5

# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

# PROTECCIÓN DE LA INFORMACIÓN EN SERVIDORES UNIX, MEDIANTE TÉCNICAS, MÉTODOS Y HERRAMIENTAS.

## TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:
**LICENCIADO EN INFORMÁTICA**
PRESENTA:
**SAÚL FABIÁN FABIÁN**

ASESORA:

LIC. EN INFORMÁTICA LUZ MARÍA RAMÍREZ ROMERO



MÉXICO, D.F. ABR. 22 2002

2002

# PROTECCIÓN DE LA INFORMACIÓN EN SERVIDORES UNIX, MEDIANTE TÉCNICAS, MÉTODOS Y HERRAMIENTAS.

## AGRADECIMIENTOS.

A la Universidad Nacional Autónoma de México.

Por la oportunidad que brinda a la juventud Mexicana de seguir superándose día a día, abriendo sus puertas al mundo del conocimiento.

A la licenciada Luz María Ramírez Romero.

Con respeto por la asesoría, paciencia y dedicación brindada para la realización de este trabajo, y admiración por su amabilidad y entusiasmo que siempre la acompañan, pero sobre todo por su valiosa amistad ... gracias Luz.

A mis padres Rafael y Brigida, que con su comprensión y apoyo me dieron las fuerzas para concluir este proyecto.

A mi tío Aurelio.

Por haberme enseñado que el trabajo con esfuerzo y sacrificio es la actividad más hermosa que enaltece al hombre ... gracias tío.

A mis hermanos: Angel y Rafael por enseñarme donde se tiene la cabeza ... gracias hermanos. Angélica y Mari por enseñarme la dedicación en todo momento ... gracias hermanas y José Luis por enseñarme a tomar las decisiones con tanto valor ... gracias Luis.

A todos mis amigos y familiares por brindarme momentos agradables y compartir experiencias.

ÍNDICE.

## INTRODUCCIÓN.

Los problemas aumentan cuando un sistema informático se ve en la necesidad de incrementar sus servicios entre cliente – servidor en una red local (LAN), en una Intranet ó en Internet. Esta situación aumenta la complejidad del sistema posibilitando que un servidor sea comprometido en su seguridad.

Se sabe que los crímenes por computadora son generados[1] en un 6% por ex–trabajadores de la propia organización, 13% por gente externa y el 81% son responsabilidad de los propios empleados de la compañía.

Debido a que lo más valioso que hay en la mayor parte de los sistemas es la información que contienen se debe proteger del robo, falsificación o que sea borrada, en el valor de la información también puede influir el tiempo de reacción con el que se cuenta para recuperarse en caso de pérdida; es decir, que el sistema vuelva a ser operativo, y la información pueda ser accesible y confiable para todos los usuarios. El valor de la información también enfrenta otros riesgos como el robo o la falsificación con las consecuencias negativas que se tendrían si cayeran en manos no deseadas (por ejemplo de la competencia).

Para lograr que la información sea segura debemos empezar protegiendo desde su origen (el servidor) ya que los ataques pueden ser generados por usuarios de nuestra propia organización por lo cual debemos implementar medidas que permitan mantener confiable un servidor UNIX, después aseguraremos la información que fluye entre clientes – servidores en una red local, de una Intranet[2] o de Internet puesto que están expuestos de los ataques externos, razón por la cual necesitamos emplear medidas que aseguren los atributos de la información (es decir que sea veraz, oportuna, integra y confiable).

Cabe hacer notar que este trabajo pretende analizar lo que sucede en cualquier sistema operativo de UNIX, específicamente en una arquitectura basada en Intel con sistema operativo de Linux (Red Hat 6.1) y en una plataforma sparc de Sun Microsystems (Solaris 5.7), para demostrar si el uso de dichas herramientas hace más eficiente la seguridad y hacer nuestras consideraciones respecto de las ventajas o desventajas de los mismos, así como algunas aportaciones que no son suficientes pero si necesarias para mantener un sistema con una seguridad informática óptima y contribuir para que la información sea veraz, integra, confiable y oportuna.

La presente investigación inicia con los antecedentes de ataques informáticos que nos proporcionan un punto de partida, después analizar la seguridad interna de un servidor empezando con el sistema de archivos en el Sistema Operativo Unix, luego analizáremos el sistema bitácoras (logs) comentando sus ventajas y desventajas. Posteriormente veremos como proteger la información que fluye dentro y fuera del sistema; para ello continuaremos estudiando algunos algoritmos criptográficos que servirán para fundamentar el uso de las herramientas de seguridad, posteriormente estudiaremos dichas herramientas que nos permitirán mantener un sistema con un nivel optimo de seguridad, comentando sus ventajas y desventajas, para después proponer las recomendaciones basadas en el uso de una combinación de herramientas, ya sea en hardware o en software (firewall, tripwire, nmap, nessus, secure shell, cops) junto con una buena estrategia de reglas y políticas así como la necesidad de implantar un calendario de respaldos que nos dará la capacidad de mantener una seguridad informática óptima, de tal forma que generemos un ambiente de tranquilidad para todos y no hay que olvidar que:

***la seguridad la hacemos todos.***

---

[1] Dr. Gene Spafford, Congreso DISC 97, diciembre 1997.
[2] Es un conjunto de redes LAN cuyo objetivo es compartir sus recursos.

## OBSERVACIÓN.

El interés para la elaboración de este trabajo surgió por el acercamiento que tuve en mi servicio social en el centro de Informática de la Facultad de Contaduría y Administración (CIFCA), y posteriormente en la compañía Gamela México S.A. de C.V. (Nintendo de México) donde me percate de los problemas en materia de seguridad informática existente en sus Centros Informáticos de dichas organizaciones[3].

Los problemas que identifique fueron:

- Pérdida de la Información.
- Modificación de la Información.
- Accesos ilegales al sistema.
- Negación de servicios.

## PLANTEAMIENTO DEL PROBLEMA.

Para solucionar los problemas mencionados en la observación, investigue las causas, y con base a los fundamentos de éste trabajo propongo técnicas y herramientas que en conjunción con una buena estrategia de reglas y políticas para el uso de los equipos nos brindarán un nivel óptimo de seguridad informática a todos los usuarios de equipos de computo, incluyendo a los diseñadores y administradores de sistemas.

## HIPOTESIS.

Si las organizaciones en la que enviar o recibir información de carácter confidencial es parte de sus actividades cotidianas contaran con un sistema de seguridad bien planeado e implantado de manera eficiente, la productividad y prestigio de las mismas se incrementarán, reflejándose esto en la calidad de servicio que presta y en la confianza que las personas tienen hacia la organización compensando así el costo de la implantación con los beneficios que se obtengan a largo plazo.

## VARIABLE INDEPENDIENTE.

La carencia de una serie de herramientas, políticas y métodos en un sistema de información causará niveles de servicios inadecuados, así como el impedimento de detección de problemas con el sistema, afectará directamente a la productividad y prestigio de la organización quedando muy debajo de la competencia.

## OBJETIVO.

El objetivo de elaborar esta tesis es mostrar una serie de herramientas, técnicas y métodos como opciones para un administrador del sistema para que pueda contar con un esquema de seguridad informática que esté acorde a las necesidades del sistema, del presupuesto asignado y del rendimiento esperado.

Otro objetivo es generar una cultura de seguridad informática que permita tener información disponible, veraz, confiable y oportuna para todos los usuarios y así contribuir a alcanzar las metas y objetivos de la organización.

---

[3] Para identificar estos problemas se aplicó un cuestionario a cuatro personas que conocen la problemática de la organización, dichos cuestionarios se pueden ver en el apéndice 4.

# CAPÍTULO 1.

# ANTECEDENTES DE LA SEGURIDAD EN CÓMPUTO.

## 1.1. Introducción.

En este capítulo estudiaremos los antecedentes de la seguridad en cómputo que nos permitirá, de alguna manera, iniciar la elaboración de este trabajo.

El estudio de estos antecedentes es mostrar algunos conceptos relacionados con los ataques a sistemas, así como las posibles actividades que realizan los atacantes en un sistema informático.

También estudiaremos algunos casos donde se han infiltrado en los sistemas y han realizado algunas actividades en dichos sistemas. Estas actividades varían en función de lo que el atacante desee hacer, por mencionar algunas actividades: borrado de la información, modificación de la información, ver información de otras personas.

El objetivo de estudiar estos aspectos es:

A) Tomar conciencia de los posibles daños que puede generar un atacante a nuestro sistema.
B) Estar consciente de que cualquier sistema puede ser atacado.
C) La necesidad de preparar un mecanismo para defendernos de estos ataques.

También es importante mostrar los principales tipos de ataques o por los menos los más conocidos con el objetivo de comprender mejor los conceptos de aquellos tipos especiales de ataques informáticos para posteriormente plantear la importancia que tiene la información y la seguridad en materia informática, alternando una propuesta para solucionar esta problemática.

