistema, por lo tanto habrá que presentar especial atención a la hora de seleccionarlas, ya que la elección de una contraseña sencilla supone una vía de acceso al sistema para los piratas informáticos. En el momento en el que una contraseña sea conocida por cualquier persona que no sea su propietario legítimo, el sistema ya no estará a salvo.

Pero las contraseñas, aunque supongan la primera línea de defensas, no van a suponer un remedio infalible y por lo tanto no son el único aspecto a tener en cuenta. El administrador debe mantener su sistema perfectamente configurado y realizar controles periódicos para verificar la integridad y la consistencia del mismo. Además, debe controlar la entrada de usuarios remotos, detectar inconsistencias en los sistemas de archivos, realizar copias de seguridad de los datos, etc. Pero todo no va a ser responsabilidad del administrador. Los usuarios deberían tomar conciencia de que ellos también forman parte del sistema y por lo tanto deberían aceptar las

responsabilidades que ello conlleva. Entre ellas podemos enumerar el control del acceso a sus propios datos, la elección de contraseñas que nos sean sencillas de adivinar. Aparte de todos estos factores, siempre existen otros que no pueden ser controlados directamente ni por el administrador ni por los usuarios del sistema. Nos estamos refiriendo a fallos en los programas del sistema operativo, a los protocolos de comunicación diseñados sin tener en cuenta aspectos tan importantes como la autentificación o la confidencialidad de los datos y así un largo etc. de factores que hacen que el mantener la seguridad en un sistema informático sea algo realmente difícil

2.7 LA IMPORTANCIA DE LAS CONTRASEÑAS EN EL SISTEMA UNIX

CONTRASEÑA.

La contraseña es un mecanismo indispensable, importante para poder establecer acceso y además de evitar que cualquier usuario acceda al sistema y realizar cualquier tipo de acción no permitida.

Las Contraseñas Convencionales de UNIX por lo regular usan contraseñas sencillas para autenticar a los usuarios: el usuario conoce la contraseña y la escribe para iniciar una sesión.
Estas contraseñas convencionales han sido de UNIX desde sus primeros años.

- La ventaja de ese sistema es que funciona sin equipo especial (como lectores de tarjetas de huellas digitales).
- La desventaja de estas contraseñas es que se pueden burlar fácilmente, en especial si se inician sesiones en una computadora a través de una red.

En años recientes, las contraseñas convencionales han dejado de ser confiables en un ambiente de redes, puede ofrecer demasiadas oportunidades a un agresor para capturarlas y usarlas después.(Las contraseñas todavía son efectivas en los sistemas aislados que tienen terminales conectadas directamente). En estos tiempos los agresores sofisticados tiene disponibles en la red varias herramientas para hacerlo. La única forma segura de usar una computadora con UNIX a través de una red como Internet es usar ya sea contraseña descartable o criptografía, o ambas.

En UNIX el acceso al sistema se realiza a través de una cuenta. Cada una de estas cuentas tiene asignado un nombre de usuario o **login** y una contraseña o palabra de paso. Normalmente, cada cuenta está asignada a un usuario, que tendrá que introducir tanto el nombre de usuario como la contraseña cada vez que pretenda acceder al sistema. Es por esta razón que las contraseñas son tan importantes para la seguridad de un sistema. Una cuenta de entrada al sistema para los piratas, por lo que habrá que tener una serie de consideraciones en cuenta a la hora de que el administrador del sistema conceda las cuentas a sus usuarios.

Las contraseñas de los usuarios se almacenan en un archivo de acceso público llamado /etc/passwd. Debido a su contenido, este archivo es uno de los más conflictivos de todo el sistema, y suele ser el blanco de la mayoría de los ataques a los que se ve sometido un sistema.

El archivo passwd contiene una entrada para cada usuario. Cada entrada es una línea que consta de 7 campos separados por 2 puntos (:). La línea siguiente muestra el formato de cada línea en el archivo passwd seguida de la explicación de cada campo.

nombre-login:palabra –clave:ID-usuario:ID-grupo:info-usuario:directorio:programa

nombre-login : Este es su nombre de login, el nombre que introduce como respuesta al Indicador de login.

Contraseña: es su contraseña encriptada que en UNIX no se almacena en el archivo passwd; en lugar de ellos se almacena en un archivo llamado /etc/shadow y la letra x se utiliza como el contenedor del campo contraseña en el archivo passwd.

- ID-usuario: Este campo contiene el número de ID del usuario. El ID el usuario es un número único asignado a cada usuario, y ID de usuario 0 indica el super-usuario.
- ID-grupo: Este campo contiene el ID de grupo. El ID de grupo identifica al usuario como un miembro de un grupo.
- info-usuario: Este campo se utiliza para identificar aún más al usuario. Normalmente contiene el nombre del usuario.
- directorio: Este campo contiene la ruta de acceso absoluta del directorio de conexión asignado al usuario.

Programa: Este campo contiene el programa que se ejecutan después que el usuario se conecta. Normalmente es el programa del shell. Si no se identifican el programa, se asume que es /usr/bin/sh. Puede cambiar su shell o /usr/bin/csh para conectarse a shell, o cualquier otro programa.

Ejemplo del archivo password

```
$cat/etc/passwd
root:x: 0:1:admin:/:/usr/bin/sh
mayra:x:110:255:Mayra Hernández:/home/mayra:/usr/bin/sh
emna:x:120:255:Emma Coiffer:/home/emma:/usr/bin/sh
dani:x:130.255:Dani Gregorie:/home/dani:/usr/bin/csh
$_
```

Cómo bien sabemos un pirata se especializa en encontrar contraseñas, por lo que tocaremos como, es que lo hace ya anteriormente mencionamos que las contraseñas se encuentran almacenadas normalmente en el archivo /etc/passwd. Pero estas contraseñas no se encuentran almacenadas tal cual, sino que se utilizan para cifrar un bloque de bits mediante una función llamada crypt ( ), cuyo resultado se almacena en dicho archivo. Si un pirata consigue hacerse con una copia del archivo /etc/passwd no tendrá conocimiento de las contraseñas de los usuarios del sistema, pero puede utilizar diversas técnicas para adivinarlas. La técnica más común es la llamada ataque a la fuerza bruta o ataque con diccionario, que consiste en cifrar todas las palabras de un diccionario y ver si el resultado obtenido coincide con la clave cifrada de algún usuario a partir de la cadena cifrada de algún usuario que se almacena en el archivo /etc/passwd, ya que el algoritmo utilizado para realizarla el cifrarlo no permite la operación inversa. Para realizar este tipo de ataque, el pirata utiliza un adivinador de contraseñas. Lo normal en un principio es que el pirata ponga en marcan el adivinador de contraseñas con un diccionario formado por una serie de palabras que muchos usuarios utilizan como contraseña, como por ejemplo, los nombres de entrada de todos usuarios del sistema, palabras de uso común, marcas, nombres de persona, etc. Así como variaciones de todas ellas. Así que si queremos hacerle un poco más complicada la vida a los piratas, deberíamos saber que tipos de contraseñas deberían evitarse y cuáles son aquellas que por su dificultad para ser adivinadas se aconseja que se utilicen.

2.8 MALAS Y BUENAS CONTRASEÑAS

Malas contraseñas

Una contraseña mala es aquella que puede ser violada o adivinada.

Hay usuarios a los que el echo de tener que escribir una contraseña en el proceso de entrada al sistema le puede parecer una pérdida de tiempo o una tontería y por ello siempre tratan de elegir una contraseña lo más fácil posible y que sea rápida de escribir. Al final terminan eligiendo una palabra sumamente sencilla e incluso, en el peor de los casos, utilizan su propio nombre de usuario como contraseña.

Está claro que la elección de contraseñas fáciles de adivinar suponen una amenaza para la seguridad del sistema y no deberían utilizarse. A continuación mostramos una lista de las que deberían evitarse:

- El nombre de usuario o cualquier variación de éste (al revés en mayúsculas, etc.)
- Nuestro nombre, parte del, alias, diminutivo o cualquier variación de ellos.
- Nombres de hijos, amigos, esposa (o), novio (a), mascotas, etc.
- Cualquier información referente a uno mismo (NIP, número de la seguridad social, número telefónico, matrícula del coche, dirección, fecha de nacimiento, etc.)
- El nombre de la máquina desde la que se conecte o de su sistema operativo.
- Palabras existentes en diccionarios españoles, extranjeros o de términos técnicos (ni siquiera al revés, duplicamos, en mayúsculas, etc).
- No utilice palabras de moda, marcas conocidas, lugares geográficos o ni similares.
- Evite contraseñas con muchos caracteres repetidos y, sobre todo, aquellas formadas únicamente por números, letras o letras seguidas de un único dígito.
- Evite subcadenas cortas (como mínimo 7 caracteres).
- No utilice contraseñas cortas (como mínimo 7 caracteres).
- No utilice patrones típicos, como qwerty, 123456, abcdf o smmiliares.
- Evite cualquier variación de todo lo anterior.

Buenas contraseñas

Ya sabemos la importancia que tiene la elección de una buena contraseña para la seguridad del sistema, por ello, lo ideal sería que todas las contraseñas de un sistema fuesen difíciles de adivinar. Algunas de las normas que deberían cumplir una buena contraseña son las siguientes:

- Mezclar letras minúsculas y mayúsculas y no sólo al inicio o al final de la palabra, sino todo, en medio.
- Tener caracteres numéricos y de puntuación, espacios.
- Que sea fácil de recordar y no deben de escribirse.
- Debe ser larga (7 u 8 caracteres).
- Que sea rápida de teclear para así evitar que una persona que el mire por encima del hombro pueda adivinarla.
- Si tiene cuentas en distintas máquinas una contraseña distinta para cada una de ellos.

Si su contraseña cumple la mayoría de las normas anteriormente citadas y no se encuentra en ninguna de las categorías del apartado malas contraseñas, tenga por seguro que su contraseña es lo suficientemente resistente como para soportar cualquier tipo de ataque con diccionarios.

Contraseñas más comúnmente utilizadas por los usuarios

Como sucede con muchas otras cosas, los usuarios suelen tener los mismos hábitos a la hora de seleccionar sus contraseñas. En cualquier sistema es normal encontrar un alto porcentaje de usuarios que utilizan su nombre de usuario, nombre, apellidos o cualquier variación de éstos como contraseña, así como otros que escogen la contraseña en temas relacionados con la informática, ciencia ficción, mitología, una marca o personaje famoso o nombres de compañías multinacionales, aunque los hay más originales que prefieren una contraseña formada por la frase de moda de la temporada u otras palabras populares.

2.9 MEJORANDO LA ELECCIÓN DE UNA CONTRASEÑA

Este proceso de elección de una contraseña segura pasó a paso. En el cual mostrare a continuación como seleccionar una contraseña.

- Comprueba la velocidad con la que se escribe la contraseña. Es importante que la contraseña pueda ser escrita rápidamente, para así evitar que otras personas puedan ver cómo la escribe.
- Comprueba la longitud de la contraseña introducción. Cuánto más larga sea la contraseña, menos será la posibilidad de ser adivinada.
- Comprueba la existencia tanto de las letras mayúsculas como minúsculas. Una buena contraseña está formada tanto por letras mayúsculas como minúsculas; pero teniendo en cuenta que no es conveniente que sean mayúsculas o minúsculas únicamente la primera o la última letra, sino las letras situadas en posiciones intermedias.
- Comprueba la existencia de caracteres numéricos y de puntuación. Si añadimos números o caracteres no alfanuméricos a nuestra contraseña, estamos disminuyendo considerablemente la posibilidad de que pueda ser adivinada. Habrá que tener en cuenta que una contraseña formada únicamente por números no es una buena contraseña, así como que no es aconsejable por el único carácter numérico esté al inicio o al final de la misma.
- Comprueba la existencia de caracteres que se repitan muchas veces. No es conveniente que un carácter se repita mucho en una contraseña.
- Comprueba que la contraseña no tenga ningún tipo de subcadenas, tales como el nombre de usuario, el nombre e la máquina o información del campo del usuario. Este chequeo se realiza parta todo tipo de subcadenas, sin importar que cualquier carácter esté en mayúsculas o que las cadenas estén invertidas.
- Comprueba la existencia de subcadenas repetidas dentro de la propia contraseña. Se intentan evitar contraseñas como HolaHola.

Una vez que conocemos todos los tests que realiza el programa, vamos a ver cómo nos puede ayudar a la hora de seleccionar una contraseña segura. A continuación vamos a mostrar el proceso completo de elección de una contraseña hasta llegar a una lo suficientemente resistente.

Supongamos que hemos pensado en una contraseña como por ejemplo:

mayras (Nota: Suspenso)
está claro que esta contraseña es bastante mala por varias razones: Es una contraseña corta, sólo contiene letras minúsculas y, lo peor de todos, seguro que evitar esta última adversidad, vamos a modificar un poco la palabra para evitar que pueda ser encontrada en un diccionario: mairas (Nota: Suspenso)

A pesar de todo sigue estando suspenso, ya que sigue teniendo los mismos problemas de antes. Pero esto no es un gran problema, ya que si 6 caracteres son pocos, no nos cuesta ningún trabajo añadirle un par más de ellos para así dejarlo en 8 caracteres:

[mayras) (Nota: Aprobado (5.59))

Al menos ya hemos conseguirlo que el programa nos dé un aprobado, pero nota todavía no es lo bastante alta como para ser una contraseña lo suficiente resistente. Vamos a realizar otra pequeña modificación, consiste en intercambiar los 4 primeros caracteres de la contraseña por los 4 últimos.
ras) [may (Nota: Aprobado (6.35))

Ya parece que la nota va mejorando, pero todavía podemos darle unos pequeños retoques a la contraseña para hacerla más resistente. Por ejemplo, vamos a cambiar la ‘s’ por la ‘S’:

raS) [may (Nota: Notable (7.3))

Hemos mejorado bastante, pero todavía podemos mejorar más la nota. Una candidato para ello parece ser la ‘y’ del final, que podríamos convertirla en un ‘7’, de esta forma tendríamos todo el repertorio completo de caracteres en la contraseña (letras mayúsculas, letras minúsculas, números y caracteres no alfanuméricos).

rasS) [ma7[(Nota: Sobresaliente (9.25))

Casi hemos llegado al sobresaliente. Pero si examinamos la contraseña vemos que hemos cometido una pequeña falta, que es, ni más ni menos, que el situar nuestro único número al final de la contraseña. Eso tiene muy fácil solución intercambiamos el '[' del último lugar:.

raS) [maY) (Nota: sobresaliente (9.25))

Ya está, ya tenemos el sobresaliente. Todo ha sido como un juego y ya tenemos una contraseña casi imposible de adivinar**.**

El único problema que podemos tener es que parece algo difícil de recordar, pero es que no hay nada perfecto en este mundo. Lo único que podemos hacer es intentar hacer la contraseña lago más fácil de recordar intercambiando el ')' del cuarto lugar por el '[' del último lugar:
raS) [ma7) (Nota: Notable (9.25)).

2.10 ¿CÓMO PROTEGER Y CAMBIAR CONTRASEÑAS?

¿Cómo proteger una contraseña?

Unos de los principales aspectos que se han de tener en cuenta para no comprometer la seguridad de un sistema es que la responsabilidad de reforzar la seguridad de las contraseñas no sólo recae en el administrador, sino en gran medida, en los usuarios. El administrador no puede velar por la seguridad de todas las contraseñas del sistema. Es responsabilidad de todos los usuarios el ver por la seguridad de sus contraseñas de cualquier usuario podrían verse comprometidos no sólo los archivos de ese usuario, sino la seguridad de todo el sistema e incluso introducir una contraseña en el sistema y tener en mente algunos factores como los que se citan a continuación.

- Deben evitarse las cuentas sin contraseña.
- Es preciso comprobar que las contraseñas de aquellas cuentas que vienen predeterminadas con el sistema han sido modificadas.
- Evite que otras personas conozcan y utilicen su contraseña.
- No escriba su contraseña mientras otra persona esté mirando cómo lo hace.

- No almacene su contraseña en ningún sitio ni la envíe mediante correo electrónico.
- No comparta su contraseña con otras personas.
- Cambie su contraseña periódicamente y, sobre todo, si sospecha que otra persona la conoce.
- No escriba su contraseña en papel. Manténgala siempre en su cabeza y, sobre todo, no la olvide.
- No utilice su nombre de usuario o el nombre de la máquina como contraseña..
- No utilice la misma contraseña en distintas máquinas.
- No escriba su contraseña si un programa sospechoso se la solicita.

Entrando en detalles las contraseñas son de gran utilidad para poder tener acceso a la computadora la identidad (dos palabras que tiene un gran significado para los expertos en seguridad, pero que para el resto de nosotros quieren decir lo que todos pensamos).

Cuando damos inicio a una sesión se indica a la computadora la identidad del usuario al teclear el nombre de usuario cuando aparece el iniciador **login**. Luego respuesta con el indicador **password** se introduce la contraseña, para demostrar que el usuario es quien dice ser. Ejemplo.

login: **mahernandez**
password: **fish22d**

En cuanto se proporciona la contraseña correspondiente, inicia la sesión y permite el acceso a todos los comandos, archivos y dispositivos que le pertenecen a ese usuario. Cuando no corresponden ninguno de los datos anteriores marcados en le ejemplo, UNIX no comienza la sesión. En algunas versiones después de un determinado numero de intentos nulos la cuenta se bloquea por lo que este bloqueo tiene las siguientes funciones.

1. Ayuda a proteger al sistema de quienes tratan de adivinar una contraseña. Antes de que logren adivinar la cuenta la bloquea
2. Da aviso de que alguien ha intentado penetrar en la cuenta.

Este bloqueo es de gran ayuda evitando el uso no autorizado, pero también se puede usar para ataques de negación de servicio, o por un agresor que quiere bloquear a ciertos usuarios del sistema para evitar que lo descubran. Aunque el sistema UNIX puede dar 10 oportunidades de introducir correctamente la contraseña, en una de esas se puede llegar adivinar por lo que realmente no seria del todo tan confiable.

¿Cómo cambiar una contraseña?.

La contraseña se cambia con el comando de **passwd** de UNIX. passwd primero solicita que se introduzca la contraseña actual y luego la contraseña nueva. Al pedir la contraseña vigente, passwd evita que alguien se siente delante de una terminal que está en sesión y cambie la contraseña sin que el dueño se entere.
Por lo que UNIX pide que se escriba 2 veces la contraseña cuando se hace el cambio.

% **passwd**
Changing password for mahernandez
Old password: **fish22d**
New passwd: **cri83os2**
Retype new passwd: **cri83os2**

Después de hacer el cambio, por obvias razones la anterior ya no es útil. Es importante recordar la contraseña. Si se olvida, habrá que pedirle al administrador del sistema que la cambie para que se posible iniciar una sesión y volver a cambiarla.

3. SISTEMAS DE ARCHIVOS UNIX

3.1 ORGANIZACIÓN DEL SISTEMA DE ARCHIVO EN UNIX.

Dentro del sistema de archivos en UNIX encontramos que un archivo es una secuencia de bytes. (Un byte es un pequeño trozo de información, normalmente compuesto por 8 bits. Para nuestro propósito, un byte es equivalente a un carácter.) El sistema no impone estructura alguna a los archivos, ni asigna significado a su contenido; el significado de los bytes depende únicamente de los programas que interpretan el archivo. Además, esto es cierto no solo para archivos en disco sino también para dispositivos periféricos, cintas magnéticas, mensajes de correo, caracteres de teclados, salidas de impresoras, datos que fluyen en interconexiones, cada uno de estos archivos no es mas que un secuencia de bytes desde el punto de vista del sistema y sus programas.

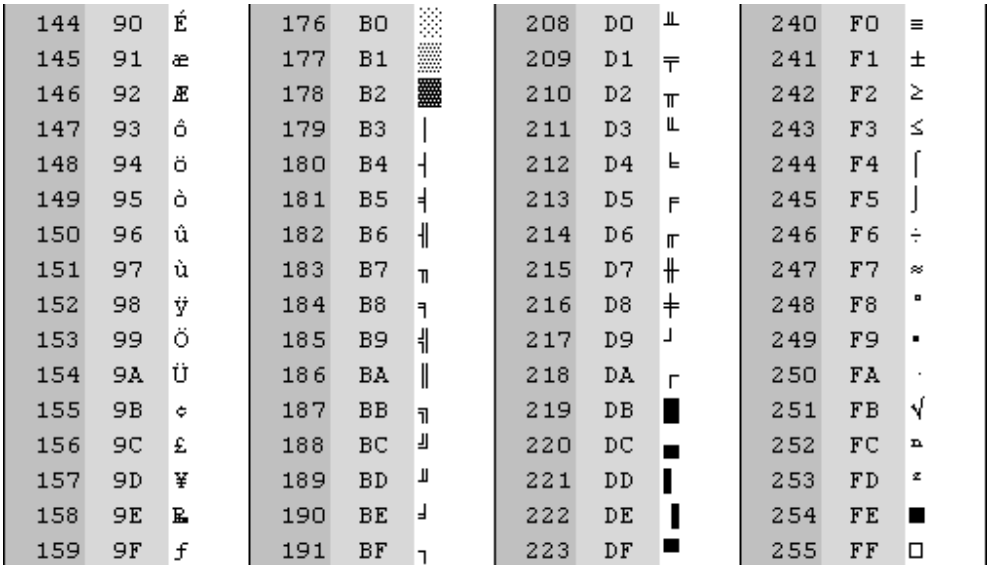
Cada byte de un archivo contiene un número de tamaño suficiente para representar un carácter. El código empleado en la mayoría de los sistemas UNIX es ASCII (“Código Norteamericano Estándar para Intercambio de Información”), Pero algunas computadoras, entre las que sobresalen las IBM, usan un código llamado EBCDIC (“Código Extendido de Intercambio Decimal Codificado en Binario”), a continuación el código ASCII.

Carácteres no imprimibles				Carácteres imprimibles								
Nombre	Dec	Hex	Car.	Dec	Hex	Car.	Dec	Hex	Car.	Dec	Hex	Car.
Nulo	0	00	NUL	32	20	Espacio	64	40	@	96	60	`
Inicio de cabecera	1	01	SOH	33	21	!	65	41	A	97	61	a
Inicio de texto	2	02	STX	34	22	"	66	42	B	98	62	b
Fin de texto	3	03	ETX	35	23	#	67	43	C	99	63	c
Fin de transmisión	4	04	EOT	36	24	$	68	44	D	100	64	d
enquiry	5	05	ENQ	37	25	%	69	45	E	101	65	e
acknowledge	6	06	ACK	38	26	&	70	46	F	102	66	f
Campanilla (beep)	7	07	BEL	39	27	'	71	47	G	103	67	g
backspace	8	08	BS	40	28	(	72	48	H	104	68	h
Tabulador horizontal	9	09	HT	41	29	)	73	49	I	105	69	i
Salto de línea	10	0A	LF	42	2A	*	74	4A	J	106	6A	j
Tabulador vertical	11	0B	VT	43	2B	+	75	4B	K	107	6B	k

Salto de página	12	0C	FF	44	2C	,	76	4C	L	108	6C	l
Retorno de carro	13	0D	CR	45	2D	-	77	4D	M	109	6D	m
Shift fuera	14	0E	SO	46	2E	.	78	4E	N	110	6E	n
Shift dentro	15	0F	SI	47	2F	/	79	4F	O	111	6F	o
Escape línea de datos	16	10	DLE	48	30	0	80	50	P	112	70	p
Control dispositivo 1	17	11	DC1	49	31	1	81	51	Q	113	71	q
Control dispositivo 2	18	12	DC2	50	32	2	82	52	R	114	72	r
Control dispositivo 3	19	13	DC3	51	33	3	83	53	S	115	73	s
Control dispositivo 4	20	14	DC4	52	34	4	84	54	T	116	74	t
neg acknowledge	21	15	NAK	53	35	5	85	55	U	117	75	u
Sincronismo	22	16	SYN	54	36	6	86	56	V	118	76	v
Fin bloque transmitido	23	17	ETB	55	37	7	87	57	W	119	77	w
Cancelar	24	18	CAN	56	38	8	88	58	X	120	78	x
Fin medio	25	19	EM	57	39	9	89	59	Y	121	79	y
Sustituto	26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
Escape	27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
Separador archivos	28	1C	FS	60	3C	<	92	5C	\	124	7C	|
Separador grupos	29	1D	GS	61	3D	=	93	5D	]	125	7D	}
Separador registros	30	1E	RS	62	3E	>	94	5E	^	126	7E	~
Separador unidades	31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

TABLA ASCII -II

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩



144	90	É	176	B0	▒	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	□

Tabla ASCII extendida de 1981 (hoja CP 437)

Los programas obtienen los datos de un archivo por medio de una llamada al sistema (una rutina del núcleo) llamada *read.* Cada vez que se demanda a *read,* esta regresa la siguiente porción de un archivo, la siguiente línea de texto tecleada en la terminal, por ejemplo *read* también indica cuantos bytes trae el archivo, por lo que al final del archivo es identificado en el momento en que *read* dice " se traen cero bytes". Si se hubieran quedado algunos bytes en el archivó, *read* los hubiera leído. En realidad, tiene sentido no representar el final de un archivo con algún valor en especial, ya que, como se menciono anteriormente, el significado de los bytes depende de como se vaya a interpretar el archivo. Pero todos los archivos tienen un final, y deben accesarse por medio de *read*, regresar un cero es una manera de representar el final de un archivo (independientemente de cualquier representación) sin introducir ningún carácter especial.

Cuando un programa lee de la terminal, el núcleo envía al programa cada una de las líneas de entrada solo cuando se teclea su carácter de nueva-línea (es decir, cuando se oprime RETURN). Por lo tanto si se comete un error mecanográfico, uno puede corregirlo si es que se lo detecta antes de teclear el carácter de nueva-línea. Si no es así la línea ya ha sido leída por el sistema y no podrá corregirse.

El sistema UNIX funciona totalmente en base a archivos. Los programas, los datos, los directorios y aún los controladores de dispositivos tales como discos, modems e impresoras, son archivos.

Archivos.

En el que cada archivo tiene un nombre, un contenido, un lugar de ubicación e información de tipo administrativo, tal como dueño y tamaño. El contenido de un archivo puede ser texto, programas fuente, programas ejecutables, imágenes, sonidos y otros.

La estructuralmente un archivo es una secuencia de bytes de datos que reside en forma semipermanente en algún medio estable, como ser una cinta magnética o un disco.

Directorios.

Un directorio es un archivo que contiene una lista de nombres de archivo e información acerca de los mismos. Dentro del sistema de archivos, un directorio es una localización capaz de contener otros directorios o archivos. Dos archivos que se encuentren en distinto directorio pueden tener el mismo nombre sin confundirse.

El comando ls permite listar el contenido de un directorio:

```
ls /var
```

El argumento es un directorio; la salida son los nombres de archivos y subdirectorios en ese directorio.

```
ls nota
```

El argumento es un archivo, la salida es el nombre del archivo.

```
ls -l /var
```

Muestra los archivos y subdirectorios contenidos en /var en formato largo.

```
ls -ld /var
```

Muestra características del directorio /var en lugar de los archivos y subdirectorios contenidos en él.

Los nombres de los archivos.

Se pueden utilizar para cualquier carácter, incluso los no visibles. Los caracteres **$; \ & ! * |** causan confusión y no conviene usarlos. Se aconseja usar solamente letras, números, punto, guión y subrayado. UNIX diferencia mayúsculas y minúsculas: el archivo **NOTA.TXT** es distinto del archivo **nota.txt** y también de **Nota.txt.** no se distingue entre nombre de archivo y extensión: nota.nueva.txt es un nombre de archivo válido, y LCK. modem también. Un archivo que comienza por un punto no es visible:

```
touch .noseve
ls
```

ls -a

ls no lo muestra, pero sí ls -a.

rm .noseve Lo borra como a cualquier archivo.

Directorio propio / Directorio actual.

Al ingresar al sistema cada usuario entra en su directorio propio, un directorio privado que no es tocado por el sistema ni por los otros usuarios. El directorio en el cual se encuentra posicionado el usuario en un momento dado se denomina su directorio actual.

cd /usr/bin

pwd

Cambia al directorio /usr/bin; el directorio actual es /usr/bin.

cd

Le devuelve al usuario a su directorio propio, que es ahora el directorio actual.

echo $HOME

Les muestra el nombre del directorio propio. HOME es una variable de ambiente que contiene el nombre del directorio propio del usuario.

cd $HOME Tiene el mismo efecto que cd.

Los Nombres de directorios.

Un directorio puede contener otros directorios así como archivos ordinarios, lo que genera un árbol o jerarquía de directorios.

cd / El directorio superior o directorio raíz se denomina /.

pwd El directorio actual es el directorio raíz.

cd Vuelve al directorio propio del usuario.

El carácter / se usa también para separar los componentes de un nombre de archivo completo: /export/home/usuario1/nota.

La porción /export/home/usuario es la ruta, nota es el nombre del archivo. Si se omite la ruta, se asume que el archivo se encuentra en el directorio actual. Un nombre tal como

textos/libro/capitulo.1 indica que su ruta comienza a partir del directorio actual. Sería lo mismo escribir ./textos/libro/capitulo.1 ya que el punto designa el directorio actual. Dos puntos seguidos designan el directorio de nivel superior al actual.

Si el directorio actual es /home/usuario1

ls ../usuario2 Muestra el contenido de los archivos en /home/usuario2

El carácter / separa directorios incluidos como parte de un nombre de archivo o directorio.

ls dir1 Es un direccionamiento relativo.

ls /home/esteban/dir1 Es un direccionamiento absoluto.

Pueden usarse comodines para referenciar directorios y archivos:

cat /home/esteban/*/*

Cambios en la jerarquía de directorios.

mkdir nuevo.dir Crea un nuevo directorio.

rmdir nuevo.dir Borra un directorio existente; actúa sólo sobre directorios vacíos.

mkdir dir1

mkdir dir1/dir2

touch dir1/dir2/arch2 dir1/arch1

ls -lR Muestra todos los archivos y directorios creados;

rm -r dir1 Borra el directorio y todos los archivos y subdirectorios que pueda contener.

¿Qué es y que hay dentro un archivo?

Un archivo es una secuencia de bytes. Un byte es equivalente a un carácter. El sistema no impone estructura alguna a los archivos ni asigna significado a su contenido; el significado de los bytes depende totalmente de los programas que interpretan el archivo.

En ningún caso hay ningún byte que no haya sido colocado por el usuario o un programa. No hay carácter de fin de archivo. El núcleo del sistema UNIX se mantiene al tanto del tamaño de los archivos sin introducir ningún carácter especial. El carácter de la nueva línea es interpretado por las terminales como nueva línea y retorno de carro (CR LF, carriage-return y line-feed), necesario

para desplegar correctamente los renglones en la terminal. Al apretar la tecla Enter el núcleo transmite a la terminal CR LF, pero en un archivo se guarda sólo LF.

¿Qué hay dentro de un archivo? El núcleo no puede decirnos nada del tipo de archivo: no lo conoce. Todos los archivos tienen la misma estructura interna. El comando file lee algunos bytes y busca señas sobre el tipo de contenido. En los sistemas UNIX hay sólo una clase de archivo. Lo único que se requiere para accederlo es su nombre.

Las propiedades de los archivos. Nos dicen que cada usuario es dueño de los archivos creados por él, hasta que los borre o los asigne a otro usuario. Cada usuario pertenece a un grupo, y puede compartir archivos con los usuarios de ese grupo. Cada archivo está asignado a un grupo de usuarios, al cual debe pertenecer su dueño. El comando ls -l muestra dueño y grupo de los archivos listados.

Tipos y permisos de Archivos.

Cada archivo tiene un conjunto de permisos asociados con él que determinan qué puede hacerse con el archivo y quién puede hacerlo.

Los permisos de un archivo se indican con 10 caracteres:

1 Carácter para tipo de archivo,

3 caracteres **[rwx]** para permisos del dueño,

3 caracteres **[rwx]** para permisos del grupo,

3 caracteres **[rwx]** para permisos de otros.

Carácter para tipo de archivo:

d directorio

l enlace simbólico

- archivo normal

b archivo controlador de dispositivo orientado a bloques

c archivo control de dispositivo orientado a caracteres

Caracteres de permisos:

r acceso de lectura (**read**)

w acceso de escritura (**write**)

x acceso de ejecución (**execute**)

Cuyo significado varía según se trate de archivos o directorios:

Permiso	Archivos	Directorios
r	leer archivos	ver contenido de directorios
w	grabar en un archivo	crear y borrar archivos
x	ejecutar como programa	ingresar a un directorio
-	sin derechos	sin derechos

El ingreso a un directorio (permiso x) permite ejecutar un archivo contenido dentro de él, o trasladarse a ese directorio; para estas operaciones no es necesario poder ver los nombres de los archivos contenidos (permiso r).

Un archivo se declara ejecutable dándole permiso de ejecución. Se ejecuta dando su nombre. Los comandos de UNIX son archivos ejecutables. Ejemplos de permisos de archivo:

rwxr--r--

rw-rw-r--

rw-------

El sistema de archivos en UNIX.

En un sistema de archivos de UNIX puede contener miles de archivos, cientos de directorios y cientos de enlaces simbólicos, dependiendo de la distribución y de lo que se haya instalado. Como referencia, la distribución Debian/GNU 2.1 viene con cerca de 2500 paquetes para instalar. Una instalación normal puede consumir un 25% en herramientas de administración, y un 10 % en herramientas de desarrollo. Borrar, alterar o cambiar permisos de archivos puede conducir a resultados impredecibles.

$ ls -F / nos permite recorrer el sistema de archivos, bajando luego a los subdirectorios. Los requerimientos de las redes cambiaron la organización funcional del sistema de archivos. En un lugar grande, la red se configura con un conjunto de máquinas heterogéneas, cada una responsable del mantenimiento y uso de cada archivo. Los archivos pueden ser específicos para

cierta máquina, compartidos por varias máquinas del mismo tipo, o accesibles a casi cualquier tipo de máquina en la red. Enlaces simbólicos hacia la anterior localización de los archivos ayudan al acceso. Un directorio puede ser real o un enlace simbólico a otro; usar /bin/pwd por si llegado al directorio directamente o a través de un enlace simbólico.

A continuación un lista de archivos y directorios mas comunes de UNIX.

/bin	Archivos ejecutables, comandos de usuario
/boot	Archivos de arranque
/cdrom	Punto de montaje para la unidad de CD-ROM
/dev	Archivos especiales de dispositivos
./dsk	Dispositivos de disco
./fd	Dispositivos descriptores de archivo
./kd	Dispositivos de teclado y despliegue
./kmem	Memoria
./null	Dispositivo para descarte de salidas
./osm	Mensajes de error del núcleo
./pts	Pseudo ttys; igual que /dev/pts*
./rdsk	Dispositivos crudos de disco
./term	Terminales; igual que /dev/tty*
./xt	Pseudo ttys; para capas DMD
/dosc	Punto de montaje para la partición DOS
/etc	Configuración de paquetes, configuración de sistema
./init.d	Scripts de arranque y detención de programas
./rc?.d	Enlaces a scripts, con K o S (Kill o Start), y número de secuencia para controlar el arranque
./skel	Archivos de inicialización para nuevos usuarios
/export	Directorios de usuarios en sistemas grandes
/floppy	Para montar una unidad de disquete
/home	Objetos relacionados con los usuarios

/lib	Bibliotecas de desarrollo y material de apoyo
/lost+found	Archivos perdidos
/mnt	Punto de montaje de dispositivos externos
/proc	Archivos de control de procesos
/root	Directorio propio para el supervisor (root)
/sbin	Archivos ejecutables de administración
/tmp	Archivos temporales
/usr	Ejecutables, documentación, referencia
./X11R6	Sistema X-Windows
./bin	Más ejecutables
./doc	Documentos de paquetes de software
./include	Encabezados .h de bibliotecas en C
./info	Archivos de info, información de UNIX (GNU)
./lib	Más bibliotecas en C
./local	Ejecutables instalados por el administrador
./man	Subdirectorios de páginas del manual
./sbin	Más archivos ejecutables de administración
./share	Compartidos
./src	(source) código fuente del kernel
/var	Archivos de log, auxiliares, archivos que crecen
./backup	Respaldo de algunos archivos del sistema
./catman	Páginas man ya formateadas
./lib	Información propia de programas
./lock	Control de bloqueos
./log	Archivos de registro de mensajes (log) del sistema
./spool	Colas de impresión, intermedios de correo y otros ./run información de procesos (PIDs)

3.1.1 TIPOS DE ARCHIVOS

Entre ellos los archivos ordinarios y los directorios, además de los archivos especiales.

Archivos normales.

Conocidos como archivos regulares o archivos ordinarios. Estos pueden contener programas de texto ASCII, código fuente, etc. **(9)**.

Directorios

También se utilizan para almacenar datos. En este caso se almacena información relacionada con otros archivos. Solo el núcleo del sistema operativo puede alternar el contenido de los directorios.

Los directorios en UNIX son archivos que contienen información que nos permite localizar a otros archivos. La estructura del directorio es que cada entrada en el directorio contiene el nombre del archivo y su número de nodo-i . La cantidad de bytes reservada para el nombre del archivo depende del sistema, aunque un valor muy utilizado son 256 caracteres. Toda la información relativa al archivo está almacenada en su nodo-i. Todos los directorios en UNIX son archivos y pueden contener cualquier número de entradas, además no existe limitación en el número de archivos o subdirectorios que se pueden almacenar en un directorio. Como sabemos los directorios se crean como mkdir y se borran como rmdir.

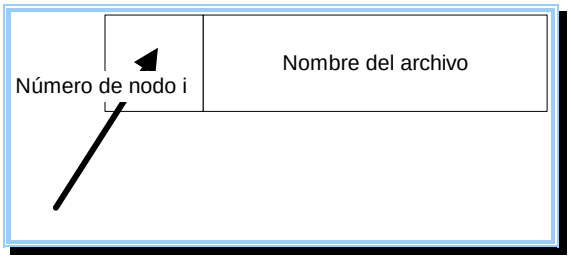


Figura 3.1 Esquema de una entrada en un directorio.

Los directorios sólo pueden ser modificados por el sistema operativo, ningún usuario tiene derecho de escritura en ellos. Incluso el administrador del sistema.

(9) Libro Unix y Linux, Sebastián Sánchez, Edit. Alfa Omega-Ra-Ma pagina 299.

Archivos especiales.

Los archivos son la parte fundamental de UNIX. el acceso a los dispositivos se realiza también mediante archivos, denominados especiales, que representan cada uno de los dispositivos físicos. Por ejemplo un puerto de impresora tiene saciado un archivo. Introducir información en ese archivo equivale a enviar información a la impresora. Analógicamente existen archivos especiales para el puerto de comunicaciones, el teclado, un terminal o una unidad de disco.

Para el usuario un archivo especial tiene el mismo (aspecto) que un archivo ordinario.

Ejemplo: puede utilizar la orden de copia para enviar unos datos a la impresora, proporcionando como archivo destino el archivo asignado a la impresora. La principal ventaja de UNIX a la hora de tratar todo como un archivo radica en que permite se lleva a cabo una verdadera independencia del dispositivo. Es decir, el acceso a un puerto serie, por ejemplo, se lleva a cabo de la misma manera en cualquier sistema UNIX. Además si sabe manejar los archivos sabrá manejar todos los dispositivos del sistema.

Además de los archivos especiales, directorios y normales también se dividen en:

Tuberías con nombre sirven para permitir comunicación entre dos procesos que se estén ejecutando en la misma salida.

Enlaces pueden ser de dos tipos, enlaces duros o enlaces blandos.

Unix trata a los archivos como simples secuencias de bytes. De este modo, al no imponerse ningún formato a los archivos, se proporciona un método más flexible para su acceso. Son en última instancia, las aplicaciones las que deben interpretar la información almacenada.

Independientemente del formato de los archivos, UNIX busca la independencia de dispositivos o, disco forma, el modo de acceder al archivo debe ser el mismo siempre, resida éste físicamente donde resida. Al soportar UNIX independencia de dispositivos, se van las mismas funciones para acceder a archivos que se encuentren en disco duro, CD-ROM, cintas, etc. **(10)**.

Los nodos-i. Son aquellos que contiene la información necesaria para que UNIX pueda manipular el archivo. UNIX mantiene una tabla con todos los nodos-i del sistema de archivos. Dentro de esta tabla cada nodo es identificado mediante un número llamado nodo-i.

- El nombre del usuario propietario del archivo.
- El tipo de archivo (ordinario, directorio...)
- El tamaño del archivo.

(10) Libro Unix y Linux, Sebastián Sánchez, Edit. Alfa Omega-Ra-Ma pagina 299.

- Dónde se encuentran almacenados los datos.
- Los permisos del archivo.
- La última vez que fue modificado el archivo.
- La última vez que se accedió, modifico a el archivo.
- El número de enlaces al archivo.

Con todo esto podemos decir que el nodo-i es la identificación del archivo, puesto que hay un solo nodo-i para cada archivo. Por lo que el número de nodo-i es único ya que especifica la posición ocupada por el nodo-i en la tabla.

3.1.2 ESTRUCTURA JERÁRQUICA

Los sistemas UNIX contienen muchos archivos organizados mediante el empleo de directorios y subdirectorios. Basándonos encontramos que tiene una estructura de árbol.

Todo el sistema de archivos de UNIX está basado en un único directorio denominado directorio raíz. De él cuelgan el resto de los directorios y archivos del sistema. Es decir, si estableciéramos una relación de parentesco entre los archivos del sistema podríamos decir que el directorio raíz es el (padre) de todos los directorios.

A continuación se muestra la estructura de directorios de cualquier sistema de archivos de una máquina UNIX. Estructura del árbol de archivos de un hipotético sistema UNIX **(11)**.