## 1.2. Antecedentes de la seguridad en cómputo.

El concepto de atacante de computadoras (hacker) comenzó a usarse aplicándolo a un grupo de pioneros de la informática del Instituto Tecnológico de Michigan (MIT), a principios de la década de 1960. Desde entonces, y casi hasta finales de la década de 1970, un hacker era una persona obsesionada por conocer lo más posible sobre los sistemas informáticos. Los diseñadores de sistemas operativos como Bill Gates (creador de Windows), Steve Jobs (creador de Aple), Ken Thompson y Dennis Ritchie (Creadores de UNIX), pueden considerarse hackers en este sentido de la palabra.

Pero a principios de la década de 1980, mucha gente (principalmente jóvenes) influenciados por la difusión de la película Juegos de Guerra, y el ampliamente publicado arresto de una "banda de hackers" conocida como la 414, los hackers pasaron a ser considerados como chicos jóvenes capaces de violar sistemas informáticos de grandes empresas y del gobierno.

El hacker puede realizar dos tipos de actividades: acceder a un sistema informático, o bien algo más general, como explorar y aprender a utilizar un sistema informático.

En la primera connotación, el término lleva asociados las herramientas y conocimientos propios para obtener cuentas de usuarios válidos de un sistema informático. Con respecto al segundo término un atacante (hacker) es una persona que tiene el conocimiento, habilidad y deseo de explorar completamente un sistema informático.

## 1.3. Hechos destacables.

En 1996, la página Web de Kriesgman (http://www.kriesgam.com/), una de las principales fábricas de pieles de Estados Unidos fue atacada[4] por unos chicos que pusieron imágenes y frases en defensa del animal y la ecología.

[4] Publicado por el Cert, en http://www.cert.org

- Hackers vulneraron Red del Pentágono.

En 1999 la red informática del Pentágono fue expuesta a intensas irrupciones de grupo de hackers. El Subsecretario estadounidense de Defensa, declaró a los medios que "se trata del ataque más organizado y sistemático experimentado por el Pentágono". La Secretaría de Defensa de Estados Unidos no desea proporcionar detalles del asunto a fin de no perjudicar las investigaciones correspondientes, que han sido encargadas al FBI. En tal sentido, se limitó a señalar que los hackers en ningún momento tuvieron acceso a información clasificada, sino "sólo" a los registros de personal y sueldos.

- Hackers atacan sitio de Hillary Clinton.

En 1999 la primera dama estadounidense, Hillary Clinton, aspira a convertirse en senadora por Nueva York. Como parte de su campaña, su equipo creó un sitio Web (http://www.hillary2000.com) que fue tomado por piratas informáticos.

La intervención realizada por los hackers no fue relativamente grave, ya que sólo implicó un redireccionamiento del URL[5], que hizo que quienes intentaran acceder al sitio Web de la candidata fuesen llevados a una página Web creada por "Los Amigos de Giuliani" también candidato para ser senador por Nueva York.

Jerry Irvine, experto consultado por CNN, señaló que lo más probable es que los hackers hayan recurrido a un truco conocido como DNS poisoning; es decir un "envenenamiento" del sistema de nombres de dominios (Domain Name Server), haciendo que al escribir una dirección en la Web los usuarios sean llevados a una dirección distinta.

Los autores de la página Web sobre Giuliani desmienten categóricamente ser los autores del sabotaje de la página de Hillary Clinton. A la fecha el problema no ha sido solucionado, por lo que la página de la candidata cambio a http://www.hillary2000.org.

- Hackers vulneran sitio de Symantec.

03 de Agosto de 1999. El sitio Web de Symantec fue alterado por hackers. La noticia causó revuelo en círculos informáticos, toda vez que la compañía es uno de los principales proveedores mundiales de software de seguridad y antivirus.

Como parte de esta interrupción contra los servidores de Symantec, los hackers cambiaron la portada del sitio Web corporativo con un texto que insultaba a la compañía. Según BBC News, los piratas informáticos también lograron infiltrar los servidores de Symantec con un programa tipo "gusano", que automáticamente se propaga por sistemas interconectados y que está en condiciones de causar daños similares a los virus.

Consultado por BBC, un portavoz de Symantec confirmó la alteración del sitio Web, aunque desmintió que los hackers hubieran logrado instalar un "gusano" en sus sistemas.

El portavoz intentó quitar importancia a la situación, señalando que siempre existe el riesgo de que una compañía se vea afectada por tales ataques y que lo importante es corregir el daño con prontitud y restablecer el sitio Web original.

A juicio del portavoz, el prestigio de Symantec no se verá alterado por el ataque, a pesar de ser una compañía líder del rubro de la seguridad informática.

Symantec denunció el hecho al FBI, que inició de inmediato las investigaciones correspondientes.

[5] Una URL es una dirección de página de Internet, por ejemplo: http://www.fca.unam.mx

## 1.4. Seguridad: barrera al comercio electrónico.

Recientemente ha aparecido publicada una encuesta sobre las barreras al comercio electrónico, llevada a cabo por ITAA (Information Technology Association of America) y la consultora Ernst & Young.

Es un hecho que el comercio electrónico no ha experimentado todavía el crecimiento ni la aceptación que el entusiasmo inicial pronosticaba para el futuro inmediato.

La encuesta tenía por cometido analizar cuáles eran los mayores factores que actúan como freno a la expansión de la actividad comercial en Internet y de acuerdo con los resultados obtenidos, la barrera más importante es: la falta de confianza hacia los sistemas (señalada por el 62% de los encuestados).

Esta desconfianza hacia las nuevas tecnologías, se articula en torno a tres temores fundamentales:

1) La privacidad, que los usuarios finales sienten amenazada en la medida en que desconocen hasta qué punto los datos personales que suministran a un servidor de comercio electrónico serán tratados de forma confidencial. Sienten incertidumbre cuando piensan que sus datos se almacenarán sin seguridad, siendo accesibles por un hacker o un desempleado desleal.

2) La autenticación, que preocupa a los usuarios, quienes dudan si la persona con la que se comunican es verdaderamente quien dice ser.

3) La seguridad global, que preocupa a los usuarios, pues temen que la tecnología no sea suficientemente robusta para protegerlos frente a ataques y apropiaciones indebidas de información confidencial, especialmente en lo que respecta a los medios de pago.

Es preocupante el hecho de que de toda la actividad de compra vía Internet, lo que más sigue preocupando es la operación de pago, es decir, el momento en el que el comprador se enfrenta a la ventana donde han introducido su número de tarjeta de crédito y duda a la hora de pulsar el botón de "Enviar", al pensar si será víctima de un robo.

Estos temores, tienen su fundamento real y su solución no resulta sencillo. En el primer caso, la tecnología, y en concreto la criptografía, ofrecen las herramientas necesarias para la protección de la información almacenada en las bases de datos corporativas, información como listas de clientes, sus datos personales y de pago, listas de pedidos, etc.

Existen muchas técnicas de control de acceso que hábilmente implantadas garantizan el acceso a la información confidencial exclusivamente a aquellos usuarios autorizados para ello. Ahora bien, se han producido incidentes de servidores de comercio que almacenaron esta clase de información sensible en archivos que eran consultados en una página Web siendo accesibles por cualquier navegante. Por lo tanto, aunque la criptografía ofrece medios aptos, depende en última instancia de la empresa que proporcione el nivel de compromiso respecto a la seguridad de los datos que conserva en sus archivos y su política de control de acceso.

La delgada línea que protege la privacidad del usuario está constituida en este caso por la integridad del servidor de la empresa.

La solución inmediata que ofrece la criptografía (codificación de la información) viene de la mano de los certificados digitales. La tecnología de certificación está suficientemente madura como para autenticar adecuadamente a las partes involucradas en una transacción. La más comúnmente utilizada es SSL y a pesar de la tan vapuleada limitación criptográfica fuera de Norteamérica sobre las claves débiles, lo cierto es que a la hora de autenticar a las partes, principalmente al servidor, SSL funciona satisfactoriamente.

Otra cuestión es: ¿incorporan los servidores de comercio todas las medidas necesarias para asegurar las transacciones con el usuario?. Las herramientas ofrecen solución tecnológica a los retos que se le presentan a la seguridad en el comercio electrónico, pero ¿se usa correctamente? ¿Se utiliza con todas sus características?. Parece que las verdaderas barreras al comercio electrónico no son tanto tecnológicas como humanas. Una vez más, el eslabón más débil de la cadena es de índole personal, no tecnológica.

## 1.5. Objetivos que se deben cuidar en seguridad en computo.

El objetivo es describir cuáles son los métodos más comunes que se utilizan hoy para perpetrar ataques a la seguridad informática (confidencialidad, integridad y disponibilidad de la información) de una organización o empresa, y que herramientas podemos implementar para la defensa, ya que saber cómo nos pueden atacar (y desde donde), es tan importante como saber con que soluciones contamos para prevenir, detectar y reparar un siniestro de este tipo. Estas soluciones son una combinación de herramientas que tienen que ver con tecnología y recursos humanos (políticas, capacitación).

Los ataques pueden servir a varios objetivos incluyendo fraude, extorsión, robo de información, venganza o simplemente el desafío de penetrar un sistema. Esto puede ser realizado por empleados internos que abusan de sus permisos de acceso, o por atacantes externos que acceden remotamente o interceptan el tráfico de red.