En todos los sistemas existe un directorio casa (home) por cada usuario que dispone de una cuenta en ese sistema. En el sistema de la figura 3.2 podemos ver los directorios home de dos usuario (ricardo y miguel). Habitualmente el nombre del directorio home con el login del usuario. El administrador del sistema permite que el usuario haga uso de este directorio para que almacene en él sus datos. Siempre que inicie una sesión de UNIX comenzará desde su directorio home.

(11) Libro Unix y Linux, Sebastián Sánchez, Edit. Alfa Omega-Ra-Ma pagina 299.

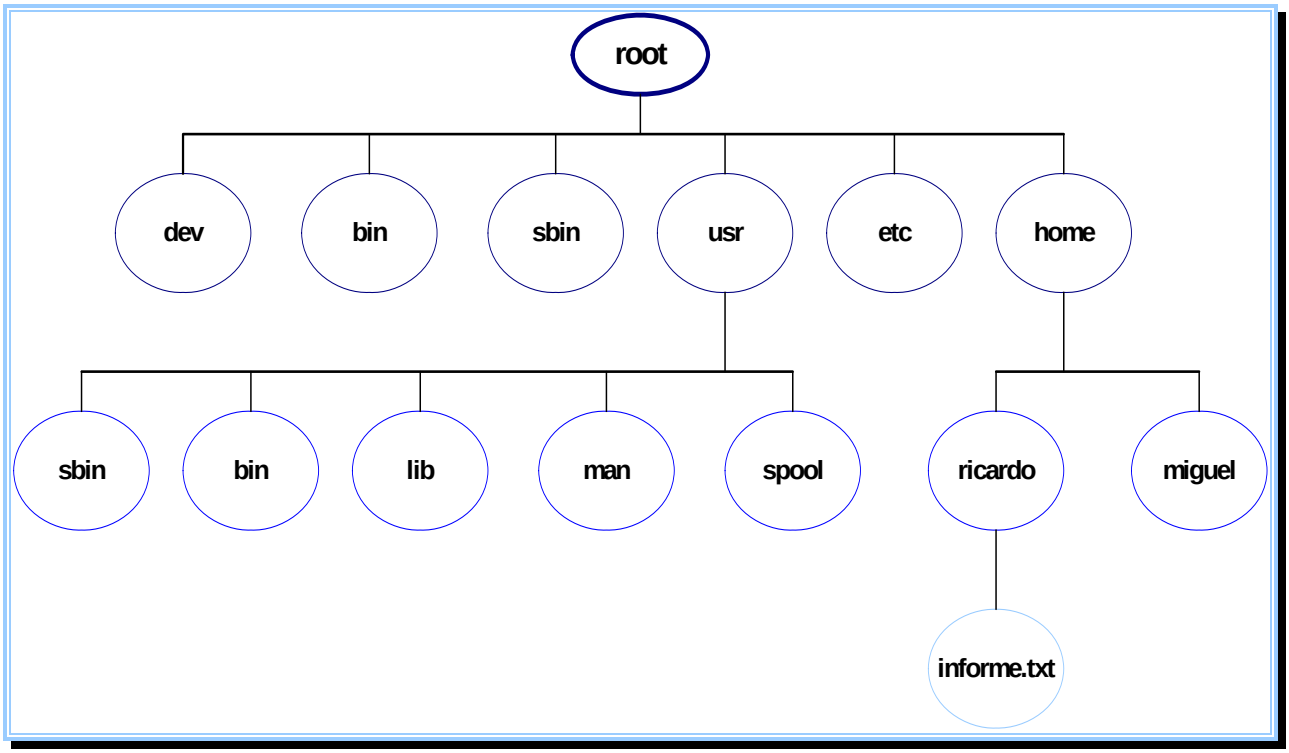


Figura 3.2 Estructura de árbol de archivos.

Lo habitual es que el administrador asigne ciertos permisos a los usuarios permitiendo que éstos puedan realizar operaciones en sus directorios home (por ejemplo, introduciendo otros archivos o creando otros subdirectorios) pero evitando que puedan modificar el resto de la estructura de directorios. El administrador también tiene competencias sobre el número de directorios que puede contener un directorio casa, pudiéndole restringir a un número máximo y evitando así que el usuario pueda descompensar el sistema de archivos **(12)**.

La división en sistemas de archivos corresponden a su ubicación física de los archivos. Se dispone de diversos discos duros cada uno de ellos contendrán uno o varios sistemas archivos. Un cd-rom o un disco floppy también constituyen un sistema de archivos. También existen de archivos remotos, situados en otros equipos accesibles a través del sistema.

Para que el contendido de un sistema de archivos sea accesible por los usuarios, el sistema debe montarse dentro de la estructura de directorios de UNIX. Ésta es una operación que lleva a cabo el administrador del sistema mediante la orden **mount**. Un usuario normal puede ejecutar esta orden sin parámetros para averiguar qué sistemas de archivos están montados.

(12) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 45

Los sistemas de archivos se montan sobre directorios, de tal manera que su contenido aparece situado bajo el directorio sobre el que están montados. De esta manera un usuario no percibe la existencia de varios sistemas de archivos, sino que ve la estructura de directorios como un bloque único.

En UNIX hay un sistema de archivos de especial importancia: el sistema de archivo raíz, o root. Es el que aparece bajo el directorio raíz (/) del sistema. se caracteriza porque está siempre montado. De él parten el sistemas de archivos.

Es el más importante porque esta siempre presente. Observe que algunas órdenes dependen de la división en sistemas de archivos de la estructura de directorios.

Lo que realmente hace la orden **mount** es mostrar los sistemas de archivos montados actualmente. Con los parámetros adecuados el directorios raíz del sistema de archivos del dispositivo con el directorio que se encuentra en el sistema de archivos raíz.

Opciones de la orden **mount:**

-t Indica el tipo de sistema de archivo que deseamos montar. Esta opción sólo es válida en el caso de que el sistema soporte diversos tipos de sistemas de archivos.
-a Monta todos los elementos de archivos incluidos en /etc/fstab.
-v Modo verboso.
-r Monta el sistema de archivos en modo sólo lectura.
-w Monta el sistema de archivos en modo lectura-escritura. Este es el modo por defecto.

Sus sintaxis : [-tahvrm] [dispositivo dir]

Ejemplo de la orden mount:

```
# mount
/dev/sda2 on / type ext2 (rw)
node on /proc type proc (rw)
/dev/sda1 on / dos type msdos (rw)
/dev/fd0 on/mnt/floppy type ext2 (rw)
#
```

En este ejemplo la orden mount sin parámetros muestra los sistemas de archivos montados en ese instante. En concreto, y de izquierda a derecha, señala lo siguiente el archivo de dispositivo correspondiente al sistema de archivos montado, el punto de montaje, el tipo de sistema y los derechos de acceso.
A diferencia de la orden mount
Sintaxis : umont dispositivo

El camino o path

Sabemos que los archivos se quedan definidos unívocamente por su nombre, tanto es así que se emplea para identificar al archivo cuando queremos hacer referencia al él. Sin embargo, para poder hacer referencia de una forma concreta a los archivos que pertenecen a un árbol de direcciones es necesario su camino, ruta o path. Este no es más que una secuencia de caracteres que contiene los nombres de los directorios, separados por barras (/), por lo que es necesario pasar para llegar desde el directorio raíz hasta el archivo en cuestión. **(13).**

Ejemplo: de la figura 3.2 anterior el camino para llegar a el archivo informe.txt es

/home/ricardo/informe.txt

El camino permite especificar la situación de un archivo en el árbol de directorios. En este caso el camino comienza en el directorio raíz (/). Los caminos de este tipo se denominan caminos completos, o absolutos y son un procedimiento para hacer referencia de una forma única a la situación de un archivo a la situación en el árbol de directorios.
Los sistemas de archivos UNIX suelen estar situados en dispositivos de almacenamiento modo bloque, tales como crear discos. Las unidades de cinta generalmente se reservan únicamente para realizar copias de seguridad o los famosos backups para instalar el sistema operativo. El núcleo del sistema trabaja con el sistema de archivos a un nivel lógico y no trata directamente con los dispositivos físicamente. Cada disco es considerado como un dispositivo lógico que tiene asociados unos números de dispositivos, estos números se utilizan como índices dentro de una tabla de funciones del núcleo para determinar cuál de ellas es necesario emplear para manejar el disco.

(13) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 46

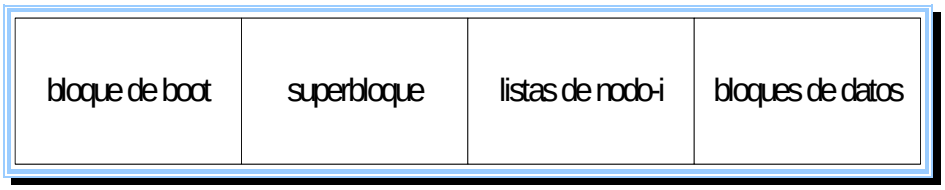


Figura 3.3 estructura del sistema de archivos UNIX

3.1.3 PERMISOS PARA ACCEDER A DIRECTORIOS

Los directorios tienen modos de permisos que funcionan de una forma similar al de los archivos. Sin embargo, los permisos de acceso a directorios tienen significados diferentes:
Lectura, escritura y ejecución.

UNIX gestiona los accesos a los archivos (directorio, dispositivo, archivos normales) por medio del triplete lectura-escritura-ejecución que se repite en tres ocasiones, una primera para el propietario, la segunda para el grupo y la tercera para los demás. Para los archivos (normales, dispositivos, conexiones con nombre) los derechos de acceso son:

- Lectura **(r)** lee el contenido del directorio, es decir él numero de nodo-i y su nombre. Por ejemplo, el usuario no puede ejecutar en el mandatos ls –l, sino solamente ls. El permiso de lectura **(r)** en un directorio significa que puede usar la orden **ls** para obtener la lista de los nombres de archivo.
- Escritura **(w)** escribe en el directorio, es decir crea y elimina objetos (archivos, directorios
- Ejecución **(x)** El permiso de ejecución **(x)** en un directorio significa que puede usar la orden **cd** para cambiar a ese directorio o usar el nombre del directorio como parte de un nombre de camino.

La orden **chmod** se emplea para cambiar el modo de accesos al archivo. Dar permiso de acceso a todos los usuarios en el directorio llamado mi_bin

$ chmod 777 mi_bin [Retorno] ..... Cambia el modo de acceso de un directorio
$_..... Hecho, se devuelve el indicador.

Como los permisos de archivos cada digito representa uno de los grupos de usuarios (propietario, grupo y otros). El dígito 7 significa autorizar el acceso de escritura, lectura y ejecución a un grupo particular de usuarios.

El ejemplo siguiente ilustra la importancia de los derechos de acceso sobre los directorios.

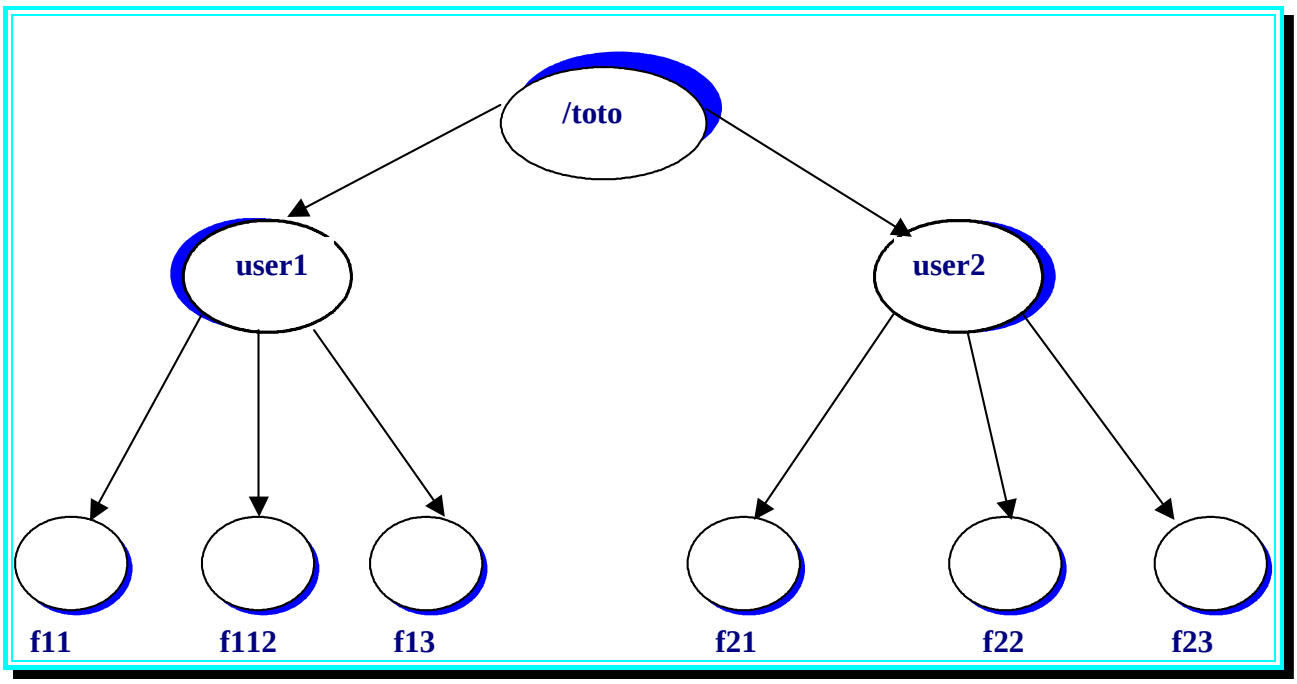


Figura 3.4 ejemplo de directorios.

Con los derechos de acceso posicionados como en el ejemplo 3.4, los archivos accesibles a user1 para lectura son :

- **f11, f12, f13.**

No tiene acceso a ninguno de los archivos de user2 porque no tiene derecho en ejecución sobre el directorio /toto/user2 para group.

Los archivos accesibles en modo lectura para user2 son:

- **f21, f22, f23.**
- **f11** y **f13** (el archivo **f12** requiere el derecho de lectura para el grupo)

Los archivos que puede suprimir user2 son:

f21, f22, f23.

f11, f12, f13 (el directorio /toto/user1 tiene el derecho en ejecución para el grupo)

Para ejecutar un programa en un directorio, solamente es necesario el derecho en ejecución sobre este último. Si el derecho en lectura no esta autorizado para este directorio, el usuario no podrá ejecutar todo programa que se encuentre en el (con los derechos de lectura-ejecución)

Todo en el sistema UNIX son archivos. El sistema de archivos es indispensable para el éxito y utilidad del sistema UNIX.

3.2 MECANISMOS DE PROTECCIÓN

Las cuentas desprotegidas vienen a ser algo parecido a una cuenta sin contraseña. Al igual que con las contraseñas, donde cada usuario debe de ser consiente de lo importante de la seguridad del sistema y proteger su cuenta y los archivos que en ella se encuentran. Es por ello que en el sistema de archivos de UNIX nos proporciona mecanismos para poder controlar el acceso y lo que se puede hacer con los que tengamos en el almacenado. Estos mecanismos son los que a continuación mencionaremos y que poseen cada uno de los archivos y directorios del sistema, (en realidad esta información se encuentra almacenada en una estructura llamada nodo índice que posee cada objeto del sistema de archivos).

- Los permisos del Propietario (UID) y Grupo (GID)

Los permisos son un conjunto de 9 bits que controlan quién puede leer, escribir, ejecutar, acceder a un determinado archivo o directorio, por lo que estos 9 bits se encuentran agrupados de tres en tres siguiendo el patrón rwx y cada uno de estos grupos de 3 bits se representa mediante un número octal **(14).**

En la tabla se muestran ocho combinaciones posibles de permisos que se pueden hacer a cada grupo de 3 bits.

Octal	Binario	Permisos
0	000	---
1	001	--x
2	010	-w-
3	011	-wx

(14) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 59.

4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

El 1ro son los permisos del archivo, el 2do son los permisos del grupo, el 3ro son los permisos del resto de los usuarios.

-rwxr-x-x 1 usr1 users 32768 Aug 30 17:50 fich1

En el caso del archivo fich1, los permisos del propietario serían rwx, que pertenecen al grupo r-w y los del resto de los usuarios a –x. Por lo que la aplicación de estos grupos depende de UID y el GID. Si nuestro UID coincide con el UID del archivo al que queremos acceder, se nos aplicarán los permisos GID coincide, se nos aplican los permisos para el grupo. De lo contrario ni el UID ni el GID, se nos aplicarán los permisos correspondientes al resto de los usuarios .

Junto a estos 9 bits aparecen otros 3 que afectan a la operación de los archivos ejecutables y directorios el bit SUID, SGID y el sticky.

- **Bits SUID y SGID (15).**

Con código octal 4000 y 2000, respectivamente, se aplican principalmente a archivos ejecutables. Estos bits permiten que el usuario que ejecuta un programa puede acceder a archivos que de otra forma no podría.

Cuando se ejecuta un programa con el bit SUID, el proceso que ejecuta el programa adopta durante toda la ejecución de éste el UID del propietario. Con el bit SGID ocurre lo mismo, excepto que la identidad que se adopta es la del grupo. Cuando un programa tiene activado el bit SUID, aparece una ’s’ en vez de una ‘x’ en los permisos del propietario del archivo. Si se activa el bit SGID aparecerá la ‘s’ en los permisos de grupo.

(15) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 60, 61,62.

- **Bit sticky**

Código octal 1000 el cual puede utilizarse tanto en programas como en directorios. Cuando se utiliza sobre un programa, éste queda continuamente cargado en memoria, de manera que no hay necesidad de hacer intercambios de páginas entre la memoria principal y secundaria.

Si este bit sticky se activa sobre un directorio, ningún usuario puede borrar o renombrar ese directorio, archivos que pertenezcan a otros usuarios. Este bit es útil especialmente en directorios de escritura pública como /tmp, ya que evita que cualquier usuario pueda borrar o modificar los archivos de los demás.

Con el bit activado podemos identificar un programa o directorio tan sólo debemos comprobar que la 'x' de los permisos del resto de los usuarios del sistema haya sido sustituida por una 't'.

Cambio de los permisos, el propietario (UID) y grupo (GID).

Los permisos anteriores se pueden modificar mediante el programa chmod. El propietario y el grupo de un archivo se pueden cambiar mediante los programas chown y chgrp, respectivamente. El programa chown sólo puede ser utilizado por el superusuario o cualquier usuario, siempre que éste sea el propietario del archivo y pertenezca al grupo al que se va a cambiar el archivo. **(16).**

Ejemplo de un extracto de un ls-l:

-rw-rw-r--1	usr1	users	249	Aug	30	13:50	fich1
-rw-------1	usr1	users	1134	Aug	29	11:54	fich2
-rwxrwxrwx1	usr1	users	839	Aug	29	11:59	fich3
-rws--x--x 1	usr2	users	16384	Aug	28	12:00	fich4
drwxrwxrwt 5	root	root	1024	Aug	10	12:30	temp

El archivo fich1 tiene permiso de lectura y escritura para el propietario y el grupo, mientras que para el resto sólo lo tiene de lectura. El archivo fich2 sólo tiene permiso de lectura y escritura para el propietario.

El archivo fich3 puede ser leído, modificado por cualquier usuario, ya que tiene los permisos de lectura y escritura para el propietario. El archivo fich4 tiene activado el SUID, ya que aparece una 's' en los permisos del propietario.

(16) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 60, 61,62.

Además puede ser ejecutado por cualquier usuario del sistema. por último tenemos un directorio llamado Temp.. sabemos que es un directorio porque a diferencia de los demás otra característica y es que tiene activado el bit sticky, ya que aparece una ‘t’ en los permiso del resto de los usuario.

Las órdenes que habría que dar establecer permisos actuales de cada uno de estos archivos son las siguientes:

chmod 664 fich1
chmod 600 fich2
chmod 777 fich3
chmod 4711 fich4
chmod 1777 temp

Es el caso de que quisiéramos cambiar el propietario de fich4 a usr1, tendríamos que ser root, y dar la siguiente orden:
chown usr1 fich4

3.2.1 PROTECCIÓN DE INICIALIZACIÓN

Varios de los programas que por lo regular ejecutamos contienen archivos de inicialización en los que se establecen opciones o valores predeterminados para las variables que manejan. Numerosos de estos ni siquiera conocemos lo que hacen y mucho menos nos interesamos en tratar. Es por ello que estos archivos son unos de los principales objetivos de los bien llamados **Hacker** , ya que estos pueden hacer modificaciones considerables sin que lo notemos, poniendo en peligro el sistema.

El inconveniente principal de los archivos es, el permiso de escritura. Con que alguno de estos archivos tenga activado este permiso para el grupo o demás usuarios, cualquier **Hacker** podría hacer cambios o insertar código no deseable. Por lo que resulta requerido que todos los archivos de inicialización tengan permiso único de escritura para el propietario, y si quiere hacer uso máximo de precaución se puede desactivar todos los permisos para el grupo y demás usuarios.

Archivos de inicialización del sistema

Archivos del sistema tales como /etc/rc.config, /sbin/init.d/*; /etc/r o /etc/rc.d/* , (el nombre y ubicación de estos archivos dependen del sistema que este utilizando). Son un lugar para que un pirata haga y coloque un mecanismo por medio del cual tenga acceso al sistema. Por lo que estos archivos y directorios deberían estar protegidos contra escritura, de modo que solo el superusuario pueda modificarlo. Con esto evitando riesgos, revisando los archivos periódicamente con el fin de no poner en riesgo al sistema.

Archivos de inicio del shell

Los siguientes archivos que mencionaremos son ejecutados por el shell cuando éste es iniciado y puede variar por el tipo de shell que estemos ocupando en este caso mencionaremos el shell de Korn.

- **Archivo /etc/profile**
- **Archivo .profile y .kshcr**

Archivo /etc/profile

Se ejecuta primer una vez que el usuario ha accedido al sistema. Conocido por todos los usuarios además que también contiene inicializaciones y valores de variables predeterminadas, así como alias globales para todo el sistema. A este archivo sólo debería acceder el supeusuario para escritura, ya que si un pirata consigue acceder a él, seria fatal lo ideal de este archivo es que los permisos sean 644.

Archivo .profile y .kshcr

Se ejecuta el archivo $HOME/.profile. existe una copia de este archivo en cada directorio de entrada de los usuarios que utilizan el shell de Korn, de manera que puede ser modificado por el usuario para adaptarlo a sus necesidades. Por último se encuentra el archivo $HOME/.kshrc, que es invocado cada vez que se ejecuta un nuevo shell. Ambos archivos deberían tener permiso de escritura sólo para el propietario.

Problemas de seguridad en los archivos de arranque del shell

Con alguno de los anteriores ya mencionados tuvieran activado el permiso de escritura, tanto para el grupo como para el resto de los usuarios, puede sufrir un ataque. Esto consiste en insertar determinados mandatos en los archivos, a manera de que se ejecuten por los shell del usuario, genere un script que le permita al pirata convertirse en ese usuario con sólo ejecutarlo. Ejemplo:

```
cp /bin/ksh /tmp/ .$(whoami)
chmod  4555 /tmp/ .$(whoami)
```

Esto es que si un pirata consigue insertar estos mandatos en el archivo /etc/profile o en el $HOME/.profile del superusuario, tendría el sistema a su entera disposición.

Mas archivos de inicialización, ente ellos los siguientes:

- **.emacs**
- **.exrc**

.emacs

Cuándo inicia este editor emacs busca un archivo de inicialización llamado $HOME/.emacs. en el que se puede incluir comandos que ejecuten secuencias de código escritas en emacs Lips que serán ejecutadas durante el proceso de carga del editor. Por lo que un pirata podría aprovechar la característica del programa y así incluir en este archivo sus propios comandos.

.exrc

Este archivo es utilizado como inicialización por la familia **vi** de editores (vi, ex, view y edit). Cuando se inicia cualquiera de ellos, buscan primero a el archivo .exrc en el directorio actual y los ejecutan de manera similar al anterior. Esta ultima característica es la que realmente supone un gran riesgo para el sistema. Por aquello de suponer que nos encontramos en un directorio público como /tmp o un directorio propiedad de otro usuario. Si ejecuta el programa **vi** y en ese directorio existe un archivo .exrc de otro usuario, éste será ejecutado igual por el editor, por lo regula los piratas suelen escribir sus propias versiones del archivo .exrc y las colocan en todos los archivos del sistema en los que tengan permiso de escritura, de forma que si un usuario ejecuta cualquiera de los programas de la familia **vi** en esos directorios, se convierte en una víctima de los Hacker.

Este es un posible archivo .exrc que puede ser utilizado por los piratas.

!(cp /bin/ksh /tmp/.ksh; chmod 4555 /tmp/ksh) &

Esta ejecución de este mandato es un shell con un bit SUID activado en el directorio /tmp. Cualquier usuario que lo ejecute se convertirá en el propietario del shell, que es precisamente el usuario que víctima de la trampa de los piratas. Para no dejar rastro, es normal que el pirata opte por borrar el archivo .exrc del directorio una vez realizada la operación y borrar la pantalla rápidamente para que el usuario no lo detecte.

Pero para esto xiste una solución que consiste en inicializar la variable de entrono EXINIT con el valor set noexrc, lo que impedirá que se ejecute el archivo .exrc al iniciarse el editor.
Otra solución consiste en verificar periódicamente el sistema de archivos buscando versiones del archivo .exrc en los directorios públicos. Para ello podemos utilizar el siguiente shell script.

```
#! /usr/bin/ksh
# findexrc.ksh –Busca en todos los directorios públicos del
# sistema de archivos de versiones piratas del archivo  '.excrc'.
PATH=/bin:/usr/bin
# Archivo a buscar.
ARCHIVO=.exrc
# UID a partir del cual realizar las comprobaciones.
UID=99
If [[$ (id – u) –eq 0]]
then
#  root

DIRS=$(find / - type d \ ( \ ( -group $(id –gn) –a –perm \  -000020 \ ) –o –perm –
000002 \ ) 2 >/dev/null)
else
# usuario normal
DIRS=$(find / -type d\ (\( -group $(id  -gn) –a –perm \ -000020\) –o –perm –
00002\) 2 >/dev/null)
```

```
fi

For I  in  $DIRS
do
  grep –h “! (“ $i/$ARCHIVO && echo  “$i/$ARCHIVO:Archivo \
conflictivo”
done 2 >/dev/null
```

Ejemplo de un shell script **(17).**

3.2.2 VARIABLES IMPORTANTES DE SHELL

La mayoría de los shells proporcionan una serie de variables que gobiernan su funcionamiento. Dichas variables permiten determinada tareas más cómodamente y de una manera estándar, aunque si llegasen a ser modificadas sin conocimiento del usuario, podrían llegar a convertirse en un problema de seguridad. Existe una gran variedad de ataques en los que están involuntariamente variables del shell, por eso es conveniente que estudiemos detenidamente algunas de ellas **(18).**

- **Variable PATH**
- **Variable IFS**
- **Variable HOME**
- **Variable TMOUT**

Variable PATH

Es una de las variables más conflictivas del shell. Esta contiene un conjunto de directorios en los que se buscarán los programas que ejecuten el usuario. Esta variable se consulta cada vez que se ejecuta una orden que no es un comando interno del shell, un alias o un nombre

(17) (18). Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 68, 69

absoluto de archivo. Normalmente, esta variable se inicializa en los archivos de arranque del shell y puede tener un aspecto como el siguiente: **(19).**

PATH=/bin./usr/bin:/usr/loal/bin

PATH=.:/bin:/usr/bin:/usr/local/bin

La diferencia entre estas y la anterior, es la inclusión del directorio actual al principio de la lista de directorios. Aunque parezca complicado, esta diferencia es suficiente para comprometer la seguridad del sistema, sobre todo si el usuario tiene privilegios especiales. Veámoslos con un ejemplo. Suponga que se encuentra en un directorio público y un pirata ha colocado en él un caballo de Troya que imita la ejecución del programa **rm,** pero además crea un shell con el bit SUID activado propiedad del usuario que ejecuta el troyano. Si la variable PATH hubiese estado inicializada de forma que el directorio actual fuese el primer directorio de la lista, habría ejecutado el caballo de Troya en vez del verdadero rm.

Se podría pensar en que probablemente el directorio actual fuese el último de la lista de directorios, pero eso tampoco es buena idea, ya que el pirata podría crear caballos de Troya con nombres parecidos a los de los programas que algún día nos equivocamos al escribir el nombre del programa, ejecutaríamos el caballo de Troya.

Variable IFS (Internal Field Separator)

Esta es utilizada por el shell como separador de campos. El valor predeterminado de esta variable son los caracteres de espacio, tabulador y nueva línea (esto quiere decir que este valor es utilizado si la variable no está inicializada) esta variable es muy útil, ya que facilita la lectura de datos y la sustitución de comandos, pero puede convertirse en un problema de seguridad si no se utiliza con precaución (de echo, algunos programas, como awk o perl, utilizan su propio separador de campos como medida de seguridad).

Los problemas se dan cuando un pirata decide utilizar el carácter ‘/’ como separador de campos, lo que puede provocar la ejecución de programas indeterminados por parte del shell.

Ejemplo: supongamos que se requiere ejecutar un programa llamado /usr/bin/less, pero un pirata ha modificado la variable IFS del shell inicializándola con el valor ‘/’.

(19) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 69, 70.

Cuando el shell analice la orden introducida por el usuario, creerá que ésta ha sido usr bin less. Como consecuencia, la orden que ejecutada no será la que el usuario introdujo, sino que se ejecutara un programa llamado usr, al que se le pasarán los parámetros bin y less. Si el pirata había creado en ese directorio un archivo llamado usr, se ejecutara este último, comprometiendo la seguridad del sistema. **(20)**

Tener cuidado cuando esta variable se va a ejecutar es un shell scrpit propiedad del superusuario y con el bit SUID activado o un programa que haga uso de las funciones system ( ) o popen ( ), ya que éstas últimas utilizan la variable IFS de igual manera que si el comando se ejecutase desde la línea de comandos. Una buena manera de evitar estos riesgos consiste en inicializarse la variable IFS al principio de la ejecución de los shell scripts o los programas que hagan uso de las funciones anteriormente mencionadas.

Variable HOME

Contiene la ruta completa de nuestro directorio de entrada. Es normal que el shell expanda el carácter '~ ' por el contenido de la variable HOME, de forma que sea más cómodo al referenciar un determinado archivo.

El peligro de esta variable aparece cuando un pirata inicializa su contenido con otro directorio distinto al de la entrada del usuario, con lo que podrá conseguir que es usuario ejecutase un archivo determinado. Ejemplo: si tenemos un shell script que hace referencia a un archivo llamado ~/. logout y antes de ejecutarlo la variable HOME es inicializada con el directorio /tmp, el archivo que se trataría de ejecutarse sería el /tmp/.logout en vez de $HOME/logout. **(21)**

Variable TMOUT.

Esta es importante ya que puede servir de ayuda a reforzar la seguridad de nuestra cuenta. Si la variable se encuentra inicializada con un valor entero positivo, el shell esperará durante el número de segundos indicado la recepción de un comando por parte del usuario.

Si sobrepasa este tiempo, el shell finaliza su ejecución. Esta variable puede ser muy útil en el caso de que dejemos nuestra cuenta desatendida durante un largo periodo de tiempo, ya que terminaría la sesión transcurrido el tiempo especificado, impidiendo que la cuenta quede abierta y pueda ser intervenida por otro usuario.

(20) (21). Libro teoría y diseño de los S.O. Juan M. Morena Pascual, Juan A. Pérez Campanero, Edit. Anaya Multimedia pagina 70,71.

Normalmente, esta variable suele inicializarse en los archivos de arranque del shell y se suele dar un valor comprendido entre 600 y 900 segundos.

Una alternativa al utilizar esta variable puede ser el programa lock, que permite bloquear el terminal con una contraseña durante un periodo de tiempo determinado. Esta última alternativa puede tener problemas, ya que si damos un periodo de tiempo demasiado corto y éste transcurre sin que hayamos regresado, el terminal se desbloquea y regresa de nuevo al shell, pudiendo quedar a merced de cualquier usuario **(22)**.

3.2.3 LA MASCARA DE CREACIÓN DE ARCHIVOS

UMASK o Mascara de creación de archivos **(User file creation mode mask)** es un número octal de tres dígitos que los sistemas UNIX utilizan para determinar los permisos de los archivos que se vayan creando. El valor preestablecido por el núcleo es 666 para archivos 777 para programas y directorios. Como estos valores no suelen ser los mas apropiados, ya que se crearían archivos y directorios sin permisos, lo normal es modificar su valor **(23)**.

El valor de umask nos indica los permisos que no habrá de darles a los archivos que automáticamente se creen en el sistema. los permisos que se le concederán a los archivos recién creados se obtienen realzando una sencilla operación aritmética 666 – umask para archivos y 777 –umask para programas y directorios.

Tabla de valores más comunes de umask y sus permisos asociados.

umask	Archivos	Progs/direct.
000	-rw-rw-rw-	drwxrwxrwx
002	-rw-rw-r--	drwxrwxr-x
006	-rw-rw----	drwxrwx--x
007	-rw-rw----	drwxrwx---
026	-rw-r--r--	drwxr-xr-x
022	-rw-r-----	drwxr-x--x
027	-rw-r-----	drwxr-x---
066	-rw-------	drwx--x--x
067	-rw-------	drwx--x---
077	-rw-------	drwx------

(22) (23) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 71, 72

UMASK (comando) su valor usualmente especificado en los archivos de arranque del shell mediante el comando interno UMASK. El valor ya preestablecido en casi todos los sistemas es 022, que concede permiso con su máscara de creación de archivos de arranque del shell y buscando una línea así.

```
# Máscara de creación de archivos
umask=022
```

- En caso de trabajar con personas de un mismo grupo, es posible que nos interese más una máscara de creación de archivos como 022, con la que todas las personas del grupo tengan acceso a leer y modificar los archivos creados.
- En caso de querer restringir mas los permisos de acceso nuestros archivos, se puede optar por umask 027, que no permite leer a los usuarios del grupo. Lo que un umask 077 permite únicamente al propietario acceder a los archivos recién creados.

3.3 ARCHIVOS DE DISPOSITIVOS Y DEL SISTEMA

Los archivos de dispositivos son el mecanismo que nos proporcionan los sistemas UNIX para comunicarse con el hardware y los dispositivos del sistema. Este mecanismo proporciona una gran flexibilidad, sobre todo a los programadores, que no se tienen que preocupar por el tipo de dispositivo que se esté utilizando, pero tiene el inconveniente de que puede comprometer la seguridad del sistema si un pirata consigue acceder a cualquiera de ellos sin autorización. Existen dispositivos, como /dev/kmem, que permiten acceder a la memoria del sistema. Lógicamente, a este archivo sólo debería poder acceder, tanto para lectura como para escritura, el superusuario, ya que si un pirata consigue acceder a él podría, entre otras muchas cosas, modificar, cambiar los permisos de los procesos, modificar los datos que se encuentren en memoria, obtener las contraseñas de los usuarios e incluso bloquear el sistema. Algo parecido pasaría con distintas particiones de los discos duros, modems, redes, terminales, etc.

Después de todos lo visto anteriormente, parece lógico pensar que hay peligro, y realmente lo es si sus permisos no están debidamente protegidos o no están correctamente establecidos.

Una práctica bastante común en los piratas informáticos consiste en crear archivos de dispositivos en el sistema tras haber accedido a él como root. Normalmente suelen crear los en directorios ocultos y darles nombres de programas del sistema. Todo archivo de dispositivos que no se encuentre en el directorio /dev puede ser considerado como sospechosos. El siguiente shell script permite detectar los archivos de dispositivos existentes en el sistema que se encuentren en un directorio que no sea /dev. **(24)**.

```
#! /usr/bin/ksh
# finddev.ksh  - Detecta los archivos de dispositivos que se
# encuentran fuera del directorio`/dev´
# Nota: Este programa debe ser ejecutado por el superusuario.

PATH=/bin:/usr/bin
Find /    \ ( -type  b –o –type c/) 2 >/dev/null | grep –v \  "^  /dev " |   xargs ls –1
```

Los archivos de dispositivos pueden ser mediante el programa mknod. Lógicamente, este programa debe estar bien protegido, de forma que sólo pueda ser ejecutado por el superusuario.

Archivos del sistema con el bit suid activado

Si un pirata consigue acceder al sistema como superusuario, lo más normal es que se las imagine de alguna manera para mantener los privilegios la próxima vez que accede al sistema, entre todas las posibilidades existentes, una de las más utilizadas es la creación de programas con el bit SUID activado propiedad del superusuario. Si el pirata coloca este tipo de programas en la cuenta de un usuario, resulta muy fácil detectar el programa con una orden como la siguiente:

```
find /  -type  f  -perm  - 004000 – exec ls –l { } \;
```

Pero si el pirata decide ocultar el programa en uno de los directorios en los que se almacenan los archivos del sistema y además le asigna el nombre de uno de estos programas, puede ser casi imposible detectarlo.

(24) Libro Seguridad en UNIX , Manuel Mediavilla, Edit. Ra-ma,, pagina 86, 87, 88.

Detección de caballos de Troya en los directorios públicos del sistema

Otro mecanismo utilizado por los piratas informáticos consiste en la creación de caballos de Troya suelen ser programas ejecutables públicos del sistema. Estos caballos de Troya suelen ser programas ejecutables para que el usuario no pueda conocer su contenido, y suelen tener nombres atractivos para que la curiosidad les haga ejecutarlos.

Por lo regular estos crean archivos .rhosts, modifican los permisos de determinados archivos del directorio de entrada del usuario, añaden líneas conflictivas en los archivos de arranque del *shell* o crean *shell* con el *bit SUID* activado. Todos estos métodos persiguen un objetivo común: obtener los privilegios del usuario que ejecutan el caballo de Troya. Para no convertirse en la víctima de este tipo de trampas, el método de otros usuarios, y menos si están en un directorio público, ya que podría tratarse de caballos de Troya. Detectar este tipo de programas es realmente complicado, pero no imposible.

3.4 EL ARCHIVO DE CONTRASEÑAS

La información crítica que controla las identificaciones de los usuarios está mantenida en un sencillo archivo de datos llamado **/etc/passwd**. Como puede verse en este archivo por todos los usuarios, pero no es modificable.

```
$ ls -l /etc/passwd
-r--r--r--    1 root     root       526 Apr 10 19:49 /etc/passwd
$
```

Estos permisos deberían ser mantenidos cuidadosamente si el archivo /etc/passwd fuera modificable por alguien, la seguridad del sistema se quebraría fácilmente. Cada usuario tiene en el archivo de contraseñas, y también son necesarios varios ids de presentación estándar globales del sistema para el correcto funcionamiento de UNIX. cada línea del archivo de contraseña consta de varios campos, delimitados por : dos puntos. El id de presentación del usuario es el primer campo de la línea y el segundo es un emplazamiento para la contraseña del usuario (x). El tercero es la representación numérica del id de usuario, y el cuarto es la representación numérica

del id de grupo. Estos dos campos actúan junto con los permisos del archivo para determinar quién puede acceder a cada archivo del sistema. el quinto campo es un comentario que generalmente contiene el nombre y la dirección del usuario. El penúltimo campo contiene el directorio propio del usuario, y el último campo contiene el nombre de camino completo del shell de presentación del usuario. Si el último campo falta, implícitamente señala a **/sbin/sh.**

En versiones más antiguas del sistema UNIX, el segundo campo contenía la contraseña real cifrada de cada usuario. Sin embargo en otras versiones se introdujo un segundo archivo en el sistema para contener la contraseña cifrada y algunos otros datos. Este archivo es **etc/shadow**, y sólo debería ser legible por **root**, tal como se muestra aquí.

```
$  ls  -l  /etc/shadow
-r--r--r--     1   root       root        187  Apr   10  19:49  /etc/shadow
$
```

El archivo **/etc/shadow** contiene los ids de presentación de usuario, sus contraseñas cifradas, un código numérico que describe cuándo fue modificada por última vez la contraseña y el número mínimo y máximo de días requerido entre cambios de contraseña.

Habrá una línea en **/etc/shadow** por cada línea de **etc/passwd.** Cuando **/etc/shadow** esta presente, en el campo de contraseña en **etc/passwd** es reemplazado por el único carácter **x**. En caso contrario, el segundo campo de **etc/passwd** aparecerá como el segundo campo del ejemplo anterior en **/etcshadow**.

El archivo **/etc/shadow** es creado por la orden **pwconv**, la cual lee en **/etc/passwd** la información necesaria. Cada vez que se modifique manualmente **/etc/passwd**, debería ejecutarse inmediatamente **pwconv** para asegurarse que os cambios sean actualizados en **etc/shadow**. Los archivos de contraseñas y **shadow** no deberían ser modificables de modo que sólo el superusuario pueda cambiarlos.

3.5 LAS CUENTAS DE USUARIO

Dentro del sistema UNIX es necesario contar con una cuenta de usuario que se compone de:

nombre de usuario (login).

contraseña (password).

Las cuentas de usuario son creadas por el administrador que en Unix es un usuario especial llamado **root** (ver más adelante). Los usuarios deberán pertenecer al menos a un grupo de usuarios ya que obligatoriamente deben tener asignado un grupo principal o grupo primario.

Cuando un usuario entra en un sistema Unix, debe identificarse indicando su **nombre** de usuario (en inglés '**login**') y su **contraseña** (en inglés '**password**'). Si se equivoca al introducir su nombre o su contraseña, el sistema le denegará el acceso y no podrá entrar por lo que esta se bloqueara al igual como sucede cuando estamos dentro la red.

Una vez se haya identificado de forma satisfactoria, el usuario podrá utilizar el sistema y ejecutar todas las aplicaciones que le sean permitidas, así como leer, modificar o borrar aquellos archivos sobre los cuales tenga permiso.

Las cuentas de usuario a parte de darnos al usuario un nombre y una contraseña, también le proporciona una **ruta para almacenar sus documentos** y su perfil generalmente dentro de la carpeta /home/nombre-usuario y comúnmente denominada **carpeta home del usuario** y un **intérprete de comandos (shell)** que le permitirá ejecutar aplicaciones.