A través de los años se han desarrollado formas cada vez más sofisticadas de ataque para explotar "hoyos de seguridad" en el diseño, configuración y operación de los sistemas. Esto permitió a los nuevos atacantes tomar control de sistemas completos, produciendo verdaderos desastres que en muchos casos llevaron a la desaparición de aquellas organizaciones o empresas con altísimo grado de dependencia tecnológica (servicios automatizados, etc.).

### 1.5.1. Eavesdropping y Packet Sniffing.

Muchas redes son vulnerables al eavesdropping, o la pasiva intercepción (sin modificación) del tráfico de red. En Internet esto es realizado por packet sniffers, que son programas que monitorean los paquetes de red que están direccionados a la computadora donde están instalados. El sniffer puede ser colocado tanto en una estación de trabajo conectada a red, como a un equipo router o a un gateway de Internet, y esto puede ser realizado por un usuario con legítimo acceso, o por un intruso que ha ingresado por otras vías.

Este método es muy utilizado para capturar nombres de usuario y sus passwords, que generalmente viajan claros (sin encriptar) al ingresar a sistemas de acceso remoto.

### 1.5.2. Snooping y Downloading.

Los ataques de esta categoría tienen el mismo objetivo que el sniffing, obtener la información sin modificarla. Sin embargo los métodos son diferentes. Además de interceptar el tráfico de red, el atacante ingresa a los documentos, mensajes de correo electrónico y alguna otra información, guardando en la mayoría de los casos esa información a su propia computadora.

El Snooping puede ser realizado por simple curiosidad, pero también es realizado con fines de espionaje y robo de información o software. Los casos más populares de este tipo de ataques fueron: el robo de un archivo con más de 1700 números de tarjetas de crédito desde una compañía de música mundialmente famosa, y la difusión ilegal de reportes oficiales reservados de las Naciones Unidas, acerca de la violación de derechos humanos en algunos países europeos en estado de guerra.

### 1.5.3. Tampering o Data Diddling.

Esta categoria se refiere a la modificación desautorizada a los datos, o al software instalado en un sistema, incluyendo borrado de archivos. Estos ataques, son particularmente serios, cuando el que lo realiza ha obtenido derechos de administrador (super usuario), tiene la capacidad de ejecutar cualquier comando y alterar o borrar cualquier información, hasta llegar a la baja total del sistema en forma deliberada. Aún si no hubo intenciones de ello, el administrador posiblemente necesite dar de baja por horas o días hasta revisar y tratar de recuperar aquella información que ha sido alterada o borrada.

Como siempre, esto puede ser realizado por atacantes internos (insiders) o atacantes externos (outsiders), generalmente con el propósito de fraude o dejar fuera de servicio un competidor.

Múltiples páginas Web (sites) han sido víctimas del cambio de sus páginas principales (home page) por imágenes terroristas, insultos o chistes, o el reemplazo de versiones de software para guardar (download) por otros con el mismo nombre pero que incorporan código malicioso (virus, troyanos).

### 1.5.4. Spoofing.

Esta técnica es utilizada para actuar en nombre de otros usuarios, usualmente para realizar tareas de snoofing o tampering. Una forma comun de spoofing, es conseguir el nombre y password de un usuario legítimo para, una vez ingresado al sistema, tomar acciones en nombre de él, como puede ser el envío de falsos correos electrónicos.

El intruso usualmente utiliza un sistema para obtener información e ingresar en otro, y luego utiliza éste para entrar en otro, y en otro. Este proceso, llamado Looping, tiene la finalidad de evaporar la identificación y la ubicación del atacante. El camino tomado desde el origen hasta el destino puede tener muchas posiciones, que pueden exceder los límites de un país. Otra consecuencia del looping es que una compañía o gobierno pueden suponer que están siendo atacados por un competidor o quizá una agencia de gobierno extranjera, cuando en realidad están seguramente siendo atacados por un insider, o por un estudiante a miles de kilómetros de distancia, pero que ha tomado la identidad de otros.

Los protocolos de red también son vulnerables al spoofing. Con el IP spoofing, el atacante genera paquetes de Internet con una dirección de red falsa, pero que es aceptada por el destinatario del paquete.

### 1.5.5. Jamming o Flooding.

Este tipo de ataques desactivan o saturan los recursos del sistema. Por ejemplo, un atacante puede consumir toda la memoria o espacio en disco disponible, así como enviar tanto tráfico a la red que nadie más puede utilizarla.

Muchos ISPs (proveedores de Internet) han sufrido bajas temporales del servicio por ataques que explotan el protocolo TCP. Aquí el atacante satura el sistema con mensajes que requieren establecer conexión. Sin embargo, en vez de proveer la dirección IP del emisor, el mensaje contiene falsas direcciones IP (o sea que este ataque involucra también spoofing).

Otra acción común es la de enviar millares de correos electrónicos sin sentido a todos los usuarios posibles en forma continua, saturando los distintos servidores destino.

### 1.5.6. Caballos de Troya.

Consiste en introducir dentro de un programa una rutina o conjunto de instrucciones, por supuesto no autorizadas y que la persona que lo ejecuta no conoce, para que dicho programa actúe de una forma diferente a como estaba previsto (Formatear el disco duro, modificar un archivo, enviar un mensaje, etc.).

## 1.6 Explotación de errores de diseño, implementación u operación.

Muchos sistemas están expuestos a "hoyos de seguridad" (bugs) que son explotados para acceder a archivos, obtener privilegios o realizar sabotaje. Los sistemas operativos abiertos como UNIX tienen "hoyos de seguridad" más conocidos y controlados que aquellos que existen en sistemas operativos cerrados, como Windows NT, 2000.

## 1.7. La seguridad total es muy cara.

Hoy es imposible hablar de un sistema ciento por ciento seguro, sencillamente porque el costo de la seguridad total es muy alto. Por eso las empresas, en general, asumen riesgos: deben optar entre perder un negocio o arriesgarse a ser atacadas (hackeadas). La cuestión es que, en algunas organizaciones, tener un sistema de seguridad muy limitado les impediría hacer más negocios, "Si un hacker quiere gastar cien mil dólares en equipos para descifrar un mensaje codificado, lo puede hacer porque es imposible de controlarlo. Y una organización por no tratar de evitarlo se podrían gastar millones de dólares".

La solución a medias, entonces, sería acotar todo el espectro de seguridad, en lo que hace a plataformas, procedimientos y estrategias. De esta manera se puede controlar todo un conjunto de vulnerabilidades, aunque no se logre la seguridad total. Y esto significa un gran avance con respecto a unos años atrás.

## 1.8. Seguridad informática.

Toda organización debe estar a la vanguardia tecnológica, y deben de tener la capacidad de disponer de información confiable y en tiempo oportuno, esto constituye una ventaja fundamental, así se puede asegurar las siguientes ideas:

- Donde se tiene la información, es tener poder.
- La información se reconoce como:

    1. Crítica, indispensable para garantizar la continuidad operativa de la organización.
    2. Válioso, es un activo de la organización que tiene valor en sí mismo.
    3. Sensitiva, debe ser conocida por las personas que necesitan los datos.
    4. Donde identificar los riesgos de la información es de vital importancia.

Por consiguiente, la seguridad informática debe garantizar:

- La disponibilidad de los sistemas de información.
- La recuperación rápida y completa de los sistemas de información.
- La integridad de la información.
- La confidencialidad de la información.

## 1.9. Conclusión.

Como ya se dijo en el presente capítulo la información en un sistema informático, es muy valiosa para una organización porque permite a los integrantes de dicha entidad a tomar decisiones, corregir los errores del pasado y hacer planes para el futuro, y para lograr esto la información que se proporcionará tiene que estar siempre disponible, sebe ofrecerce oportunamente, debe ser veraz y que puedan hacer uso de ella quienes tienen autorización para ello.

Si la información no se protege, se corre el riesgo de un atacante la tome y con esto pierda su valor así como su utilidad porque ya no es confiable, generando un malestar para los integrantes de toda la organización, la información también corre el peligro de sufrir daños por robo, falsificación, eliminación generando un daño a nuestro prestigio.

Con la protección de la información se brinda a todos los integrantes de dicha entidad la confianza, para que usen la información que el sistema genera, y puedan usarla en procesos posteriores y así contribuir de esta manera con los objetivos de la organización.

Hay que estar conscientes de que un sistema con poca seguridad informática tiene más posibilidades de ser atacada y generar con ello desconfianza en la información del sistema, debido a ésta problemática se necesita hacer un plan que nos permita defender el valor de la información y proporcionar las condiciones necesarias para que el valor de la información se refleje en el bienestar de la organización.

En el próximo capítulo se explicará que es un sistema de archivos, cúal es su importancia y se análizaran técnicas para protegerlos. Asimismo se explicará la influencia o relación que tiene la protección de archivos del sistema en una organización.