Cuando el usuario ejecuta una aplicación, el sistema carga la aplicación en memoria y la ejecuta, se les denominan **procesos**. Los procesos en ejecución pertenecen a algún usuario. El sistema asigna a los procesos el usuario que los ejecuta. Ejemplo, si el usuario "pepe" ejecuta la aplicación "konqueror", en la lista de procesos del sistema aparecerá un nuevo proceso llamado "konqueror" cuyo propietario es "pepe". **Obligatoriamente, todos los procesos del sistema pertenecen a algún usuario**.

Cuando se crea un nuevo archivo, el propietario del archivo será el **usuario** que lo ha creado y el grupo del archivo será el **grupo principal** de dicho usuario. Ejemplo, si "pepe" cuyo grupo principal es "profes" crea un nuevo archivo llamado examen.txt, el propietario de examen.txt será "pepe" y el grupo propietario será "profes", o lo que es lo mismo, el archivo pertenecerá al usuario pepe y al grupo profes. **Obligatoriamente, todos los archivos del sistema pertenecen a algún usuario y a algún grupo**.

examen.txt pertenece al usuario pepe y al grupo profes

La cuenta de usuario le permite acceder al sistema tanto de forma presencial (sentado delante del ordenador) como de forma remota accediendo desde otro equipo por la red. Los permisos que tiene el usuario cuando utiliza el sistema presencialmente son los mismos que tiene cuando lo hace remotamente. Lo habitual es utilizar el sistema de forma remota ya que al ser Unix un sistema multiusuario, la única forma de que varios usuarios lo utilicen de forma simultánea es remotamente.

El sistema Unix codifica los usuarios con un número diferente a cada uno que es el **identificador de usuario (uid = User IDentifier)**. Internamente el sistema trabaja con el uid, no con el nombre del usuario. Normalmente a los usuarios que creemos se les asignan uids desde 1000 en adelante. Los números uid menores que 100 se reservan para usuarios especiales del sistema.

En Unix por defecto, la información de los usuarios de un sistema se guarda en el archivo **/etc/passwd**. Es un archivo de texto que puede visualizarse con cualquier editor. Cada linea del archivo /etc/passwd almacena los parámetros de un usuario. Solo puede modificarlo el administrador (root). A continuación mostramos el archivo passwd:

/etc/passwd

Las contraseñas de cada usuario se guardan encriptadas con un sistema de codificación irreversible, en el archivo **/etc/shadow** que también es un archivo de texto.

Los Grupos de usuario son útiles, para poder administrar los permisos de los usuarios de una forma más flexible, el sistema Unix permite la organización de usuarios en grupos y establecer permisos a los grupos.

Ejemplo, si en un centro educativo el grupo "profesores" tiene acceso a ciertas carpetas, cuando demos de alta un profesor nuevo, tan solo tendremos que añadirle al grupo "profesores" para que pueda acceder a todas esas carpetas. Es lo que se denomina administración de permisos por grupos.

Todos los usuarios pertenecen al menos a un grupo que es el **grupo principal del usuario**, también llamado grupo primario del usuario, pero pueden pertenecer a más grupos. En caso de que pertenezcan a más grupos, éstos serán **grupos secundarios**.

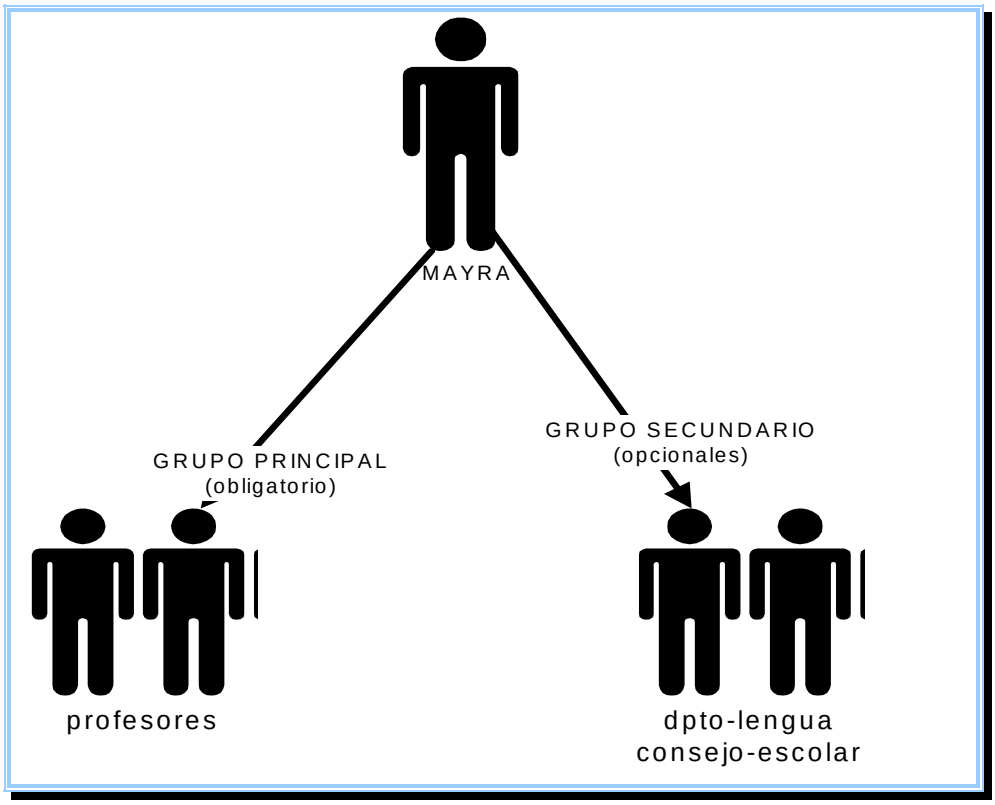


Figura 3.5 Grupos. Todo usuario debe pertenecer a un grupo principal obligatoriamente

Los grupos pueden contener varios usuarios. Los grupos de usuarios solo pueden contener usuarios, nunca podrán contener a otros grupos.

El sistema Unix codifica los grupos de usuarios con un número diferente a cada uno que es el **identificador de grupo (gid = Group IDentifier)**. Internamente el sistema trabaja con el gid, no con el nombre del grupo. Normalmente a los grupos que creemos se les asignan gids desde 1000 en adelante. Los números gid menores que 100 se reservan para grupos especiales del sistema.

En Unix por defecto, la información de los grupos de un sistema se guarda en el archivo **/etc/group**. Es un archivo de texto que puede visualizarse con cualquier editor. Cada linea del archivo /etc/group almacena los parámetros del grupo y los usuarios que contiene. Solo puede modificarlo el administrador (root). Las contraseñas de los grupos se guardan encriptadas con un sistema de codificación irreversible, en el archivo **/etc/gshadow** que también es un archivo de texto.

El usuario root, a veces llamado **superusuario**, es el usuario administrador del sistema. Está identificado con el **número de usuario cero (uid=0)** y tiene permisos sobre todo el sistema sin ningún tipo de restricción. El usuario root puede acceder a cualquier archivo, ejecutar, instalar y desinstalar cualquier aplicación, modificar los archivos de configuración del sistema y administrar usuarios. Si dispones de la contraseña de root tendrás control total sobre todo el sistema, sin excepción alguna.

Para que una persona pueda utilizar el Sistema Operativo UNIX, el administrador del sistema (root) 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 público (login) y una contraseña secreta (password). A partir de este momento, el usuario puede entrar en su cuenta a través de un terminal del ordenador. UNIX identifica a los usuarios que trabajan en la computadora cuando están registrados en el archivo /etc/passwd.

En UNIX C1 (C1 es el nivel de seguridad que soporta el sistema operativo UNIX) existen dos tipos de cuentas:

1. **Superusuario (root)**. Posee todos los privilegios del sistema. Tiene su UID=0 y no se aplican restricciones a ninguna de sus acciones en el sistema.
2. **Usuario**. Es cualquier persona que no es root. Podrán realizar aquellas acciones que les permitan los archivos a los cuales referencia.

Un usuario cualquiera puede realizar una sesión de trabajo en dos modalidades diferentes:

1. **Interactivamente**. Se conecta a su cuenta a través de un terminal y lanza comandos interactivos al sistema, esto es, esperando que el sistema le de respuesta más o menos rápidas.
2. **Mediante lotes**. El usuario puede lanzar tareas en el sistema en cualquier momento y sin requerirse su presencia en el terminal. Para ello se usa el comando at .

El superusuario

El más importante de todos los usuarios del sistema es, por supuesto, root (por convención, los sistemas UNIX definen una cuenta de usuario llamada root para el superusuario, pero puede llamarse de cualquier otra manera). El usuario **root** se caracteriza porque tiene como UID el

número 0, al cual el sistema trata de manera diferente. Suele ser habitual que el GID del superusuario sea también 0.

El superusuario tiene un control casi completo sobre todo el sistema. Puede saltarse todos los mecanismos de seguridad del sistema y acceder a todos sus recursos sin ningún tipo de restricción. Por estos motivos, el echo de la existencia de un superusuario supone uno de los mayores problemas de seguridad de los sistemas UNIX. si un pirata consigue acceder al sistema como root, podrá realizar todo lo que desee. De ahí que la mayoría de los ataques sobre sistemas UNIX se centren en la consecución de los privilegios del superusuario.

Cosas que puede hacer el superusuario.

- Mandar cualquier señal a cualquier proceso.
- Activar cualquier señal a cualquier proceso.
- Cambiar su UID de forma que pueda convertirse en cualquier usuario.
- Parar el sistema.
- Añadir, modificar o borrar cuentas de usuario.
- Ejecutar cualquier programa.
- Leer, escribir, modificar o borrar cualquier archivo.
- Acceder a cualquier dispositivo.
- Crear nuevos dispositivos.
- Modificar la fecha y la hora del sistema.
- Leer o escribir en cualquier posición de memoria.
- Montar o desmontar sistemas de archivos.
- Modificar el nombre del sistema.
- Configurar el interfaz de red.
- Sobrepasar los límites impuestos por el sistema.
- Cambiar la contraseña de cualquier usuario.

Cosas que no puede hacer el superusuario.

- Descifrar las contraseñas de los usuarios (Todavía no se conoce ningún método para descifrar las contraseñas almacenadas en el archivo /etc/passwd.
- Realizar cambios en sistemas de archivos que han sido montados como de sólo lectura.

- Demostrar un sistema de archivos que contengan archivos abiertos.

Convertirse en el Superusuario

Diversas maneras de acceder al sistema como **root**.

- La más sencilla puede ser entrar directamente en el sistema desde el indicador de entrada, pero esto en algunos casos puede resultar un inconveniente, ya que implica tener que salir de nuestra cuenta. Incluso es posible que no podamos acceder al sistema como **root** desde cualquier terminal, como veremos más adelante.
- Otra manera es utilizar el programa **su** (el nombre le viene de sustitute User, cuya traducción en sustituir usuario), el cual nos permite asumir la identidad del otro usuario sin necesidad de salir de nuestra cuenta. Si invocamos el programa **su** sin argumentos, se nos pedirá la contraseña del UID 0 o, lo que es lo mismo, con los permisos del superusuario. Los privilegios de este nuevo shell se mantendrán hasta que decidamos salir del mismo.

Una buena idea es acostumbrarse a escribir **/bin/su** en vez de escribir simplemente **su**, ya quede esta manera se esta seguro de ejecutar el verdadero **su** y no otro programa (como ejemplo un caballo de Troya).

En muchos sistemas es necesario pertenecer a un grupo de determinado (generalmente **wheel o system**) parta poder acceder al sistema como **root** mediante el programa **su**. Este impide que la gran mayoría de los usuarios puedan ni siquiera intentar acceder al sistema como **root**.

Precauciones que debe tener en cuenta al acceder al sistema como **root**.

La cuenta del superusuario no fue diseñada para el uso de personal del administrador del sistema. La verdadera misión de esta manera es evadir, en determinadas ocasiones, las protecciones que poseen los sistemas UNIX. su uso debería restringirse a ocasiones especiales, muy puntuales y en las que su uso sea la única alternativa posible. Una vez que hayamos terminado de ejecutar las acciones que requieren la intervención del superusuario, lo mejor sería abandonar su cuenta y acceder al sistema como un usuario normal.

El utilizar esta cuenta en ocasiones que nos sean las más apropiadas puede poner en peligro la integridad y la seguridad del sistema. Veamos algunos ejemplos.

- Al acceder al sistema como **root**, cualquier error al escribir una orden pueden poner en peligro al sistema. Suponga "el administrador quiere borrar del sistema todos aquellos archivos que tengan como longitud 0 bytes. Ya que piensa que si están vacíos no tiene sentido seguir conservándolos. Para ellos, ejecuta la siguiente orden:

```
find / -size  0c  -exec  rm {}  \;
```

Cuando la ejecución de esta orden haya concluido, todos los archivos de dispositivos entre otros, habrán sido borrados, aunque no fuese ésa la intención inicial.

Siempre que se vaya a acceder al sistema como root se debe procurar hacerlo desde la consola o un terminal seguro. Si los hacemos desde cualquier ordenador o terminal, es posible que la contraseña introducida sea capturada por un troyano, un capturado de teclas e incluso por un husmeador o **sniffer** (Son programas que capturan los paquetes que viajan a través de la red).

3.5.1 CUENTAS CON Y SIN CONTRASEÑAS

Cuentas con contraseñas.

Un problema muy parecido al anterior lo tenemos con aquellos usuarios que suelen escoger una contraseña fácil de adivinar, como puede ser su propio nombre de entrada o una palabra procedente de cualquier diccionario. Para evitar esto, la única solución consiste en utilizar un adivinador de contraseñas junto con un buen diccionario y utilizarlo.

Cuentas sin contraseña.

Suponen la 1er línea de seguridad de defensa del sistema. si permitimos la existencia de cuentas sin contraseña en nuestro sistema. cualquier persona que conozca la existencia de estas podrá utilizarlas para acceder a él.

Para determinar las cuentas de nuestro sistema que no tienen contraseña podemos ejecutar la siguiente orden:

awk -F: '!$2' /etc/passwd

3.5.2 CUENTAS DE USUARIOS ESPECIALES

En UNIX existen cuentas de usuarios no humanos usadas por el sistema; suelen tener un asterisco * en el campo de la contraseña en el archivo /etc/passwd o /etc/shadow, lo que impide el ingreso por estas cuentas.

daemon: propietario de archivos y procesos propios del sistema operativo no privilegiados, que sería inseguro conceder a root. Con el mismo propósito existe un grupo daemon.

bin: propietario de directorios de comandos y de muchos de los propios archivos de comandos. Actualmente existe la tendencia a dejar los archivos y directorios de comandos propiedad de root.

sys: propietario de imágenes del kernel y la memoria, tales como /dev/kmem (espacio de direcciones del kernel), /dev/mem (memoria física del sistema), /dev/drum o /dev/swap (imagen del área de intercambio del sistema). Los programas que acceden estos archivos lo hacen setuid a sys (corren con sys como dueño).

nobody: UID -1 o -2; -1 puede aparecer como 32767, lo que confunde a los programas adduser cuando determinan el siguiente UID libre. En Linux la cuenta nobody tiene UID 65534. Este usuario sin privilegios es útil para acceder archivos remotos vía red usando NFS, el sistema de exportación de archivos de UNIX, así como para correr otros comandos como finger donde sería riesgoso conceder privilegios aún de usuario común. Usuarios habilitados a hacer uso de algún servicio del sistema sin estar registrados en él, como en los servidores web, reciben la identidad nobody. La cuenta nobody no debe ser propietaria de ningún archivo.

En los sistemas UNIX existe una serie de usuarios especiales que se utilizan para realizar tareas privilegiadas. Estos usuarios suelen aparecer al principio del archivo /etc/passwd y

normalmente tienen asignado un UID concreto, que no suele ser superior al 100. En este pequeño fragmento del archivo /etc/passwd podemos ver varios de ellos. **(25).**

```
root: oNHIHNIw9sXR2:0:0:System Operator:/root:/bin/bash
daemon:*:1:1:System Background Account:/sbin:
bin:*:2:2:System Librarian Account:/bin
adm:*3:4:Administration subsystem:/var/adm:
lp:*:4:7:Line Printer Subsystem:/var/spool/lpd:
ftp:*:5:50:FTP User:/home/ftp:
mail:*:8:12:Mail Subsystem:/var/spool/mail:
news:*:9:13:News Subsystem:/var/spool/news:
uucp:*:10:14:UNIX-to-UNIX Copy:/var/spool/uucp:
operator:*:11:0:Operator:/root:
guest:Tm96MnRDV1L6:15:15:Guest:/var/spool/pc:/bin/date
nobody:*:99:99::/tmp:
```

La mayoría de estos usuarios no pueden acceder al sistema desde el indicador de entrada login, ya que tienen un `*´ como contraseña, pero sí que podemos convertirnos en ellos (siempre y cuando seamos el superusuario) utilizando el programa su o la llamada al sistema setuid ( ).

3.5.3 CUENTAS DE USO COMPARTIDAS

Las cuentas compartidas, normalmente se crean para que un grupo de personas trabajen en un mismo proyecto. La existencia de estas cuentas pueden suponer un problema de seguridad, ya que si se produce una entrada en el sistema a través de esa cuenta, nos resultará difícil quién fue la persona que propició la instrucción. Además, es normal que una cuenta a la que tengan acceso a 25 usuarios sea utilizada por más de 100.

Para evitar este tipo de problemas, lo ideal sería crear una cuenta para cada una de las personas que forman el grupo de trabajo, asignarles un grupo determinado en el sistema y establecer el **umask** de todos los usuarios del grupo a 007 para que puedan compartir los archivos. Este sistema supone un mayor trabajo para el administrador, pero a cambio mantiene el sistema mucho más seguro.

(25) Libro Seguridad en Unix, Manuel Mediavilla, Editorial Ra-ma, Pagina 80.

3.6 SHELL RESTRINGIDOS

Muchas veces tenemos en el sistema cuentas que pueden ser utilizadas por cualquier usuario para acceder al sistema o ejecutar un determinado programa. Un ejemplo típico es la cuenta del usuario **guets**. Normalmente, estas cuentas no tienen contraseña y suelen ser uno de los primeros objetivos de los piratas informáticos.

Muchos de los **shells** que solemos utilizar en el sistema tienen una versión restringida. Uno de los ellos es el *shells* de Korn. Si es invocado con la opción –r o renombrado siguiendo el patrón ***r* sh**, comenzará su ejecución en modo restringido.

Éstas son algunas de las restricciones que lleva a efecto el *shell* de Korn en modo restringido:

Se habilita el comando cd.

Impide que el usuario pueda cambiar el valor de las variables SHELL, ENV y PATH.

No permite el uso de redirecciones para crear archivos (>,>|,>>,<>).

Impide el uso de nombres absolutos o relativos de archivos.

Establecer una cuenta en el sistema con un *shell* restringido puede no ser una tarea fácil. Si olvidamos cualquier detalle, un pirata se podría introducir en el sistema y provocar un autentico desastre. A continuación daremos una serie de reglas que el administrador debería tener en cuenta al crear este tipo de cuentas:

- Se debería crear un directorio especial para cada una de estas cuentas y copiar en ellas todos los programas que el *shell* deba ejecutar. Tanto el directorio de entrada como todos los archivos de la cuenta deben tener permiso de escritura únicamente para el propietario.
- Se debe cambiar el *shell* del usuario por un *shell* restringido, de forma que la línea del archivo /etc/passwd quede como la siguiente:

  guest: Tm96SMnRDV1L6:25:25:Huésped:/home/guest:bin/rksh

- Debe comprobarse que exista un archivo **.profile** en el directorio de entrada del usuario y que este inicialice debidamente las variables PATH y SHELL

  PATH =/home/guest/bin
  SHELL=/bin/rksh
  esport PATH SHELL

- Evitar que el **path** se encuentren programas que permitan ejecutar un *shells*, tales como vi, emacs, less, mail, more, man, etc.

Una última alternativa, mucho más segura que cualquier otra, consiste en inhabilitar todo este tipo de cuentas en el sistema.

4 MEDIDAS DE SEGURIDAD A TOMAR EN CUENTA PARA UN SISTEMA LOCAL EN UNIX

4.1 PROBLEMAS DE SEGURIDAD EN EL ARCHIVO

Hay varias formas de descubrir o dañar información de modo que se deniegue el servicio. Casi rodos los ataques que se conocen se pueden evitar restringiendo el acceso a cuentas y archivos críticos, protegiéndolos de usuarios sin autorización. Si se siguen las recomendaciones para proteger la integridad del sistema también se previenen ataques de denegación del servicio. La siguiente tabla muestra algunos de los ataques posibles y medidos para prevenirlos.

ATAQUES	PREVENCIÓN
Volver a dar formato a una partición de disco o ejecutar la instrucción newfs/mkfs	No permitir que nadie use la computadora en el modo de un solo usuario. Proteger la cuenta del superusuario. Impedir físicamente que se escriba sobre los discos que se usan solo para lectura.
Borrar archivos críticos (archivos que se necesitan en /dev/etc/passwd)	Proteger los archivos y cuentas del sistema usando los modos apropiados /755 o711). Proteger la cuenta del superusuario. Fijar que el dueño de archivos montados por NFS sea root y exportarlos en modo de sólo lectura.
Apagar la electricidad que usa la computadora.	Colocar la computadora en una ubicación segura. Cerrar con llave el gabinete que contiene los interruptores de protección. (Pero se debe revisar la accesibilidad para desconexiones de emergencia. Hay que recordar que una computadora que se incendia no es una computadora muy segura).

Cortar los cables de la conexión a la red.	Los cables deben estar dentro de ductos hasta que lleguen a su destino. Restringir el acceso a las habitaciones en las que los cables estén expuestos.

Como administradores de un sistema UNIX debemos mantener su seguridad total. Aparte de hacer que cada uno de los usuarios mantenga su propia seguridad (modos de permiso de sus archivos y directorios), el administrador debe controlar todo, los archivos más importantes, cambió de las claves de acceso, desconexión de los terminales si están desocupados, etc. Vamos a ver una serie de tareas para el mantenimiento eficaz de esta seguridad:

Proteger los archivos

Se refiere al modo normal en que deben estar los modos de permisos de algunos archivos o directorios importantes.

- El directorio raíz (/) debe estar en modo 555 (dr-xr-xr-x), y a lo sumo en modo 755 (drwxr-xr-x)
- El archivo /etc/passwd debe estar sólo en modo lectura, esto es, en modo 444. nadie debe poder modificar este archivo, excepto el administrador. Muchos de los ataques al sistema UNIX se basan en la manipulación de este archivo.
- Poner los directorios del sistema tales como /usr, /lib/, /usr/lib, /bin, /usr/bin y /etc, con el modo 555
- Poner el directorio temporal /tmp con el modo 766 (sin disponibilidad de ejecución par el grupo del usuario y los démas).

Desconexión de terminales desatendidos

En otras palabras una terminal desatendida es aquélla que aunque está activa, la persona que tiene iniciada la sesión en él, se encuentra ausente momentáneamente. Existen dos formas para controlar este tipo de situaciones.

- Usar el valor de la variable de temporización **TMOUT** que provocará que finalice automáticamente este tiempo que contiene dicha para controlar este tipo de situaciones.
- No pasar por alto y utilizar **lock** o **xlock** que bloquearía el sesión o terminal **X** hasta que se introduzca nuevamente la clave correcta.

La seguridad que debe tener el administrador para su terminal

Aquí se nos menciona que excepto el administrador del sistema, se introduzca como tal, aunque sepa la clave, desde cualquier terminal que no haya dispuesto el administrador en el archivo /etc/securetty (este archivo deberá tener el modo de permisos a 500). Un ejemplo si se desea que solo el administrador pueda entrar por la consola del sistema o por el terminal número 1, el archivo /etc/securetty será de la forma:

```
#cat /etc/securetty
console
tty1
#
```

Observar los archivos de control

En ciertos sistemas se tiene de forma automática archivos que contienen información que nos permite detectar si ha habido intentos de romper la seguridad del sistema. Este control se establece en los archivos para averiguar si alguien estuvo intentando entrar y no ha podido, las personas que han entrado al sistema que guardan dicha información son los siguientes:

/usr/adm/sulog	Almacena información relativa a la orden **su**.
/etc/wtmp	Almacena información de todos los **login** con éxito.
/etc/btmp	Almacena información de todos los **login** sin éxito.

Los archivos en el sistema UNIX que mantienen información relativa a intentos erróneos de conexión, ultimas conexiones sesiones con **uucp** o **samba**, etc. Se almacenan en el directorio **/var/log.** Se aconseja vigilar los archivos contenidos en este directorio con objeto de detectar los posibles intentos de ataques al sistema, sobre todo en máquinas que se encuentren conectadas permanentemente a Internet o similar.

La protección frente a los otros usuarios. Cuándo un usuario comparte una máquina que los otros usuarios compartan sus datos como ya antes mencionamos, los archivos se dividen en tres niveles de permiso: los de usuarios individuales, los del grupo al que el usuario pertenece y los de todos los demás de usuarios de la máquina. Normalmente, en una máquina pequeña donde los usuarios comparten una fuerte comunidad de intereses, al administrador del sistema establecerá un único grupo para todos los usuarios, de modo que todos los usuarios puedan proteger archivos para sus usos propios. En instalaciones mayores donde existen varias comunidades no relacionadas puede haber muchos grupos diferentes.

Como por ejemplo la orden ls –l muestra los permisos de un archivo o directorio, como se muestra aquí:

```
$  ls  -l /etc/inittab
-r- -r- - r- -  l root sys  526 Apr 10 19: 49/etc/inittab
```

Aquí el propietario es root y el grupo es sys

Los archivos tienen tres grupos de permisos de cada uno de los tres niveles de seguridad: es decir, tiene acceso de lectura, escritura y ejecución para el propietario, para el grupo y para todos los usuarios. Cada archivo es propiedad de un id de presentación y pertenece a un grupo.

Problemas de seguridad en los archivos.

Una vez que hemos entendido los permisos de nuestro directorio de entrada, otros problemas son los derivados de la copia de archivos entre usuarios. Cuando un usuario le quiere enviar a otro un archivo, el usuario receptor suele desproteger su directorio de entrada por completo para que el otro usuario pueda copiar el archivo sin problemas. Si después olvida restaurar los permisos originales, deja la cuenta a merced de los piratas. Para evitar esto, existen varias soluciones. La más práctica consiste en enviar los archivos por correo electrónico, utilizando si fuese necesario programas, y eso no siempre ocurre. Otra alternativa, de la que hemos hablado ya, consiste en activar el permiso de lectura activado.

Medidas de seguridad que podemos tomar para nuestro sistema UNIX.

Dentro del sistema de archivos, existen diversos problemas de seguridad que podemos encontrar en el sistema UNIX relacionado con el sistema de archivo. El realizar periódicamente comprobaciones de seguridad, como a continuación se presentan permitirán tener un sistema de archivos protegido a la mayoría de los ataques.

- Comprueba que los archivos y directorios importantes del sistema tengan los permisos adecuados.
- No ejecutar programas que no le pertenezcan, a no ser que esté completamente seguro de su procedencia. Evitar ejecutar especialmente aquellos que se encuentren en directorios públicos y tengan nombres atractivos, ya que podría tratarse de algún caballo de Troya.
- Comprobar periódicamente la existencia de archivos conflictivos en el sistema (archivos con el bit SUID activado, archivos de dispositivos fuera del directorio /dev, etc.)
- Ejecutar periódicamente algún mecanismo que le permita detectar posibles modificaciones en los archivos y directorios importantes del sistema.
- Hay que establecer límites en los recursos del sistema de archivos que puedan resultar más conflictivos (números de nodos índice, tamaño del espacio de intercambio, capacidad de los discos del sistema, etc.)
- Comprobar la existencia de directorios ocultos en el sistema ocultos en el sistema

Todo esto en cuanto a los archivos, respecto alas contraseñas son la parte primordial de todo, ya que es la entrada principal por la que debe ser protegida convenientemente. A continuación puntos que deben tenerse en cuenta.

- Como opción importante utilizar siempre contraseñas difíciles de adivinar. Una contraseña con estas características debería al menos siete caracteres e incluir número y si se puede símbolos.
- El evitar utilizar contraseñas con palabras de diccionario, datos personales, palabras comunes, personas.
- No compartir las contraseñas a nadie, tener cuidado de el lugar en que los anotamos.

- Cambiar la contraseña de forma periódica, sobre todo se tiene gran responsabilidad en el puesto.
- Hay que tener desconfianza de los programas que lleguen a solicitar nuestra contraseña, podría ser una trampa.
- Si por alguna razón se llegara a tener la menor duda de sospecha de que alguien a llegado a entrar con nuestra cuenta hacer uso de razón y cambiarla en el momento.

Como siguiente opción importantísima las cuentas de usuarios, por lo que el usuario puede hacer uso.

- Proteger el directorio de entrada de su cuenta, no permita que otros usuarios puedan escribir en él.
- Proteja sus archivos de inicialización y de arranque del shell contra escritura.
- Establezca una máscara de creación de archivos segura que impida la creación de archivos con los permisos de escritura activados para el grupo y los otros.
- Cerciórese que no existan archivos que no sean de su propiedad.
- Utilice mecanismos que le permitan detectar accesos no autorizados en su cuenta.
- Para el caso del administrador del sistema debe de comprobar periódicamente el contenido, permiso de los archivos de inicialización del sistema.
- Evite la creación de cuentas compartidas en el sistema.
- Inhabilite aquellas que no hayan sido accedidas durante un determinado intervalo de tiempo.
- Evite la creación de cuentas que puedan ser accedidas por cualquier usuario. En caso de que sea necesaria de su existencia, utilice un shell restringido.
- Procure inicializar la variable IFS a un valor seguro siempre que escriba un programa o shell srcipt, sobre todo si tiene activado el bit SUID.

4.2 PROBLEMAS DE SEGURIDAD EN EL DIRECTORIO HOME

Fue falso pensar que la contraseña es la única entrada principal que encuentran los intrusos para entrar al sistema, y así llegar a los permisos de los directorios. Si nuestra cuenta no esta bien protegida, es muy probable que un intruso lo detecte y se aproveche de la situación. Las consecuencias de ello ya nos las podemos imaginar : desde el borrado de todo el contenido de

nuestra cuenta hasta la colocación de caballos de Troya en los archivos de arranque del directorio $HOME/bin. La solución a este problema puede ser asignar permisos adecuados a nuestros directorios de entrada. La decisión sobre estos ciertos permisos la establecen las necesidades del usuario, pero siempre debemos tener en cuenta que bajo ningún concepto hay que conceder permiso de lectura y acceso tanto al grupo como al resto de los usuarios. Con la siguiente orden.

chmod 755 $HOME

Desde el momento que tenemos permiso de lectura por mínimo que parezca puede llegar a ser peligroso, ya que en el se encuentran entre otros, todos los archivos de arranque del shell e inicialización de programas, es por ello que inhibir la lectura sobre nuesro directorio de entrada, pero permitiendo el acceso.

chmod 711 $HOME

Por lo que esta alternativa impide que los demás usuarios puedan ver el contenido y acceder a el archivo que tengan permiso de lectura y de los que conozcan el nombre. Esta alternativa es mucho mas flexible que la anterior, pero puede llegar a ser conflictiva si el usuario olvida quitarle el permiso de lectura a determinados archivos para los usuarios inexpertos, lo mejor es utilizar los permisos de esta ultima alternativa (700).

chmod 700 $HOME

Entre los principales problemas de seguridad con el directorio HOME son los derivados de la copia de archivos entre usuarios. Cuando un usuario le quiere enviar a otro un archivo, el usuario que recibe suele desproteger su directorio de entrada por completo para que el otro usuario pueda copiar el archivo sin problema. Si este llegara a olvidar por descuido quitar los permiso, esta cuenta se quedara en total disposición para el intruso. Para que eso no pase, existen varias soluciones. Entre ellas enviar archivos por correo, codificarlos. Otra puede ser activar permiso de accesos en el directorio de entrada tanto para el grupo como para el resto de los usuarios (permisos 711), de entrada que cualquier usuario pueda acceder a un archivo del que sepa su nombre y tenga el permiso de lectura activado.

4.3 DIRECTORIOS CONFLICTIVOS Y OCULTOS

Directorios conflictivos

Por lo regular los populares piratas implantan archivos una vez que han logrado su objetivo "entrar al sistema" esta creación de archivos les ayudan a entrar como usuarios, y así puedan colocar diversos tipos de trampas por el sistema. Sea cual sea el tipo de mecanismo empleado por el pirata, necesitara almacenar en algún lugar los archivos y datos que vaya obteniendo como resultado al momento en que ejecutan sus trampas (ejemplo: shells con el bit SUD activado). Se podría pensar que el lugar más idóneo para ellos es el directorio /tmp, y de hecho es uno de los más frecuentes, ya que es un directorio que existe en todos los sistemas UNIX y todo usuario puede escribir en él. Pero el hecho de ser un lugar tan utilizado, puede que no sea un lugar atrayente para el pirata y por ello busque terreno diferente.

Una opción al directorio /tmp puede ser cualquier directorio con el bit sticky activado o que tenga permiso de escritura para todos los usuarios del sistema. Visto esto, parece necesaria la detección y comprobación de todos los directorios públicos del sistema, ya que cualquiera de ellos podría contener algún programa que pudiera dar sospecha alguna la seguridad del sistema. Para encontrar los directorios del sistema en los cuales cualquier usuario puede escribir, lo más difícil es utilizar una orden como la siguiente:

```
find  <directorio> <condición>
find / -type   d   -perm   -000003
```

Directorios ocultos

Una de las opciones de mayor uso por los piratas informáticos para almacenar todo tipo de archivos y datos, consiste en la creación de directorios ocultos. Como ya sabemos, cualquier archivo que empiece por el carácter '.' Es ocultado por el sistema de archivos, a no ser que lo indiquemos explícitamente mediante un ls –a. Esta característica es comúnmente utilizada por los piratas para llevar a cabo su contenido, ya que evita que cualquier persona pueda descubrir el directorio a simple vista, a no ser que indique explícitamente que desea obtener un listado en el que se incluyan los archivos ocultos.

De entre todos los nombres de directorios posibles, uno de los más utilizados por los piratas es el...; ya que incluso haciendo un ls –a puede confundirse con los directorios . y .., lo que hace aún más complicada su detección. Para detectar todos los directorios ocultos de los sistemas, podemos ejecutar una orden como la siguiente:

fnd / -name ".*" -type d

Un vez obtenida la lista de directorios ocultos, lo más conveniente sería que el administrador del sistema comprobarse el contenido de dichos directorios por si contuviera algún programa conflictivo.

4.4 OTROS ATAQUES SOBRE EL SISTEMA DE ARCHIVOS

La mayoría de los sistemas que sufren ataques de este tipo, son atacados por algunos de los siguientes puntos:

- Contraseñas fáciles de adivinar. Éste es el elemento que más ataques sufre, ya que la mayoría de los usuarios eligen contraseñas fáciles de recordar, pero esto significa que también son fáciles de adivinar.

- Archivo único de contraseñas (/etc/passwd). Todos los usuarios tienen acceso de lectura a este archivo, con lo cual una vez se tiene un usuario cualquiera que permita acceso al Sistema, aunque sea el más restringido de todos, tiene la posibilidad de ejecutar un "algoritmo trampilla" que detecta cadenas de caracteres que dan como resultado la misma cadena encriptada que existe en el archivo /etc/shadow. Una vez logrado esto, ya se puede usar ese usuario con esa nueva clave.

- Existe una amplia gama de servicios de red que pueden ser explotados por un cracker. Cualquier tarea que implique mandar o recibir información a través de la red, es susceptible de ser manipulada por un "cracker". por ejemplo : El correo electrónico o el servidor de páginas web.

- NFS, aunque es una forma cómoda de compartir archivos en una red de área local, plantea riesgos de seguridad.

Por lo que debemos tomar ciertas medidas de seguridad sobre todo el sistema de archivos.

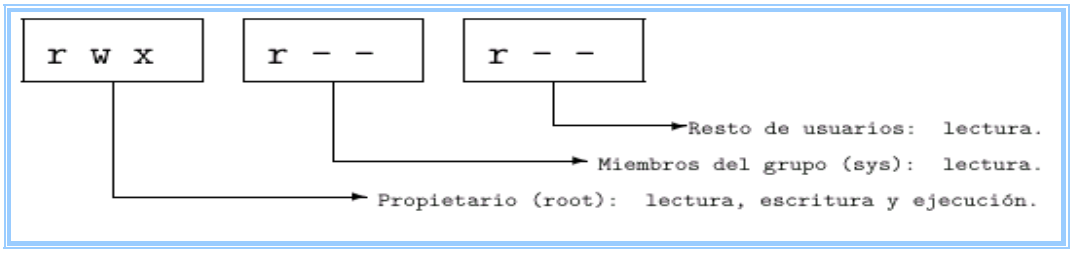
En Unix, todos los dispositivos del sistema son archivos o archivos, desde la memoria física hasta una unidad Zip, pasando por el módem, ratón, teclado o terminales. Este enfoque es uno de los más potentes, pero también es uno de los más peligrosos, ya que un error en un permiso podría permitir a un atacante borrar todo el disco duro de una máquina o leer los datos que alguien escribe en un terminal. Por esto, una correcta utilización de los permisos, atributos y otros controles sobre los archivos es vital para la seguridad de un sistema.

En un sistema Unix existen tres tipos básicos de archivos: **archivos planos**, **directorios**, y **archivos especiales** (dispositivos); generalmente, al hablar de archivos nos solemos referir a todos ellos si no se especifica lo contrario.

Los **archivos planos** son secuencias de bytes que a priori no poseen ni estructura interna ni contenido significante para el sistema: su significado depende de las aplicaciones que interpretan su contenido. Los **directorios** son archivos cuyo contenido son otros archivos de cualquier tipo (planos, más directorios, o archivos especiales), y los **archivos especiales** son archivos que representan dispositivos del sistema.

Cada sistema Unix tiene su sistema de archivos nativo(interfaz única de almacenamiento, que es la parte más visible del núcleo), por lo que para acceder a todos ellos de la misma forma el núcleo de Unix incorpora una capa superior denominada VFS (Virtual File System) encargada de proporcionar un acceso uniforme a diferentes tipos de sistema de archivos.

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



-rwxr--r-- root sys 2689 Dec 1 1998 /sbin/rc0

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

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

Listas de control de acceso: ACLs

Las listas de control de acceso (ACLs, Access Control Lists) proveen de un nivel adicional de seguridad a los archivos extendiendo el clásico esquema de permisos en Unix: las ACLs van a permitir asignar permisos a usuarios o grupos concretos; por ejemplo, se pueden otorgar ciertos permisos a dos usuarios sobre unos archivos sin necesidad de incluirlos en el mismo grupo. Este mecanismo está disponible en la mayoría de UNIX (Solaris, AIX, HP-UX, etc.), mientras que en otros que no lo proporcionan por defecto, como Linux, puede instalarse como un software adicional.

El concepto de ACL permite un control más preciso que los permisos del archivo por sí solos, al dar al propietario de un objeto la capacidad de conceder o denegar accesos con un nivel de precisión de usuario por usuario.

Las ACLs de Unix también permiten especificar los derechos de acceso para los miembros de grupos definidos en el sistema en el archivo administrativo /etc/group. El administrador del

sistema puede determinar el número máximo de entradas por ACL, configurando un parámetro regulable.

Almacenamiento seguro

- La orden crypt(1): permite cifrar y descifrar archivos en diferentes sistemas Unix;
- PGP (Pretty Good Privacy): El software PGP, permite el cifrado de archivos de forma convencional mediante criptografía simétrica; convirtiendo a este programa en una excelente herramienta para cifrar archivos que almacenamos en nuestro sistema;
- TCFS (Transparent Cryptographic File System): es un software que proporciona una solución al problema de la privacidad en sistemas de archivos distribuidos. TCFS almacena los archivos cifrados, y son pasados a texto claro antes de ser leídos; todo el proceso se realiza en la máquina cliente, por lo que las claves nunca son enviadas a través de la red.

Otros métodos de almacenamiento seguro

Sistema CFS (Cryptographic File System): Provee de servicios de cifrado a cualquier sistema de archivos habitual en Unix.

SFS (Secure File System). El SFS de Unix funciona de una forma similar a CFS pero utilizando el criptosistema Blowfish.

Criptografía

La criptografía es la herramienta principal utilizada en la mayoría de los sistemas de almacenamiento seguro; sin embargo, todos ellos plantean un grave problema: toda su seguridad reside en la clave de cifrado, de forma que el usuario se encuentra indefenso ante métodos legales - o ilegales - que le puedan obligar a develar esta clave una vez que se ha determinado la presencia de información cifrada en un dispositivo de almacenamiento.

PROGRAMAS SEGUROS, INSEGUROS Y NOCIVOS

El problema para la seguridad viene cuando es el propio administrador quien utiliza programas contaminados por cualquier clase de virus, y para evitar esto hay una medida de protección básica: la prevención.

Errores en los programas

Los errores o bugs a la hora de programar código de aplicaciones o del propio núcleo de Unix constituyen una de las amenazas a la seguridad informática. En la mayoría de situaciones no se trata de desconocimiento a la hora de realizar programas seguros, sino del hecho que es prácticamente imposible no equivocarse en miles de líneas de código.

Buffer overflows: Seguramente uno de los errores más comunes, y sin duda el más conocido y utilizado es el stack smashing o desbordamiento de pila, también conocido por buffer overflow.

Condiciones de carrera: Otro error muy conocido en el mundo de los sistemas operativos son las condiciones de carrera, situaciones en las que dos o más procesos leen o escriben en un área compartida y el resultado final depende de los instantes de ejecución de cada uno. Cuando una situación de este tipo se produce y acciones que deberían ser automáticas no lo son, existe un intervalo de tiempo durante el que un atacante puede obtener privilegios, leer y escribir archivos protegidos, y en definitiva violar las políticas de seguridad del sistema.