# CAPÍTULO 2.

# EL SISTEMA DE ARCHIVOS.

## 2.1. Introducción.

La protección de la Información debe empezar con una correcta configuración del servidor, en este capitulo se estudiará todo lo relacionado con los archivos en UNIX, ya que comprender sus virtudes y desventajas ayuda a crear un esquema de reglas y políticas que puede mantener a nuestro servidor proporcionando sus servicios. Los conceptos se estudiarán en este capítulo son:

- Saber que significa un archivo en UNIX.
- El concepto de los permisos sobre los archivos en UNIX.
- Controlar los archivos SUID, SGID.
- Como un atacante puede aprovechar estas circunstancias para obtener privilegios de Super Usuario (root)[6].
- Comprender que significa montar y desmontar un sistema de archivos, así como sus ventajas y desventajas.
- Estudio de comandos que nos ayudan a mantener controlado el sistema de archivos.

En éste capítulo, se analizarán algunos aspectos relacionado con los archivos, el objetivo es mostrar que existen permisos sobre estos archivos, permisos tradicionales como el de lectura, escritura y ejecución son los más comunes, pero también existen otros permisos como el SUID, SGID los cuales son especiales para el sistema, explicaremos sus beneficios y los riesgos que representan.

También se estudiará algunas ventajas que se puede aprovechar de los atributos de los archivos relacionados con la seguridad del servidor y las listas de control de acceso que ofrecen un nivel de seguridad adicional a los clásicos permisos sobre los archivos en UNIX.

Para concluir estas ideas es importante explicar que significa el concepto de jerarquía de archivos así también explicar que cada directorio que ya está dentro del sistema de archivos tiene una razón de ser, es decir, cada directorio fue creado para contener clases de archivos que por su función en el sistema las hace ser comunes. El objetivo de estos conceptos es comprender que respetando esta clasificación nos ayuda a mantener un control sobre el sistema, y nos facilita el mantenimiento del mismo, por ejemplo esto permite ubicar en forma rápida y fácil los archivos que son respaldados.

## 2.2. El sistema de archivos.

En este capítulo se estudiarán temas referentes a la seguridad de los archivos y el sistema de archivos.

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

En un sistema UNIX típico existen tres tipos básicos de archivos: archivos de texto claro[7], directorios y archivos especiales (dispositivos). Los archivos de texto claro son secuencias de bytes

[6] La cuenta de root es la cuenta de administrador de los sistemas UNIX, conocida también como super usuario, puede situarse en cualquier sitio del sistema de archivos y realizar el mantenimiento del sistema; instalar paquetes, actualizar software, borrar archivos. Entrar al sistema como root le permitirá tener control total sobre el sistema sin restricción alguna.

[7] También conocido como texto plano, que son archivos cuyo contenido esta formado por caracteres ascii.

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 (texto, directorios, archivos especiales). Los archivos especiales son archivos que representan dispositivos del sistema; este último tipo se divide en dos grupos: los dispositivos orientados a carácter por ejemplo terminales, impresoras, cintas magnéticas y los orientados a bloque por ejemplo unidades de disco duro.

El sistema de archivos[8] es la parte del núcleo[9] (kernel) más visible por los usuarios; se encarga de abstraer propiedades físicas de diferentes dispositivos para proporcionar una interfaz única de almacenamiento: el archivo. Cada sistema UNIX tiene su sistema de archivos nativo[10], por lo que para acceder a todos ellos de la misma forma el núcleo de UNIX incorpora una capa superior denominada VFS (Virtual File System) encargada de proporcionar un acceso uniforme a diferentes tipos de sistema de archivos.

## 2.3. Permisos de un archivo.

Los permisos de cada archivo son la protección más básica de estos objetos del sistema operativo,[11] definen quién puede acceder a cada uno de ellos, y de qué forma puede hacerlo. Cuando se hace un listado largo de ciertos archivos podemos ver sus permisos junto al tipo de archivo correspondiente, en la primera columna de cada línea; ejemplo:

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

En este caso el archivo listado es de texto claro (el primer carácter es un "-") y sus permisos son "rwxr--r--". En este ejemplo los permisos se dividen en tres ternas en función de a qué usuarios afectan; cada una de ellas indica la existencia o la ausencia de permiso para leer, escribir o ejecutar el archivo: una r indica un permiso de lectura (read), una w de escritura (write), una x de ejecución (execution) y un "-" indica que el permiso no está activado. De esta manera, 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.

La primera terna afecta al propietario del archivo, la segunda al grupo de usuarios al que pertenece el propietario cuando lo creó (recordemos un mismo usuario puede pertenecer a varios grupos) y la tercera al resto de usuarios en el sistema. De esta forma, volviendo al ejemplo anterior, tenemos los permisos siguientes:

Tabla 2.1. Explicación de permisos para las ternas: "rwx r-- r--".

| | |
|---|---|
| Propietario (root): lectura, escritura y ejecución. | r w x |
| Miembros del grupo (sys): lectura. | r - - |

[8] Estructura jerárquica de archivos y directorios. De esta forma los recursos que el sistema posee y los archivos, serán fácilmente localizados. Esta distribución forma el árbol de directorios, el cual comienza con "/ ", también conocido como "directorio raíz".

[9] El objetivo del kernel o núcleo, es que el software y el hardware de la computadora puedan trabajar juntos. Sus funciones son: administración de la memoria, para todos los programas en ejecución; administración del tiempo del procesador, que éstos programas en ejecución utilizan; se encarga de que podamos acceder a los periféricos y/o elementos de nuestra computadora de una manera cómoda.

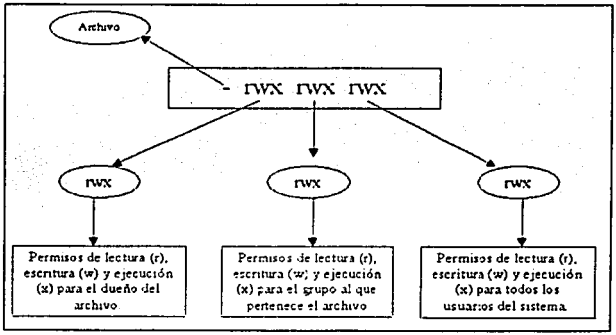
[10] En Linux se llama ext2, en Solaris ufs, en IRIX efs, etc.

[11] Un Sistema Operativo es un conjunto de programas cuya función es administrar los recursos del sistema. Sus funciones básicas son: administrar memoria, administrar tiempos de proceso del CPU, proporcionar una interfaz con el usuario, controlar dispositivos de entrada y salida.

| | |
|---|---|
| Resto de usuarios: lectura. | r - - |

Cuando un usuario intenta acceder en algún modo a un archivo, el sistema comprueba qué terna de permisos es la aplicable y se basa únicamente en ella para conceder o negar el acceso; en este caso, si un usuario es el propietario del archivo sólo se comprueban permisos de la primera terna; si no, se pasa a la segunda terna y se aplica en caso de que el permiso para el grupo coincida, y de no ser así se aplican los permisos de la última terna. De esta forma es posible tener situaciones tan curiosas como la de un usuario que no tenga ningún permiso sobre uno de sus archivos, y en cambio que el resto de usuarios del sistema pueda leerlo, ejecutarlo o incluso borrarlo. Este caso no es habitual, y de suceder el propietario siempre podrá restaurar los permisos a un valor adecuado.

Cuadro 2.1. Permisos de un archivo.




El propietario y el grupo de un archivo se pueden modificar con los comandos chown y chgrp[12] respectivamente; ambas reciben como parámetros al menos el nombre de usuario o grupo (los nombres válidos de usuario son los que poseen una entrada en /etc/passwd mientras que los grupos válidos se leen de /etc/group), al que se otorgará la posesión del archivo, así como el nombre de archivo a modificar.

```
# ls -l archivo
-rw-r---wx      1       root    sistemas        799     Feb     8       19:47   archivo

# chgrp administracion archivo
# ls -l /tmp/archivo
-rw-r---wx      1       root    administracion  799     Feb     8       19:47   archivo
#
```

En muchas variantes de UNIX es posible cambiar a la vez el propietario y el grupo de un archivo mediante chown, separando ambos mediante un carácter especial, generalmente ":" o ".".

```
# ls -l archivo
-rw-r---wx      1       root    administracion  799     Feb     8       19:47   archivo
# chown saul:sistemas archivo
```

[12] Comandos de UNIX que cambian a un archivo de propietario (chown) o de grupo (chgrp).

```
# ls -l /tmp/archivo
-rw-r---wx      1       saul    sistemas        799     Feb     8       19:47   archivo
#
```

Como se ve, ahora el propietario de archivo es el usuario saúl y pertenece al grupo de sistemas, sin embargo ninguno de estos comandos altera el campo de permisos; para modificar los permisos de un archivo se utiliza el comando chmod. Este comando generalmente recibe como parámetro el permiso en octal que se quiere asignar a cierto archivo, así como el nombre del mismo:

```
# ls -l archivo
-rw-r---wx      1       saul    sistemas        799     Feb     8       19:47   archivo

# chmod 755 archivo
# ls -l /tmp/archivo
-rwxr-wr-x      1       saul    sistemas        799     Feb     8       19:47   archivo
#
```

Para obtener el número en octal a partir de una terna de permisos determinada de los que se quiere calcular su equivalente octal, o que conocemos los permisos a asignar pero no su equivalente numérico; por ejemplo, para asignar a un archivo la terna rw-r---wx, lo primero que se debe hacer a partir de los bits rwx es calcular su equivalente binario, por lo que se asigna el valor "1" si un determinado permiso está activo (es decir, si aparece una r, w o x en él) y un "0" si no lo está (aparece un "-"); así, el equivalente binario de la terna propuesta es 110100011.

Ahora se debe pasar el número del sistema binario al octal: se divide en grupos de tres elementos (110, 100, 011) y de cada uno de ellos se calcula su equivalente octal.

Cuadro 2.2. Ejemplo para generar el permiso 643.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |
| Para 110 | → | | | 1 | 1 | 0 | = | $0\times2^0+1\times2^1+1\times2^2$ |
| | | | | | | | = | 0 + 2 + 4 |
| | | | | | | | = | 6 |
| Para 100 | → | | | 1 | 0 | 0 | = | $0\times2^0+0\times2^1+1\times2^2$ |
| | | | | | | | = | 0 + 0 + 4 |
| | | | | | | | = | 4 |
| Para 011 | → | | | 0 | 1 | 1 | = | $1\times2^0+1\times2^1+0\times2^2$ |
| | | | | | | | = | 1 + 2 + 0 |
| | | | | | | | = | 3 |
| Así para el permiso 110, 100, 011 genera los permisos : 643. | | | | | | | | |

De esta manera se obtienen los tres números de la terna de permisos, o lo que es lo mismo, la representación octal de los bits iniciales: 643. Por tanto, si se requiere asignar esta terna a un cierto archivo, se debe ejecutar el comando chmod indicándole este número y el nombre del archivo:

```
# chmod 643 archivo
# ls -l archivo
```

```
-rw-r---wx      1       saul    sistemas        799     Feb     8       19:47   archivo
#
```

Esto significa que el dueño del archivo (root) tiene los permisos de lectura, escritura, pero no de ejecución. Para el grupo al que pertenece (sistemas) tiene permiso de lectura pero no de escritura o ejecución, finalmente para cualquier otro usuario no tiene permiso de leer, pero sí de escribir o ejecutar.

La forma de trabajar de chmod comentada en el párrafo anterior, requiere que se indique explícitamente el valor octal de los bits rwx que se desea otorgar a un archivo; sin importar el valor de las ternas que poseía antes de ejecutar el comando, así se asignan los permisos del archivo al nuevo valor, indicado en la línea de comandos. Existe otra forma de trabajo de chmod denominada "simbólica" en la que no se indica el valor octal de todos los bits, sino que se especifica únicamente parámetros para los valores de los permisos que el archivo posee y se quiere modificar. En lugar de utilizar el equivalente octal, se puede utilizar símbolos (de ahí el nombre de esta forma de trabajo) que representan la activación o desactivación de ciertos bits en cada una de las tres ternas, la sintaxis básica de chmod en este caso son: los bits suid, sgid y sticky.

Se Puede ver que el valor simbólico comienza por cero o más letras que indican sobre qué terna de los permisos se van a realizar los cambios (estos valores son: "u" para propietario del archivo, "g" para grupo, "o" para resto de usuarios, "a" para las tres ternas; si no se especifica ninguna letra, se asume un default. A ellas les sigue un signo "+" o "-" en función de sí se activa o desactiva el bit sobre el que se está trabajando, parámetro indicado por el último conjunto formado por una o más letras, r para el permiso de lectura, w para escritura, x para ejecución, s para suid o sgid y t para bit de permanencia (Sticky Bit)[13].

Ejemplo.

```
# ls -l /tmp/archivo
-r--------      1       saul    sistemas        902     Feb     9       05:05   /tmp/archivo

# chmod +x /tmp/archivo
# ls -l /tmp/archivo
-r-x--x--x      1       saul    sistemas        902     Feb     9       05:05   /tmp/archivo

# chmod og-x /tmp/archivo
# ls -l /tmp/archivo
-r-x------      1       saul    sistemas        902     Feb     9       05:05   /tmp/archivo

# chmod ug+rwx /tmp/archivo
# ls -l /tmp/archivo
-rwxrwx---      1       saul    sistemas        902     Feb     9       05:05   /tmp/archivo
#
```

Esta forma de trabajo simbólica es menos utilizada en la práctica que la forma octal, pero en ciertas situaciones es muy útil, por ejemplo si se desea activar todos los permisos de ejecución de un archivo o si se quiere setuidarlo: con chmod +x o chmod u+s es suficiente en estos casos, y no hay motivo para preocuparse si modificamos el resto de permisos.

## 2.4. Los bits suid, sgid y sticky.

[13] Este bit provoca que el programa al que se le aplica quede residente en memoria de forma que la próxima vez que sea llamada su carga sea más rápida. Cuando se le aplica a un directorio, impide que nadie más que su dueño pueda cambiar o borrar ningún archivo del mismo.

Habitualmente, los permisos de los archivos en UNIX se corresponden con un número en octal que varía entre 000 y 777; sin embargo, existen unos permisos especiales que hacen variar ese número entre 0000 y 7777: se trata de los bits de permanencia (1000), sgid (2000) y suid (4000). El bit de suid o setuid se activa sobre un archivo añadiéndole 4000 a la representación octal de los permisos del archivo y otorgándole además permiso de ejecución al propietario del mismo; al hacer esto, en lugar de la x en la primera terna de los permisos, aparecerá una s o una S si no se ha otorgado el permiso de ejecución correspondiente (en este caso el bit no tiene efecto):

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

# ls -l /tmp/file1
-rwsrwxrwx    1    root    other    0    Feb    9    17:51    /tmp/file1

# ls -l /tmp/file2
-r-Sr--r--    1    root    other    0    Feb    9    17:51    /tmp/file2
#
```

El bit suid activado sobre un archivo indica que todo aquél que ejecute el archivo va a tener durante la ejecución los mismos privilegios que quién lo creó; dicho de otra forma, si el administrador crea un archivo y activa el bit de setuid, todo aquel usuario que lo ejecute va a disponer, hasta que el programa finalice, de un modo de privilegio total,[14] en el sistema. Podemos verlo con el siguiente ejemplo:

```
saul:/home/Gloria # cat testsuid.c
#include <stdio.h>
#include <unistd.h>
main()
{
        printf("UID: %d, EUID: %d\n",getuid(),geteuid());
}

saul:/home/Gloria # cc -o testsuid testsuid.c
saul:/home/Gloria # chmod u+s testsuid
saul:/home/Gloria # ls -l testsuid
-rwsr-xr-x    1    root    root    4305    Feb    10    02:34    testsuid