Virus y otras amenazas

No todas las amenazas lógicas provienen de simples errores: ciertos programas, denominados en su conjunto software malicioso, son creados con la intención principal de atacar a la seguridad.

Podemos nombrar a los Virus, Gusanos, Conejos, Caballos de Trolla, Bombas lógicas, Puertas traseras, Applets hostiles, Superzapping y Programas salami entro otros.

Programación segura

Parece obvio que después de analizar los problemas que un código malicioso o simplemente mal diseñado puede causar, dediquemos un apartado a comentar brevemente algunos aspectos a tener en cuenta a la hora de crear programas seguros. Vamos a hablar de **programación en C**, por ser el lenguaje más utilizado en Unix:

Para la codificación segura de este tipo de programas, proporciona unas líneas básicas:

- Máximas restricciones a la hora de elegir el UID y el GID.
- Reset de los UIDs y GIDs efectivos antes de llamar a exec().
- Es necesario cerrar todos los descriptores de archivo, excepto los estrictamente necesarios, antes de llamar a exec().
- Hay que asegurarse de que chroot() realmente restringe.
- Comprobaciones del entorno en que se ejecutará el programa.
- No utilizar creat() para bloquear.
- Capturar todas las señales.
- Hay que asegurarse de que las verificaciones realmente verifican.
- Cuidado con las recuperaciones y detecciones de errores.
- Cuidado con las operaciones de entrada/salida.

4.4.1 ATAQUES SOBRE EL ESPACIO EN DISCO.

La mayoría de los sistemas UNIX utilizan espacio de disco para que los procesos puedan realizar intercambio de páginas entre memoria principal y secundaria. Si un pirata decide crear un programa que solicito continuamente memoria al sistema, tarde o temprano tendrá que realizar intercambios de páginas y llegara un momento en que se desborde tanto el espacio de disco utilizado para el intercambio de páginas como la memoria principal, dejando colapsado el sistema.

La solución para este problema consiste en establecer límites determinados sobre los recursos con los que pueden contar los procesos invocados desde el shell. Estos límites pueden establecerse con la orden interna del shell **ulimit.** Normalmente, la mayoría de estos límites están

fijados a valores ilimitados o muy grandes. En nuestro caso, el problema se podría solucionar colocando en el archivo **/etc/profile** las siguientes órdenes:

```
ulimit    -d 10240
ulimit    -Hd  10240
```

La primera línea establece el límite de memoria para datos en 10240 Kbs, mientras que la segunda lo establece de forma que ningún usuario lo pueda sobrepasar.

Esta solución no es del todo buena, ya que el límite lo impone el shell. Si el pirata mantiene abiertas tantas sesiones como sean necesarias hasta que el conjunto agote toda la memoria disponible, aparecerá el mismo problema, pero, eso sí, el pirata no tendrá las mismas facilidades para llevar a cabo la acción.

Como en el caso de los ataques sobre el número de nodos índice, en la que la estructura utilizada por el sistema de archivos para almacenar información sobre los archivos del sistema. Cada archivo del sistema tiene asociado un nodo índice. Con el programa df podemos conocer el número de nodo índice disponibles en los distintos sistemas de archivos.

```
df    -i
```

Si un intruso realiza un ataque similar al de la sección anterior, pero en vez de solicitar memoria al sistema solicita la creación de nuevos archivos, llegara un momento en que se consuman todos los nodos índice disponible en el disco, dejándolo inutilizable. La solución a este problema es el establecimiento de una política de cuotas en el sistema, de manera que ningún usuario pueda sobrepasar el límite impuesto por el administrador del sistema.

Otro ataque es llenar una partición del disco. Si un usuario llena el disco los demás no pueden crear archivos o desarrollar trabajo útil.

Ataques de llenado de disco.

Un disco sólo puede almacenar una cierta cantidad de información. Si el disco está lleno se debe borrar información antes de guardar algo más.

A veces los discos se llenan cuando un programa o usuario por error llegan a crear demasiados archivos o muy grandes. Otras veces los discos se llenan porque los usuarios lentamente aumentan el uso de los mismos.

La instrucción du permite vigilar los directorios del sistema que contienen la mayor parte de los datos, du busca recursivamente en un árbol de directorio y lista de cuantos bloques usa cada uno, un ejemplo para revisar toda la partición /usr se puede usar lo siguiente: una vez que se conoce la información pesada se es fácil hacer una limpieza.

```
# du  /usr
29      /usr/dict/papers
3875/usr/dict
8       /usr/pub
4032    /usr
…
#
```

Por lo que además se pueden listar y buscar los nombres de los archivos mas grandes usando **find** a igual que con la opción **size** para listar sólo los archivos que excedan un tamaño dado. Además **–xdev o –local** nos sirven para evitar buscar en los directorios montados NFS (por lo que debe ser esperado **find** en cada uno de los servidores de NFS). Se dice que este método es más rápido como usar **du** y puede ser mas útil cuando se trata de encontrar unos cuantos archivos grandes que ocupan mucho espacio, como se nos muestra en este ejemplo, en el que el archivo /usr/TeXdist.tar.Z es un probable a eliminar, especialmente si ya se ha descomprimido de TeX. Los archivos /var/spool/mqueue/syslog/var/spool/uucp/LOGFILE que también son los siguientes para ser borrados después de respaldarlos en una cinta de disco o otros disco.

Otro ataque sobre el espacio en disco es cuando el usuario crea demasiados procesos, si quiere paralizar el sistema al crear demasiados procesos:

```
main ( )
{
while (1)
fork( );
}
```

Cuando se ejecuta este programa, el proceso ejecuta la instrucción fork ( ) y crea un segundo proceso idéntico al primero. Ambos procesos ejecutan fork ( ), creando cuatro procesos. El crecimiento sigue hasta que el sistema ya no puede soportar más procesos. Éste es un ataque total, porque todos los nuevos están esperando que se establezcan mas procesos. Aun si se logra parar uno de los procesos, otros tomarían su lugar.

Este ataque no inhabilitaría la mayor parte de las versiones actuales de UNIX porque se elimina el número de procesos que pueden ejecutarse bajo cualquier UID (menos para root). Este límite, llamado MAXUPROC se configura comúnmente en el Kernel cuando se construye el sistema. Algunos sistemas UNIX permiten que se fije este valor al arrancar la computadora. Por ejemplo, Solaris permite colocar lo siguiente en el archivo /etc/sytem:

```
set maxuproc =100
```

Un usuario que emplee este ataque agotará su cuota de procesos, pero nada más. Entonces, el superusuario puede usar la instrucción ps para determinar los números de procesos de los procesos involucrados y usar la instrucción kill para pararlos. No se pueden parar uno por uno porque los demás simplemente crearán otros procesos. Un mejor enfoque es usar la instrucción kill para detener cada proceso, simplemente y después pararlos todos a la vez.

```
# kill –STOP 1009 1110 1921
# kill –STOP 3219 3220
.
.
.
# kill –kill 1009 1110 1921 3219 3220…
```

Puesto que cada proceso detenido se carga a la cuota NPROC del usuario, el programa que se bifurca no podrá engendrar más procesos. Después hay que visualizar al usuario.

Otra alternativa es para todos los proceso en un grupo de procesos simultáneamente. En muchos casos en los cuales un usuario crea demasiados procesos, todos los procesos estarán en

el mismo grupo. Para averiguar cuál es el grupo de procesos, hay que ejecutar la instrucción ps con la opción –j. se identifica el grupo y se paran los procesos de golpe:

```
# kill -9 -1009
```

Es posible que el sistema se haya configurado incorrectamente. El límite de procesos por usuario puede ser igual o mayor que el límite de todos los procesos en el sistema. En ese caso un solo usuario puede inundar todo el sistema.

Si se recibe el mensaje de error "no more processes", entonces hay demasiados procesos ejecutándose en el sistema. El sistema no permite que se creen más procesos.

```
% ps –efj
No more proceses
%
```

Si se agotan los procesos hay que esperar un momento y volver a intentar. La situación puede ser transitoria. Si el problema no se corrige solo se tiene una situación interesante.

Puede ser muy difícil corregir el exceso de procesos sin reiniciar la computadora. Por dos razones.

- No se puede ejecutar la instrucción ps para determinar el número del proceso que hay que parar.
- Si en ese momento no se es el superusuario no se pueden emplear las instrucciones su o login porque estas instrucciones requieren la creación de un nuevo proceso.

Se puede evitar el segundo problema mediante la instrucción integrada del intérprete de comandos exec' para ejecutar la instrucción su sin crear un nuevo proceso.

```
% exec  /bin/su
password:  foobar
#
```

La función exec del interprete de comandos hace que un programa se ejecute (mediante la llamada del sistema exec ()) sin que se use antes la llamada del sistema fork ( ). Lo que ve el usuario es que el intérprete ejecuta el programa y luego se para.

Se puede tener problemas de exceso de procesos que saturan un sistema puede ser necesario reiniciar la computadora. Puede parecer lo más sencillo apagar y encender la computadora, pero es posible que esto dañe algunos bloques del disco, porque probablemente no se descargue la memoria temporal activa a disco. Pocos sistemas se han diseñado para pararse ordenadamente si se les corta la corriente. Mejor usar la instrucción kill para el proceso problemático o llevar el sistema al modo de un solo usuario.

En la mayor parte el superusuario puede enviar una señal SIGTERM a todos los procesos excepto los procesos del sistema y al proceso que se está escribiendo.

```
# kill  -TERM -1
#
```

Si el sistema UNIX que se usa no tiene esta característica se puede ejecutar la instrucción.

```
# kill  -TERM 1
#
```

Para enviar un SIGTERM al procesos init. UNIX automáticamente para todos los procesos y entra al modo de un solo usuario cuando se para int. Después de puede ejecutar la instrucción kill, hay que usar exec para disparar una versión de ksh o csh; estos comandos tienen integrada la instrucción kill y por lo tanto no tienen que crear un proceso nuevo para ejecutar la instrucción.

4.4.2 SIN SUFICIENTE CAPACIDAD EN EL DISCO.

Para ello nos basamos en la orden **df** (disk free) la cual nos permite ver la cantidad de espacio en disco

Sintaxis: df [i] [sistema de archivos]

Descripción de la orden **df** nos muestra, sin especificar el sistema de archivos. Información sobre todos los sistemas de archivos. Los campos mostrados se refieren al nombre del archivo de dispositivo tipo bloque, número total de Kilobytes de espacio en disco que ocupa el sistema de

archivos, número de Kilobytes ocupados, número de kilobytes disponibles, porcentaje de espacio en disco utilizado por los archivos y lugar donde está montado el sistema de archivos.

Para ver como esta compartido el espacio en disco entre los directorios utilizaremos la ordne **du** (disk usage).

Sintaxis : du [-s] [directorio (s)]

La orden **du** nos informa del espacio en bloques que ocupa el directorio (s) que les hemos dado como argumento y todos los archivos y subdirectorios que cuelgan de él. Con la opción –s sólo informará del número de bloques que ocupa el directorio, sin ver cómo se divide esta cantidad entre sus archivos y subdirectorios.

Ejemplo:
```
# du -s /bin
4154    //bin
#
```
Ejemplo de
```
$ du
 15    ./user-alpha-4/emacs-chapter-stuff
 157   ./user-alpha-4/ps-files
 2600  ./user-alpha-4
 2634  .
```

En este caso, el directorio ./user-alpha-4/emacs-chapter-stuff ocupa 15 kilobytes, ./user-alpha-4/ps-files 157 Kb, ./user-alpha-4 2600 Kb y todo el directorio actual junto con sus subdirectorios ocupa 2634. La salida de du muestra los nombres de los subdirectorios con el prefijo ./ para indicar que es a partir del directorio actual.

Podemos pedir también el espacio ocupado por un directorio por su nombre absoluto:

```
$ du /etc
349   /etc
```

La salida de df es un poco más críptica:

```
$ df
```

filesystem	1024-blocks	Used	Available	Capacity	Mounted on
/dev/hda1	237133	208827	16450	93%	/
/dev/hdb	208879	194597	3839	98%	/home

La primera columna nos indica el *filesystem* o sistema de archivos, o dispositivo físico, o en términos más comprensibles, el disco. La segunda nos dice el espacio total en el disco en unidades de 1024 bytes, o 1 kilobyte, aunque las unidades varían de una implementación de Unix a otra. La tercera y cuarta columnas indican el espacio utilizado y el espacio disponible en las mismas unidades que la segunda. La quinta columna indica el porcentaje ocupado del disco. Por último, la sexta columna indica en que parte del sistema de archivos está *montado* el disco, así que si deseamos saber cuanto espacio queda disponible para el usuario Pablo, debemos considerar en que lugar del sistema de archivos está el directorio donde trabaja. Como está en /home/pablo en éste caso, sabremos que el espacio que le queda disponible, son 3 839 Kb o 3.7490 Mb, dado que pablo es un subdirectorio dentro de home.

Cabe hacer la aclaración que UNIX provee de mecanismos para acotar el espacio en disco disponible para cada usuario, pero la mayoría de las veces esta restricción no se aplica, con lo cual si hay cinco usuarios en el sistema, estos cinco usuarios compiten por el espacio disponible en el disco donde están montados sus directorios.

La elección de restringir el espacio en disco a cada usuario está determinada por las políticas de uso que emplee el administrador del sistema.

Usualmente los intrusos ocupan todo el espacio disponible en las particiones de los discos del sistema. el programa **df** muestra información sobre el estado de los sistemas de archivos. Este programa nos puede servir para detectar sistemas de archivos que estén al límite de su capacidad. Si es invocado con la opción **–k** como en el siguiente ejemplo, mostrara la diferencia asumiendo bloques de 1 **kb**: **(26).**

df -k

Si queremos un mayor detalle, podemos utilizar el programa **du** que recorres el árbol de directorios mostrando el espacio ocupado por cada uno de ellos. Al igual que en el ejemplo anterior, la opción –k muestra la información asumiendo bloques de 1 Kb: **du -k**

Al igual que con los nodos índice, la solución para evitar que un usuario acapare todo el espacio

(26). Libro Seguridad en UNIX, Manuel Mediavilla, Edit. Ra-ma pagina 89, 90

4.4.3 ENCONTRAR MODIFICACIONES EN LOS ARCHIVOS DEL SISTEMA.

Cuando se sufre de invasión de intrusos, es muy probable que estos modifiquen, creen determinados archivos para facilitar su entrada en el sistema en una próxima ocasión. Una práctica bastante extendida consiste en modificar el comportamiento de diversos programas del sistema, como ls o ps, de forma que no generen información que pueda delatar su presencia. Otra común consiste en crear shell scripts con el bit SUID activado y ocultarlos en los archivos del sistema.

Del sistema. Aunque puede ser efectiva se corre el riesgo que el administrador se de cuenta, pero de todas formas es difícil detectar la modificación de los propios programas del sistema. Un intruso podría modificar el programa login de forma que introduciendo un nombre de usuario determinado pudiera acceder al sistema como root. Otra podría ser modificar el programa passwd para que todas las contraseñas de los usuarios quedaran almacenadas en un archivos o fueran enviadas por correo a un dirección determinada.

Informándonos nos encontramos que hay manera de enterarnos en cuanto a detectar modificaciones en los archivos es mantener una base de datos con los datos de los archivos e información adicional como *checksum* o un código de redundancia cíclico (CRC), comparándola periódicamente con los datos de los archivos almacenados actualmente en el sistema de archivos del sistema, es muy probable que un pirata haya modificado algún archivo.

Una vez creada la base de datos, tendremos información suficiente para detectar cualquier modificación que se pudiera producir sobre los archivos de los directorios especificados. Para comprobar si se ha producido algún tipo de modificación sobre cualquiera de los archivos, tan sólo tendremos que volver a ejecutar el programa. Si se ha modificado, añadió o borrado algún archivo, el programa mostrará un mensaje indicándoselo.

Si tuviéramos necesidad de realizar cualquier modificación sobre alguno de los archivos incluidos necesidad de realizar cualquier modificación sobre alguno de los archivos incluidos en los directorios sobre los que se ha construido la base de datos, tan sólo tendremos que volver a ejecutar el programa con el parámetro –f para que vuelva a generar la base de datos con los archivos actualizados. Antes de realizar esta actualización, deberíamos realizar una última comprobación sobre los archivos ejecutando el programa y comprobar que todo es correcto, ya

que de no hacerlo así puede que hagamos la actualización y generemos una base de datos que contenga archivos actualizados sin autorización.

4.4.4 PROBLEMAS CON DIRECTORIOS TEMPORALES

En la mayor parte de los sistemas en UNIX se configura que cualquier usuario pueda crear archivos de cualquier tamaño en el directorio /tmp. Normalmente no se habilita una cuenta en el directorio/tmp. Por consiguiente un solo usuario puede llenar la partición en la que el directorio /tmp esta montado lo cual imposibilitara a otros usuarios (y probablemente al superusuario), para crear archivos nuevos.

Desafortunadamente se requiere de poder colocar archivos en el directorio /tmp para funcionar correctamente. Por ejemplo vi o mail almacenan archivos temporales en /tmp. Estos programas fallan inesperadamente si no pueden crear archivos temporales. Muchos **guiones o scripts** escritos por los administradores de sistemas dependen de la posibilidad de crear archivos en el directorio /tmp y no verificar si hay espacio disponible.

Este tipo de problemas en el directorio /rmp son casi siempre accidentales. Un usuario transferirá muchos archivos grandes a ese directorio y luego se le olvidara borrarlos.

Hay maneras de prevenir este ataque al /tmp.

- Habilitando cuotas en /tmp para que ningún usuario pueda llenarlo. Una buena cuota es permitir que cada usuario llene el 40% del espacio disponible en /tmp. De este modo /tmp sólo se saturara si dos usuarios se ponen de acuerdo.
- Contar con algún proceso que vigile en directorio /tmp periódicamente y borrar los archivos que tengan una antigüedad mayor a tres o cinco días.

Siendo el superusuario es posible vigilar el directorio /tmp normalmente y borrar los archivos que tengan una antigüedad mayor a tres o cinco días.

```
# find /tmp –mtime +5 –print |  xards rm –rf
```

Glosario

- **¿Qué es un buffer?:** Es una fuente de almacenamiento temporal que reside en el propio dispositivo ya sea de entrada, o de salida.

- **Archivo.** Grupo de datos relacionados, organizado en una unidad y almacenado en el sistema de archivos de UNIX bajo un nombre de archivo. UNIX tiene cuatro tipos de archivo:

Archivos de dispositivos de bloque (dispositivos del tipo de las unidades de disco).
Archivos de dispositivos de carácter (dispositivos del tipo de las terminales).
Archivos FIFO (archivos primero en entrar, primero en salir o canalizaciones).
Archivos normales (es el tipo más comúnmente visto y usado).

- **Canalización.** Una manera de emplear la salida de un comando como entrada directa para otro comando. En la línea de comandos se simboliza mediante una barra vertical (|), como en el comando en el comando who | sort.
- **Código fuentes.** Instrucciones un lenguaje de programación que pueden convertirse en un programa.
- **Consola.** Terminal que forma parte de todos los sistemas UNIX, la cual es única porque en ella se presentan los mensajes de error. No tiene diferencias físicas con respecto a las demás terminales; en los sistemas multiterminales tradicionales, la consola sencillamente es aquella terminal conectada a un puerto especial.
- **Controlador de dispositivo.** Módulo de software dentro del Kernel de UNIX que controla la operación de un tipo especifico de dispositivo, como unidad de disco, terminal o unidad de cinta. Cada tipo de hardware del sistema cuenta con un controlador como lectura y escritura de datos.
- **Directorio actual.** También conocido como directorio de trabajo. Este directorio es en donde se localizan las actividades del usuario, y es el directorio del cual parten todos los nombres de ruta relativa. Usted debe ejecutar un comando cd para indicar cuál será el directorio actual.
- **Directorio.** Archivo que comprende una tabla de contenido de otros archivos. Desde la perspectiva del usuario, los directorios sirven como espacio en disco en el que puede almacenar archivos.

- **Ejecución en primer plano.** Modo de operación en el que un programa acepta entradas desde el teclado y despliega datos en la pantalla. De manera predeterminada, los programas UNIX se ejecutan en primer plano. También se dice que los programas que se ejecutan en primer plano son interactivos.
- **Ejecución en segundo plano.** Ejecución de un programa UNIX que no interactúa con la pantalla ni con el teclado. La ejecución en segundo plano sirve para efectuar tareas que requieran un tiempo considerable pero que no necesitan interacción con el usuario, por ejemplo, el ordenamiento de un archivo o el dar formato a un archivo para impresión.
- **En paralelo.** Operaciones ejecutadas de manera simultánea. Las computadoras con programas en paralelo los ejecutan al mismo tiempo; las computadoras con procesadores paralelos tienen dos o más procesadores que operan en conjunto. El puerto paralelo de una computadora, envía más de una unidad de información de manera simultánea en comparación con los puertos seriales, que deben enviar sólo un bit de información cada vez.
- **Exportar.** Hacer que un recurso (digamos un directorio) quede disponible para otros sistemas de la red. Algunos sistemas de conectividad de red utilizan la palabra programar para indicar que, aunque disponible, el recurso no será usado necesariamente.
- **Kernel.** Núcleo de UNIX en el cual se coordinan y controlan las actividades de la computadora. El Kernel permanece en la memoria de la máquina mientras la computadora esté en operación.
- **Multitareas.** Capacidad de ejecutar varias tareas simultáneas para cada usuario.
- **Multiusuario.** Sistema operativo que permite que varias personas utilicen al mismo tiempo la computadora,.
- **Nodo I.** Significa nodo de información. Área de un disco que contiene información importante acerca de un archivo, incluyendo el modo del mismo, su cantidad de vínculos, la información de identificación del usuario propietario, el tamaño del archivo, las mascaras de fecha y hora, e información que indica en que lugar del disco se localizan los datos asociados con el archivo o directorio.
- **Programar.** Sinónimo de exportar.
- **Red.** Grupo de computadoras vinculadas que pueden compartir datos. Los vínculos entre las computadoras pueden ser temporales e iniciarse a solicitud (por ejemplo, una red UUCP), o ser permanentes (por ejemplo, una red Ethernet).