saul:/home/Gloria # su Gloria
saul: $ id
uid=1000(Gloria ) gid=100(users) groups=100(users)
saul: $./testsuid
UID: 1000, EUID: 0
saul: $
```

Se puede observar, que el usuario Gloria, sin ningún privilegio especial en el sistema, cuando ejecuta el programa, con el bit setuid activo de prueba, está trabajando con un euid (Effective UID) 0, lo que le otorga todo el poder del administrador (observe que éste último es el propietario del ejecutable); si en lugar de este código el ejecutable fuera una copia de un shell,[15] el usuario Gloria tendría todos los privilegios del root, mientras no finalice la ejecución, es decir, hasta que no se teclee exit en la línea de comandos.

[14] Es tener todas las facultades del usuario root.

[15] Es un programa de UNIX, cuya finalidad es proporcionar la interfaz entre el sistema operativo y el usuario. Los usuarios pueden personalizar sus shells, adecuando a sus propias necesidades específicas. De esta forma los programas escritos por los usuarios normalmente se ejecutan por medio de un shell. (intérprete de comandos)

Lo que se acaba de comentar con respecto al bit setuid es aplicable al bit setgid pero al nivel de grupo del archivo en lugar del propietario: en vez de trabajar con el uid del propietario, todo usuario que ejecute un programa con el bit setgid activado, tendrá los privilegios del grupo al que pertenece el archivo.

Para activar el bit de setgid se suma 2000 a la representación octal del permiso del archivo y además se debe dar permiso de ejecución a la terna de grupo; al hacer esto, la s o S aparecerá en lugar de la x en esta terna. Si el archivo es un directorio y no un archivo plano, el bit setgid afecta a los archivos y subdirectorios que se creen en él: éstos tendrán como grupo propietario al mismo que el directorio con el bit setgid activado, siempre que el proceso que los cree pertenezca a dicho grupo.

Esto afecta a la seguridad del sistema porque los bits de setuid y setgid dan a UNIX una gran flexibilidad, pero constituyen al mismo tiempo la mayor fuente de ataques internos al sistema (entendiendo por ataques internos aquellos realizados por un usuario – autorizado o no – desde la propia máquina, generalmente con el objetivo de aumentar su nivel de privilegio en la misma). Cualquier sistema UNIX tiene un cierto número de programas con el bit setuid activado y/o, con el bit setgid activado. Cada uno de ellos, como acabamos de comentar, se ejecuta con los privilegios de quien lo creó (generalmente el root u otro usuario con ciertos privilegios) lo que directamente implica que cualquier usuario tiene la capacidad de lanzar tareas que escapen total o parcialmente al control del sistema operativo: se ejecutan en modo privilegiado[16] si es el administrador quien creó los ejecutables. Estas tareas han de estar controladas de una forma exhaustiva, ya que si una de ellas se comporta de forma anormal (un core dump) puede causar daños irreparables al sistema.

Si por cualquier motivo un programa con el bit de setuid activado falla, se asume inmediatamente que presenta un problema de seguridad para la máquina, y se recomienda desactivar el bit de setuid cuanto antes. Asegurar completamente el comportamiento correcto de un programa, es muy difícil por no decir imposible. Cada cierto tiempo suelen aparecer fallos (bugs)[17] en archivos con el bit setuid activo de los diferentes clones de UNIX[18] que ponen en peligro la integridad del sistema.

Se estudiará un ejemplo: un archivo con el bit setuid activo clásico en cualquier clon UNIX es /bin/passwd, el comando para que los usuarios puedan cambiar su contraseña de entrada al sistema. Una de sus funciones consiste en modificar el archivo de claves (/etc/passwd o /etc/shadow). Un usuario normal no tiene el nivel de privilegio[19] necesario para hacer esto (incluso es posible que ni siquiera pueda leer el archivo de claves), por lo que frente a este problema tan simple existen varias soluciones: se puede asignar permiso de escritura para todos los usuarios al archivo de contraseñas, se puede negar a los usuarios el cambio de clave o tal vez obligarlos a pasar con el administrador cada vez que quieran cambiar su contraseña. Ninguna estas soluciones es apropiada para la seguridad del sistema (quizás la última lo sea, pero es impracticable en máquinas con un número de usuarios considerable). Por tanto, se debe asumir que el bit de setuid en /bin/passwd es importante para un correcto funcionamiento del sistema. Sin embargo, en un sistema UNIX instalado predeterminado (out of the box) el número de archivos con el bit setuid activo, suele ser mayor de cincuenta; sin perjudicar al correcto funcionamiento de la máquina, este número se puede reducir a menos de cinco, lo que viene a indicar que una de las tareas de un administrador sobre un sistema recién instalado es minimizar el número de archivos setuidados o

---

[16] Es una o todas las facultades del usuario root.

[17] Caracteristica indocumentada de un programa, reflejo de un error.

[18] Son versiones de UNIX, patentadas por alguna empresa u organización, respetan las normas y estándares de POSIX y BSD, por ejemplo: AIX de IBM, Solaris de Sun microsystems, HP-UX de Hewlett Packard, IRIX de Silicon Graphics, SCO de Santa Cruz Operaciones, así como las versiones de Linux (Redhat, Debían, Mandrake, etc.) para más información puede ver el apéndice 1.

[19] Es tener una o todas las facultades para crear archivos, borrar archivos, instalar software, actualizar software, quitar software, situado en cualquier parte del sistema. Puede tener la misma función del usuario root.

setgidados. No obstante, tampoco es conveniente eliminarlos, sino simplemente desactivar su bit de setuid mediante el comando chmod:

```
# ls -l /bin/ping
-r-sr-xr-x    1    root    bin    14064  May   10    1999    /bin/ping

# chmod -s /bin/ping
# ls -l /bin/ping
-r-xr-xr-x    1    root    bin    14064  May   10    1999    /bin/ping
#
```

Es recomendable estar atentos sobre nuevos archivos de estas características que se localicen en el sistema; demasiadas aplicaciones de UNIX se instalan predeterminadamente con programas ejecutables con el bit setuid activado, cuando realmente este bit no es necesario, por lo que a la hora de instalar nuevo software o actualizar el existente se debe desactivar el bit de los archivos que no lo necesiten.

Especialmente grave es la aparición de archivos con el bit setuid activo de los que el administrador no tenía constancia (ni son aplicaciones del sistema ni aplicaciones añadidas), ya que esto casi en el 100% de los casos indica que nuestra máquina ha sido comprometida por un atacante. Para localizar los archivos con alguno de estos bits activos, se puede ejecutar el siguiente comando:

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

Por otra parte, el sticky bit o bit de permanencia se activa sumándole 1000 a la representación octal de los permisos de un determinado archivo y otorgándole además permiso de ejecución; al hacer esto, sepuede observar que en lugar de una x en la terna correspondiente al resto de usuarios aparece una t (si no se ha dado permiso de ejecución al archivo, aparecerá una T):

```
# chmod 1777 /tmp/file1
# chmod 1774 /tmp/file2
# ls -l /tmp/file1
-rwxrwxrwt    1    root    other  0      Feb   9     17:51   /tmp/file1

# ls -l /tmp/file2
-rwxrwxr-T    1    root    other  0      Feb   9     17:51   /tmp/file2
#
```

Si el bit de permanencia de un archivo está activado (recuerde que si aparece una T no lo está) le estamos indicando al sistema operativo que se trata de un archivo muy utilizado, por lo que es conveniente que permanezca en memoria principal el mayor tiempo posible; esta opción se utilizaba en sistemas antiguos que disponían de muy poca RAM[20], pero hoy en día prácticamente no se utiliza. Lo que aún que sigue vigente es el efecto del sticky bit activado sobre un directorio: en cuyo caso se indica al sistema operativo que, aunque los permisos "normales" digan que cualquier usuario pueda crear y eliminar archivos (por ejemplo, un 777 octal), sólo el propietario de cierto archivo y el administrador pueden borrar un archivo guardado en un directorio con estas características. Este bit, que sólo tiene efecto cuando es activado por el administrador (aunque cualquier usuario puede hacer que aparezca una t o una T en sus archivos y directorios), se utiliza principalmente en directorios del sistema de archivos en los que interesa que todos puedan escribir pero que no todos puedan borrar los datos escritos, como /tmp/ o /var/tmp/: si el equivalente octal de los permisos de estos directorios fuera 777 en lugar de 1777, cualquier usuario podría borrar los archivos del resto. Podría pensarse que para evitar estos problemas se debe negar la escritura en directorios como los anteriores, lo cual no es correcto puesto que programas - como compiladores,

[20] Random Access Memory (RAM). Memoria temporal que permanece hasta que la computadora se apaga. Una vez que se apago el equipo todo el contenido de la RAM se pierde.

editores o gestores de correo - asumen que van a poder crear archivos en /tmp/ o /var/tmp/, y estos programas los puede ejecutar cualquier usuario del sistema.

Para completar estas ideas retomemos lo comentado al principio del tema: el equivalente octal de los permisos en UNIX puede variar entre 000 y 777. Se ha visto que se podía sumar 4000, 2000 o 1000 a los permisos "normales" para activar respectivamente los bits setuid, setgid o sticky. Por supuesto, se puede activar varios de ellos a la vez simplemente sumando sus valores: en la situación poco probable de que necesitáramos todos los bits activos, sumaríamos 7000 a la terna octal 777:

```
# chmod 0 /tmp/archivo
# ls -l /tmp/archivo
----------    1    root    root    0    Feb    9    05:05    /tmp/archivo