- **Shell.** Programa que vincula las partes de UNIX (el Kernel) con el usuario, proporcionando así la interfaz entre ellos. Usted emite comandos, los cuales son traducidos por el shell en llamadas de sistema para el Kernel; la información devuelta por este último es presentada en la pantalla por shell se usa para indicar una interfaz de línea de comandos, aunque técnicamente el escritorio de UNIX Ware es un shell GUI.

A continuación se muestra una lista con los comandos más habituales en UNIX, así como una breve descripción de dicho comando. Para ampliar la información de cualquiera de ellos se recomienda acudir a la ayuda de sistema (*man comando*) o en cualquier *Reference Manual* de UNIX (Manual de Referencia) **(27).**

Comandos comunes del sistema de archivos

at	Ejecuta una tarea a la hora programada.
banner	Muestra un banner con los argumentos pasados.
bs	Calculadora.
cal	Muestra un calendario del mes o año indicado.
cancel	Cancela trabajos enviados a la impresora.
cat	Muestra el contenido de un archivo.
cat filenames	imprimir contenidos de los archivos nombrados
cc	Compilador de C bajo UNIX.
cd	Cambia de directorio.
chgrp	Cambia el grupo de un archivo o directorio.
chmod	Cambia los permisos de acceso de un archivo o directorio.
chown	Cambia el propietario de un archivo o directorio.
clear	Limpia la pantalla.
cmp	Compara archivos.
cmp	Compara dos archivos.
cmp file1 file2	imprimir localización de la primera diferencia
compress, pack	Comprime un archivo.
cp	Copia archivos.
cpio	Envía o recupera datos de un archivo cpio (copias de seguridad, etc.).
crontab	Programa en el cron de usuario las tareas especificadas.
cut	Corta los campos especificados de un archivo.

(27). Introducción UNIX , George Megahabghab, Editorial-Prentice –Hall, Hispano Americana, Pàgina 15, 16

date Muestra la fecha y hora del sistema. Solo root la puede modificar.

df, bdf Muestra información sobre la ocupación de sistemas de archivos.

diff Compara archivos y directorios.

diff file1 file2 imprimir todas las diferencias entre archivos

disable Desactiva la impresora, impidiendo la impresión de archivos.

du Muestra la ocupación de un directorio o conjunto de directorios.

echo Muestra por pantalla lo que se indica como argumento.

ed Editor de archivos por líneas.

ed filenames editar archivo nombrado

emacs Editor de textos más potente que vi.

enable Activa la impresora, permite la impresión de archivos.

env Muestra las variables de entorno del usuario.

exit Sale del sistema.

export Exporta el valor de la variable de entorno que se especifique.

file Devuelve el tipo de archivo que es un archivo determinado.

find Busca archivos con unas condiciones determinadas.

finger Devuelve datos sobre la actividad de los usuarios conectados a un sistema.

ftp Permite transferir archivos a o desde otros sistemas.

grep pattern filenames imprimir líneas que coincidan con pattern

grep –v pattern files imprimir líneas que no coincidan con pattern

grep, fgrep, egrep Busca una cadena de caracteres dentro de un archivo.

groups Enumera los grupos a los que pertenece un usuario.

head Muestra las primeras líneas de un archivo.

hostname Devuelve o fija el nombre de la máquina.

kill Finaliza la ejecución de un proceso.

ksh, sh, csh Invoca una subshell dentro de la shell actual.

ln Crea un enlace con uno o varios archivos.

lp Envía el contenido de un archivo al spooler de impresión.

lpstat Muestra el estado de las impresoras conectadas al sistema.

ls listar los nombres de todos los archivos en el directorio actual

ls – r listar en orden inverso; también –rt; -rlt; etc

ls –1 lista larga: más información; también ls –lt

ls filenames listar solo los archivos nombrados

ls –t listar por orden de tiempo, primero el más reciente

ls –u listar por tiempo el último en usarse; también ls –lu, ls –lut

mail, mailx Visualiza y envía correo a otros usuarios.

man Muestra en pantalla la ayuda existente respecto al argumento indicado.

mesg Controla el acceso a la pantalla por parte de otros usuarios.

mkdir Crea un directorio.

more Visualiza el contenido de un archivo por pantallas.

mount Monta un sistema de archivos (lo hace accesible a los usuarios).

mv Mueve archivos de un directorio a otro o los renombra.

mv file1 file2 mover file1 en file2, sustituir el file2 anterior si existe

newgroup Cambia el grupo actual del usuario, en caso de tener varios.

news Muestra las noticias existentes en el sistema.

nice Permite reducir la prioridad de un proceso. Solo root puede incrementarla.

nohup Realiza tareas en background y envía los resultados al archivo nohup.out.

passwd Permite modificar la clave o password de un usuario.

paste Junta archivos horizontalmente.

pg Visualiza el contenido de un archivo por pantallas.

pr Formatea un archivo para su posterior impresión.

pr – m filenames imprimir lado a lado archivos nombrados (columnas múltiples)

pr – n filenames imprimir en n columnas

pr filenames imprimir contenidos con encabezado, 66 lineas por página

ps Muestra los procesos activos en el sistema.

pwd Muestra el directorio actual.

rcp Copia archivos desde o hacia otro sistema remoto.

read	Lee la entrada que el usuario haga desde el teclado.
rlogin	Permite conectarse a otro sistema.
rm	Borra un archivo.
rm filenames	suprimir irrevocablemente los archivos nombrados
rmdir	Borra un directorio.
sar	Visualiza la ocupación de los procesadores del sistema.
sdiff	Compara archivos ASCII, listando las diferencias por columnas.
set	Muestra y pone valores a las variables de entorno del usuario.
sort	Ordena un archivo.
sort filenames	clasificar alfabéticamente los archivos por renglón
split	Divide un archivo en trozos más pequeños.
stty	Establece o visualiza algunas opciones de terminal.
su	Permite cambiar la identidad de un usuario a otro.
sync	Salva todos los buffers pendientes a disco.
tail	Muestra las últimas líneas de un archivo.
tail +n filename	comenzar a imprimir el archivo en el renglón n
tail filename	imprimir los últimos 10 renglones de archivo
tail –n filename	imprimir los últimos n renglones del archivo
talk	Establece una conexión con otro usuario para mantener una comunicación.
tar	Envía o recupera datos de un archivo tar (copias de seguridad, etc.).
tee	Dirige la entrada estándar a un archivo y al terminal a la vez.
telnet	Permite conectarse a otro sistema.
test	Prueba una condición devolviendo true o false.
time	Muestra en pantalla el tiempo que ha sido necesario para ejecutar un comando.
touch	Modifica la fecha y hora de un archivo. Si no existe lo crea con tamaño 0.
tr	Filtro que sirve para intercambiar un carácter por otro.
umask	Visualiza y establece permisos para los archivos y directorios que se generen.
umount	Desmonta un sistema de archivos.
uname	Devuelve información sobre el nombre de la máquina, versión S.O., etc.

uncompress, unpack	Descomprime un archivo comprimido.
users	Devuelve los usuarios conectados al sistema.
vi	Permite editar archivos.
wait	Espera la finalización de tareas en background.
wall	Envía un mensaje a todos los usuarios conectados al sistema.
wc	Cuenta las palabras, líneas y caracteres de un archivo.
wc filenames	contar renglones, palabras y caracteres en cada archivo
wc –1 filenames	contar renglones de cada archivo
whence	Busca un archivo dentro de los directorios definidos en la variable PATH.
which	Localiza un archivo dentro de los directorios incluidos en $PATH y aliases.
who	Muestra los usuarios que están conectados al sistema.
write	Envía un mensaje a la pantalla del usuario especificado.

CONCLUSIONES.

Primera. Podemos decir que los sistemas operativos son de suma importancia ya que estos manejan, administran totalmente nuestra máquina y los recursos que tenemos, por lo que mantener bien asegurado bien nuestro sistema operativo tanto interno, como externo, ya hemos visto como han venido cambiando estos sistemas, y muchas de la funciones que manejan por que sigo manifestando que la seguridad es un factor importante si es que queremos que nuestro sistema tenga larga vida y no nos provoque problemas.

Segunda. La seguridad que tenemos dentro de UNIX es mas que eso, ya que este sistema controla y administra desde el momento en que hacemos uso de firmarnos con nuestra cuenta de usuario, dejándonos entrar siempre y cuando sea la contraseña correcta, y de ser lo contrario bloquearla para que ningún intrusos trate de adivinar hasta encontrar la contraseña y hacer mal uso de ella, por lo que en el solo momento de entrar a la máquina de trabajo la seguridad, no depende solo del sistema operativo, si no también de la responsabilidad que tenga cada usuario para poder hacer uso de su cuenta y contraseña, su deber cambiar su password, así como buscar la mas difícil de adivinar, pero fácil de recordar para el, no divulgar.

Tercera. Los administradores con frecuencia deben de revisar, restringir los accesos a ciertos sitios, archivos, directorios, comandos que no sean necesarios para el uso de cada usuario, por lo que del administrador depende el no dejarlos entrar a menos que en casos especiales sea necesario, todo para tener una mejor seguridad dentro del sistema, y así evitar grandes problemas.

Cuarta. Día con día los diferente proveedores se encargan de buscar, modificar nuevas versiones de UNIX, para hacerlo mas seguro en ocasiones puede tener sus ventajas y por el otro dejar problemas un tanto laborioso, para aquellos que ya tienen una manera de trabajar, todo en cuanto a los comandos, permisos, comprensión al hacer uso de el sistema. Pero estas mejoras no son en vano ya que su principal objetivo de ellas, son ofrecer un mejor soporte y control entre distintas plataformas.

Quinta. La importancia seguridad, no solo se manifiesta dentro de un sistema de información mas bien es general, se ve reflejada e importante en la vida cotidiana, por lo que no es para menos, dejar de valorar que tenemos muchos medios por los cuales nos vemos afectados, y que de alguna manera es bueno mantener ese rango estricto, sobre todo, si se tienen intrusos queriendo afectarnos dentro del sistema, cosas que no solo nos afectaran a nosotros como usuarios.

Sexta. Como conclusión. Para mi la Seguridad Informática es un aspecto que muchas veces descuidamos en nuestros sistemas, pero de vital importancia para el correcto funcionamiento de todos ellos. Seria importante hacer mención en los siguientes conceptos:

- Todo sistema es susceptible de ser atacado, por lo que conviene prevenir esos ataques.
- Conocer las técnicas de ataque ayuda a defenderse mas eficientemente.
- La seguridad basada en la ocultación no existe.

Séptima. Como todas las cosas siempre se corren riesgo, y mas a un desde el momento en que nos vemos conectados a través de la red, por lo que estos aumentan mas a la hora en que comenzamos a hacer intercambio de información, además hay que tomar en cuenta que entre más gente haya en un sistema de red, es más grande el riesgo de encontrar a intrusos.

Octava. Existen varias medidas de seguridad establecidas por los administradores, y en otros las políticas establecidas por el departamento, por lo que las políticas no se pueden dejar de tomarse en cuenta para que la seguridad sea respetada y tomada en cuenta como un factor muy importante al igual de hacerla divulgada, estricta, para una mejor eficacia en todo.

Novena. La seguridad que nos proporciona el sistema UNIX comienza desde tener el control de acceso a los archivos, la protección del uso de los recursos, entre otros. Por lo que la contraseña de inicio juega un papel sumamente importante, y es por que ello que debemos estar consientes al ahora de crearlas tomando en cuenta las recomendaciones que nos indican los expertos, a la hora de hacer un cambio, todo para mantenernos a salvo entre comilla, ya que los intrusos buscan diferentes maneras de entrar, realizar robos, modificaciones alterar información.

Décima. Existen diversas fuentes donde el usuario puede ser uso de estas y sacar una contraseña, pero muchas de las ocasiones solo lo toman poco interés, por lo que al momento de hacer una nueva elección de contraseña y cambiarla, estos buscan una contraseña fácil e incluso para no hacerse la vida mas complicada de estar memorizando una por mes, dependiendo el tiempo que les indique el administrador, las dejas anotadas al posible alcance de personas que en cambio no ven solo una palabra si no la cantidad de cosas que pueden hacer con ella, es por eso que se debe de comenzar a educar e informar al usuario que son importantes.

Décima primera. Dentro la organización del sistema UNIX nos encontramos con que este sistema funciona con archivos, directorios, programas, datos, controladores, por lo que cada uno de estos deben estar bien controlados, y mas si son requeridos para hacer cambios, por lo que es importante tener bien administrado y ubicado, para la hora en que se seden permisos.

Décima segunda. Hay varias órdenes utilizadas por el administrador para hacer más accesible la información, como la orden mount que muestra los sistemas de archivos montados en ese instante, como bien sabemos los archivos quedan ubicados por sus respectivos nombres ya que cada uno de ellos debe tener un nombre el cual lo identifique, a demás que debe corresponder a alguna ruta.

Décima tercera. En cierta manera es bueno que los directorios y archivos tengan permisos que su momento algunos llegan a ser igual entre los archivos y directorio, como los permisos de escritura, lectura y escritura, esto con el fin de evitar que se modifiquen, sean eliminados o simplemente evitar que alguien mas entre a perjudicar.

Décima cuarta. Es importante conocer que los permisos de dichos directorios o archivos, no pueden ser los mismos para todos los usuarios, ya que n todos cumplen con las misma funciones laborales, por lo que no todos tienen las mismas necesidades, es por ello que el administrador los divide en grupos para pueda tener permisos.

Décima quinta. Hay diversos tipos de protección por el cual se controlan los accesos a todo aquello que se tenga almacenado como los permisos de propietario y grupos.

Décima sexta. Por lo regular desde el momento que ejecutamos un programa, se genera un archivo de inicialización, por medio del cual se establecen opciones para manejar variables, es por ello que los intrusos son los principales interesados, ya que por medio de estas pueden lograr hacer modificaciones.

Décima séptima. Se puede esperar cualquier cosa, por parte de los piratas, hoy en día existen cada vez mas medios por los cuales ellos empiezan hasta lograr su objetivo, entrar al sistema, por eso es importante tomar las medidas necesarias, para así evitar desastres.

- Contraseñas aceptables
- Confidencialidad de las claves
- Conexiones cifradas
- Nunca, bajo ningún concepto, instalar o ejecutar *software* que no provenga de fuentes fiables
- Desconfianza

Un último consejo, si se muestra cualquier sospechosa que detectemos, aunque no nos implique directamente a nosotros, ha de ser notificada al administrador o responsable de seguridad del equipo. Esta notificación, a ser posible, no se ha de realizar por correo electrónico (un atacante puede eliminar ese *mail*), sino en persona o por teléfono, dar AVISO.

Décima octava. Hacer caso siempre a las recomendaciones de los administradores, de los expertos, porque no es en vano cuando nos dan una instrucción.

- Chequear periódicamente los *logs* en busca de actividades sospechosas.
- Utilizar órdenes como ps, netstat o last para detectar cualquier evento fuera de lo normal en el sistema, pero no confiar ciegamente en los resultados que se nos muestran en pantalla: seamos paranoicos.
- Comprobar periódicamente la integridad de ficheros importantes de nuestro sistema, como /etc/passwd, /etc/exports, /etc/syslog.conf, /etc/aliases o ficheros de arranque.
- Comprobar también elementos como los permisos o el propietario de los ficheros que se encuentran en los directorios de usuarios.

Décima novena. Una de las cosas que es importante ante todo es la prevención, que mejor que tener estrategias para actuar en un momento de peligro o en su defecto después.

Vigésima. Hay que tener paciencia para enfrentar los riegos y peligros que nuestro equipo, sistema pueda sufrir

- Nunca hay que ponerse nervioso: nuestra máquina ni ha sido la primera ni lamentablemente será la última en sufrir un ataque.
- Desconfiar de cualquier programa que se encuentre en el sistema; utilizar programas del CD-ROM del sistema operativo, o versiones estáticas de los mismos, para evitar sorpresas del intruso.
- Si es posible, reinstalar el sistema operativo completo y aplicarle los parches de seguridad que el fabricante pone a nuestra disposición; permanecer atentos a los directorios de usuarios y a los programas que éstos contienen.
- Si pensamos que la integridad del sistema peligra mucho, desconectar directamente el cable de red.
- Obviamente, antes de poner el sistema de nuevo a funcionar en red, estar completamente seguro que los problemas de seguridad que el atacante aprovechó están solucionados.

BIBLIOGRAFÍA.

- Seguridad en UNIX (Sistemas Abiertos e Internet).
  Riba Gorda Garnacho, A. Calvo Orra, M. Ángel Gallardo Ortiz.
  Edit. Paraninfo, Número de Páginas 217.
  Biblioteca Central de UNAM QA76.76063r49 código 461745.
- Introducción a UNIX Sistema IV.
  Rachel Morgan – Henry Mc. Gilton.
  Edit. Mc. Graw – Hill, Número de Páginas 625.
  Biblioteca Central de UNAM QA76.76063MG718.
- Teoría y diseño de los Sistemas Operativos.
  Juan M. Morena Pascual, Juan A. Pérez – Campanero.
  Edit. Anaya Multimedica, Número de Páginas 448.
  Biblioteca Central de UNAM QA76.76.063M67 código 471950
- La Biblia de UNIX
  Steve Moritsugu y DTR Business System
  Edit. WX Multimedia, Número de Páginas 780.
  Biblioteca IPN–ESCOM QA76.76.063
- Seguridad Práctica en UNIX e Internet O'Reilly
  Edit. Mc-Graw Hill, Número de Páginas 230.
  Biblioteca IPN- ESCOM 12391
- Introducción a UNIX
  George Meghabghab, Prentice – may
  Edit. Hispano Americana, S.A. Número de Páginas 330.
  Biblioteca IPN-ESCOM 6854.
- UNIX para inexpertos
  Levine, Megabyte
  Edit. Noriega editores, Número de Página s 466.
  Biblioteca IPN-ESCOM 2492.
  Páginas consultadas 45, 46.
- UNIX y LINUX
  M.Catalina Gallego, A. Catalina Gallego.

Edit. Mc Graw Hill, Número de Páginas 284.
Biblioteca IPN-ESCOM
Páginas consultadas 37 al 46, 269 al 276.

- UNIX Sistema V Versión IV 2da Edición.
  Kenneth H. Rosen, Richard R. Rosinski, James Farber, Douglas A. Host Osborne.
  Mc - Graw Hill, Número de Páginas 1337.
  Biblioteca IPN- ESCOM
- El Retorno de Programación UNIX
  Brian W. Kernighan, Rob Pike.
  Edit. PHH Prentice Hall, Número de Páginas 369.
- UNIX Sistema V. Versión IV (Manual de Referencia)
  Stephen Coffin
  Edit. Osborne Mc. Graw – Hill. Número de Páginas 745.
- El sistema UNIX y sus Aplicaciones.
  José Casanova
  Edit. AlfaOmega, Marcobombo, Número de Páginas 129.
- Introducción a UNIX un enfoque práctico
  Amir Afzal.
  Edit. Prentice Hall, Número de Páginas 455
- Procedimientos de datos UNIX con informix –SQL, ESQL/c, c-ISAM y Turbo.
  Kamkrishna S, Tarel.
  Edit. Mc-Graw Hill. Número de Páginas 370.
- UNIX y LINUX guía práctica 2da edición
  Sebastián Sánchez
  Edit. AlfaOmega RA-MA, Número de Páginas 385
- Seguridad en Computadoras
  Simson Garfinkel y Gene Spafford
  Edit. Mc Graw Hil, O'Reilly Número de Páginas 834.
  Biblioteca IPN- ESCOM

SITIOS WEB.

http://iie.fing.edu.uy/~vagonbar/unixbas/sisarch1.htm#ArchivosSysV

http://www.ace.ual.es/~jamartin/SOA_2005_2006_html/practica2.html

http://www.itistmo.edu.mx/PSA/UNIX.html

http://beta.redes-linux.com/manuales/seguridad/DeteccionDeIntrusos.pdf

http://master.blogdiario.com/

http://www.tiendalinux.com

http://platea.pntic.mec.es/