# chmod 7777 /tmp/archivo
# ls -l /tmp/archivo
-rwsrwsrwt    1    root    root    0    Feb    9    05:05    /tmp/archivo
#
```

Si en lugar de especificar el valor octal de los permisos queremos utilizar la forma simbólica de chmod, utilizaremos +t para activar el bit de permanencia, g+s para activar el de setgid y u+s para hacer lo mismo con el de setuid; si queremos reiniciarlos, utilizamos un signo "-" en lugar de un "+" en la línea de comandos.

## 2.5. Atributos de un archivo y su relación con la seguridad del sistema.

En el sistema de archivos ext2 (Second Extended File System) de Linux existen ciertos atributos para los archivos que pueden ayudar a incrementar la seguridad de un sistema. Estos atributos son los mostrados en la tabla 2.2.

De todos ellos, de cara a la seguridad algunos no nos interesan demasiado, pero otros se deben tener muy en cuenta.

Tabla 2.2. Atributos de los archivos en ext2fs.

| Atributo. | Significado. |
|---|---|
| A | No puede actualizar cambio en el archivo. |
| S | Actualización Asincrona. |
| A | Modificable. |
| C | Archivo comprimido. |
| I | Archivo Inmutable. |
| D | No Dump. |
| S | Borrado seguro. |
| U | Archivo imborrable. |

Un atributo interesante es "a", tan importante es que sólo el administrador tiene el privilegio suficiente para activarlo o desactivarlo. El atributo "a" sobre un archivo indica que éste sólo se puede abrir en modo escritura para añadir datos, pero nunca para eliminarlos. La relación de lo anterior con la seguridad es que cuando un intruso ha conseguido el privilegio suficiente en un sistema atacado, lo primero que suele hacer es borrar sus huellas; para esto existen muchos programas (denominados zappers[21], rootkits[22], etc.) que, junto a otras funciones, eliminan

[21] Es un programa usado para borrar huellas en un sistema. Puede borrar los logs de utmp y wtmp. Así es más difícil detectar quién los ha eliminado.

estructuras de ciertos archivos de log como lastlog, wtmp o utmp. Así logran con esto que cuando alguien ejecute last, who, users, w o similares, no se de ninguna evidencia de la conexión que el atacante ha realizado a la máquina, si estos archivos de log poseen el atributo "a" activado, es más difícil que el atacante pueda borrar datos de ellos.

Ahora supongamos la siguiente cuestión: si el intruso ha conseguido el suficiente nivel de privilegio como para poder escribir - borrar - en los archivos (en la mayoría de versiones UNIX para realizar esta tarea se necesita ser root), simplemente ha de desactivar el atributo "a" del archivo, eliminar los datos comprometedores y volver a activarlo. Esto es muy simple, pero si el intruso tiene poca experiencia su acceso quedará convenientemente registrado en el sistema.

Otro atributo del sistema de archivos ext2 es "i" (archivo inmutable); un archivo con esta bandera activada no se puede modificar de ninguna forma, ni añadiendo datos ni borrándolos, ni eliminar el archivo, ni tan siquiera enlazarlo mediante ln[23]. Igual que sucedía antes, sólo el administrador puede activar o desactivar el atributo "i" de un archivo. Se puede aprovechar esta característica en los archivos que no se modifican frecuentemente, por ejemplo muchos de los contenidos en /etc/ (archivos de configuración, scripts de arranque, incluso el propio archivo de contraseñas siempre y cuando añadir o eliminar usuarios no sea frecuente en nuestro sistema). Así, de esta forma conseguimos que ningún usuario pueda modificarlos incluso, aunque sus permisos lo permitan.

Con la activación el atributo "i" en un archivo se debe tener siempre en cuenta que este no va a poder ser modificado por nadie, incluido el administrador, y tampoco por los programas que se ejecutan en la máquina; por tanto, si activáramos este atributo en un archivo de log, no se grabaría ninguna información en él, lo que no es conveniente. También se debe de recordar que los archivos tampoco van a poder ser enlazados, lo que puede ser problemático en algunas variantes de Linux que utilizan enlaces duros o fijos[24] para la configuración de los archivos de arranque del sistema.

Atributos que también pueden ayudar a implementar una correcta política de seguridad en la máquina son "s" y "S". Si borramos un archivo con el atributo "s" activo, el sistema va a rellenar los bloques en el disco que ocupaba ese archivo con ceros en lugar de efectuar un simple unlink(), para así dificultar la tarea de un atacante que intente recuperarlo; realmente, para un intruso experto esto no supone ningún problema, simplemente un retraso en sus propósitos. Por su parte, el atributo "S" sobre un archivo hace que los cambios sobre el archivo se escriban inmediatamente en el disco en lugar de esperar el sync del sistema operativo. Aunque no es lo habitual, bajo ciertas circunstancias un archivo de log puede perder información que aún no se haya escrito en el disco: por ejemplo que alguien se conecta al sistema y cuando registra la entrada, la máquina se apaga súbitamente; toda la información que aún no se haya grabado en disco se perderá. Aunque como decimos, esto no suele ser habitual, si nuestro sistema se apaga frecuentemente nos puede interesar activar el bit "S" de ciertos archivos importantes.

Una vez que se han explicado los atributos del sistema de archivos ext2 que pueden incrementar la seguridad de Linux; por ejemplo, sin entrar en muchos detalles cómo activar o desactivar estos atributos sobre archivos, y también cómo ver su estado. Para el primer caso se usa el comando chattr, que recibe como parámetros el nombre del atributo junto a un signo "+" o "-", en función de si se activa o desactiva el atributo, y también el nombre de archivo correspondiente. Si lo que

---

[22] Conjunto de programas conocidos como troyanos. Un troyano es un programa que parece ser legítimo, pero en realidad oculta un código peligroso, normalmente destinado a facilitar la entrada de un atacante o asegurar su retorno.

[23] Comando de UNIX que liga la trayectoria de un archivo con otra trayectoria, de manera que un mismo archivo pueda estar presente en dos trayectorias (PATH) distintas.

[24] Permite el acceso a los datos de un archivo desde otro nombre de archivo diferente. Los enlaces fijos garantizan la existencia de un archivo. Cuando se elimina el último enlace fijo, se suprimen el número del inodo y sus datos. Sólo pueden crearse enlaces fijos entre archivos que se encuentran en el mismo sistema de archivos.

deseamos es visualizar el estado de los diferentes atributos, utilizaremos ls -attr, cuya salida indicará con la letra correspondiente cada atributo del archivo o un signo - en el caso de que el atributo no esté activado:

```
# ls -attr /tmp/archivo
---------- /tmp/archivo

# chattr +a /tmp/archivo
# chattr +Ss /tmp/archivo
# ls -attr /tmp/archivo

s----Sa-- /tmp/archivo
# chattr -sa /tmp/archivo
# ls -attr /tmp/archivo
---S----- /tmp/archivo
#
```

## 2.6. Listas de control de acceso: acls.

Las listas de control de acceso (ACLs, Access Control Lists) ofrecen un nivel adicional de seguridad a los archivos extendiendo el clásico esquema de permisos en UNIX: mientras que con estos últimos sólo se especifican permisos para los tres grupos de usuarios habituales (propietario, grupo y resto), 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 versiones UNIX (Solaris, AIX, HP-UX, etc.), mientras que en otros que no lo proporcionan, como Linux, puede instalarse con un software adicional.

Los ejemplos que vamos a utilizar aquí se han realizado sobre Solaris; la idea es la misma en el resto de las variantes de UNIX, aunque pueden cambiar las estructuras de las listas. Para obtener una excelente visión de las ACLs es recomendable consultar la documentación de los diferentes clones de UNIX para detalles concretos de cada manejo e implementación.

Las listas de control de acceso permiten saber si a un usuario se le permite cierto tipo de acceso. sobre un archivo tenemos que hacer un listado largo:

```
# ls -l /usr/local/sbin/sshd
-rwx------   1   root   bin   2616160   Apr 28  1997   /usr/local/sbin/sshd
#
```

Viendo el resultado, directamente se sabe que el archivo sshd puede ser ejecutado, modificado y leído por el administrador (root), pero por nadie más; sin embargo, no conocemos el estado de la lista de control de acceso asociado al archivo. Para ver esta lista, en Solaris se ha de utilizar el comando getfacl:

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

Se visualizo una lista de control de acceso de Solaris; en primer lugar se indica el nombre del archivo, su propietario y su grupo, todos precedidos por "#". Lo que se ve a continuación es la propia lista de control: los campos user, group y other son básicamente la interpretación que getfacl hace de los permisos del archivo (observe que coincide con el resultado del ls -l). El campo mask es muy similar al umask clásico: define los permisos máximos que un usuario (con excepción del propietario) o grupo puede tener sobre el archivo. Finalmente, el campo effective nos informa, para cada usuario (excepto el propietario) o grupo el efecto que la máscara tiene sobre los permisos: es justamente el campo que tenemos que analizar si se quiere ver quién puede acceder al archivo y de qué forma. Podemos ver que la estructura de la lista de control de acceso otorga los mismos permisos que las ternas clásicas. Esto es algo normal en todos los UNIX: si no indicamos lo contrario, al crear un archivo se le asocia una ACL que coincide con los permisos que ese archivo tiene en el sistema (cada archivo tendrá una lista asociada, igual que tiene unos permisos); de esta forma, el resultado anterior no es más que la visión que getfacl tiene de los bits rwx del archivo.

Lo interesante de cara a la protección de archivos es extender los permisos clásicos del archivo, modificando su lista asociada. Esto lo podemos conseguir con la orden setfacl:

```
# setfacl -m user:Gloria :r-x /usr/local/sbin/sshd
# getfacl /usr/local/sbin/sshd
# file: /usr/local/sbin/sshd
# owner: root
# group: bin
user::rwx
user:Gloria :r-x            #effective:---
group::---                  #effective:---
mask:---
other:---
#
```

Como se observa, se acaba de modificar la lista de control de acceso del archivo para asignarle a Gloria permiso de ejecución y lectura sobre el mismo. El comando setfacl se utiliza principalmente de tres formas: para añadir entradas a la ACL, mediante la opción -m seguida de las entradas que se desea añadir, separadas por comas (lo que hemos hecho en este caso, aunque no se han utilizado comas porque sólo hemos añadido una entrada), o bien utilizamos el parámetro -s para reemplazar la ACL completa (hemos de indicar todas las entradas, separadas también por comas), o bien borramos entradas de la lista con la opción -d (de sintaxis similar a -m). Cada entrada de la ACL tiene el siguiente formato:

```
tipo:uid
j gid:permisos
```

El tipo indica a quién aplicar los permisos (por ejemplo, user para el propietario del archivo, o mask para la máscara), el UID indica el usuario al que deseamos asociar la entrada (como hemos visto, se puede utilizar también el login), y el GID hace lo mismo con el grupo (de la misma forma, se puede especificar su nombre simbólico[25] ). Finalmente, el campo de permisos hace referencia a los permisos a asignar, y puede ser especificado mediante símbolos rwx- o de forma octal. De esta manera el usuario Gloria tenga permiso de lectura y ejecución en el archivo; no obstante, si ahora este usuario intenta acceder al archivo con esta especificación obtendrá un error:

```
gloria:/usr/local/sbin$ id
uid=100(Gloria ) gid=10(staff)
gloria:/usr/local/sbin$ ./sshd
bash:./sshd: Permission denied
gloria:/usr/local/sbin$
```

[25] Es un nombre que hace referencia a un archivo o directorio, en alguna trayectoria (PATH).

Este error se produce porque está actuando la máscara sobre los permisos del archivo (antes hemos dicho que debemos fijarnos en el campo effective, y aquí podemos comprobar que no se ha modificado). Para solucionar este problema se debe modificar el campo mask:

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

Si ahora Gloria intenta acceder al archivo para leerlo o ejecutarlo, ya se le va a permitir:

```
gloria:/usr/local/sbin$ id
uid=100(Gloria ) gid=10(staff)
gloria:/usr/local/sbin$ ./sshd
/etc/sshd_config: No such file or directory
...
```

Aunque obtenga un error, este error ya no depende de la protección de los archivos sino de la configuración del programa en relación con las medidas de seguridad del sistema operativo UNIX[26]. No obstante, si que hay diferencias entre una ejecución de un usuario como Gloria comparadas con otra del usuario root, que son impuestas por el sistema operativo UNIX: Gloria no podría utilizar recursos a los que no le está permitido el acceso, como puertos bien conocidos, otros archivos, o procesos que no le pertenezcan. Hay que recordar que aunque un usuario ejecute un archivo perteneciente a root, si el archivo no está activado con el bit setuid los privilegios del usuario no cambian y por lo tanto no puede hacer uso de los archivos del sistema. Sucede lo mismo si el usuario tuviera permiso de ejecución normal sobre el archivo, pero éste realizara tareas privilegiadas, podría ejecutarlo, pero obtendría error al intentar violar la protección del sistema operativo.

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

```
# ls -l /usr/local/sbin/sshd
-rwx------+     1       root    bin     2616160         Apr 28  1997 /usr/local/sbin/sshd
#
```

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

```
# getfacl /usr/local/sbin/sshd >/tmp/archivo
# setfacl -f /tmp/archivo /usr/local/sbin/sshd
# getfacl /usr/local/sbin/sshd
# file: /usr/local/sbin/sshd
# owner: root
# group: bin
user::rwx
user:Gloria :r-x #effective:r-x
group::--- #effective:---
mask:r-x
other:---
#
```

Esto es equivalente a utilizar una tubería entre los dos comandos, lo que produciría el mismo resultado:

[26] Recordemos que los archivos del sistema, que son de root, no pueden ejecutarse por el resto de los usuarios, así sshd es un archivo de root, y el sistema no permitirá que nadie más lo ejecute si no es el usuario root.

# getfacl /usr/local/sbin/sshd | setfacl -f - /usr/local/sbin/sshd

Antes de finalizar este apartado dedicado a las listas de control de acceso, quizás sea conveniente comentar el principal problema de estos mecanismos. Las ACLs son de gran ayuda para el administrador de sistemas UNIX, tanto para incrementar la seguridad como para facilitar ciertas tareas. Imaginemos un caso en que un usuario autorizado de nuestro sistema aprovecha el último error (bug) de sendmail para conseguir privilegios de administrador en una máquina; cuando se ha convertido en root modifica la lista de control de acceso asociada a /etc/shadow y crea una nueva entrada que le da un permiso total a su login sobre este archivo. Una vez hecho esto, borra todo el rastro y da aviso del nuevo problema de sendmail, problema que rápidamente se soluciona, se le da las gracias y nos olvidamos del asunto. ¿Nos olvidamos del asunto? Tenemos un usuario que, aunque los bits rwx no lo indiquen, puede modificar a su gusto un archivo crucial para nuestra seguridad. Contra esto, poco se puede hacer; se debe comprobar frecuentemente los listados de todos los archivos importantes (ahora entendemos por qué aparece el símbolo "+" junto a las ternas de permisos), y si encontramos que un archivo tiene una lista de control que otorga permisos no reflejados en los bits rwx, analizar dicha lista mediante getfacl y verificar.

## 2.7. Jerarquía de archivos.

El nivel más alto de la jerarquía de archivos es conocido como root (/). El directorio root generalmente contiene varios otros directorios incluyendo:

Tabla 2.3. Directorios que integran la Jerarquía de archivos.

| Directorio. | Contenido. |
|---|---|
| root/ | Directorio home para el super usuario[27] (root), así mismo en otros UNIX se conoce su ubicación comenzando con / (raíz). |
| boot/ | Configuración de archivos de arranque del sistema para cargar el sistema operativo y la imagen del kernel (arranque del sistema[28]). |
| dev/ | Archivos de dispositivo. |
| etc/ | Sistema de archivos de configuración y programas (scripts). |
| home/ | Directorios de usuarios (y subdirectorios). |
| Lib/ | Librerías principales del sistema operativo y módulos del núcleo (kernel). |
| Lost + found/ | Directorio para almacenar archivos recuperados. |
| Mnt/ | Punto temporal para montar dispositivos. |
| proc/ | Pseudo directorio estructurado para contener información acerca del núcleo (kernel), procesos corriendo actualmente y asignación de recursos. |
| bin/ | Contiene binarios de arranque del sistema (Boot). |
| sbin/ | Sistema de administración de binarios y herramientas. |
| tmp/ | Ubicación de archivos temporales. |
| usr/ | Contiene una variedad de archivos, entre ellos binarios locales, librerías, aplicaciones y paquetes. |
| var/ | Datos variables, incluye directorios spool directorios para correo (mail) y para grupos de noticias (news). |

Este es un esquema general de un sistema de archivos. Es recomendable que no se incluyan otros directorios dentro de este nivel de archivos. Un típico error es crear un directorio de usuario fuera del directorio /home.

[27] Todos los sistemas UNIX tiene un superusuario. Su nombre de usuario (login) es "root".

[28] Proceso de iniciación de una computadora. Cuando el equipo se enciende, acude a un sector de la memoria ROM, en búsqueda de instrucciones determinadas. Éstas, a su vez, intentan localizar en el disco, otras órdenes que indican a la computadora las tareas a realizar. En ese momento, el proceso queda a cargo del sistema operativo. Se puede arrancar una computadora desde una unidad de CD, Disco flexible o en un sector del disco duro.

Para saber que de clase archivos corresponden a un directorio definido por el sistema, se estudiará con detalle éstos directorios (esto ayuda a mantener control en el sistema).

### 2.7.1. /home.

El directorio /home está estructurado para contener los directorios que serán asignados para los usuarios. (excepto algunos usuarios como: el usuario root y en algunos casos el usuario nobody) Generalmente los sistemas pequeños contendrán los directorios de usuarios en /home. (ejemplo: /home/saul).

En sistemas más grandes es recomendable estructurar el directorio /home basándose en clases o grupos de usuarios, por ejemplo:

```
/home/administración    # Usuarios del departamento de Administración.
/home/ventas            # Usuarios del departamento de ventas.
/home/compras           # Usuarios del departamento de compras.
/home/sistemas          # Usuarios del departamento de sistemas.
/home/otros             # Otros usuarios.
```

### 2.7.2. /usr.

UNIX es una plataforma muy popular para desarrollar programas de C/C++, Java, Perl, etc. Así el administrador del sistema modifica y recompila el kernel. A causa de esto, compiladores[29], librerías[30] son tratados en este directorio.

### 2.7.3. /usr/local.

Todo el software que es instalado después de la instalación del sistema operativo, los archivos binarios serán ubicados en /usr/local/bin. (En la mayoría de los casos /usr/local/bin es incluido en la variable PATH[31] de los usuarios) /usr/local contiene múltiples directorios de los cuáles destacan tres directorios estándar bin, lib y src, que contienen todos aquellos elementos necesarios para el buen funcionamiento de los programas. En una situación de respaldo es conveniente respaldar /usr, de esta forma respalda todo el árbol que depende de él (por ejemplo: /usr/src, /usr/bin, etc.).

### 2.7.4. /usr/include.

Contiene cabeceras de archivos estándar de C/C++, este directorio es utilizado en archivos de configuración como Makefile, como el directorio include primario en Makefiles. (Un archivo Makefile es especial porque es procesado por el programa make[32] con el objetivo de compilar, ligar e instalar programas.)

### 2.7.5. /usr/lib.

Contiene librerías para otros programas por ejemplo Perl y Tcl.

---

[29] Programa que lee líneas escritas en un lenguaje de programación (como Pascal), y las traduce a otro que pueda ejecutar la computadora. Los programas compilados se ejecutan más rápido que los interpretados, debido a que han sido completamente traducidos a lenguaje de máquina y no necesitan compartir memoria con el intérprete.

[30] Conjunto de programas, funciones, variables con un estándar determinado para ser utilizados por otras aplicaciones, con el objetivo principal de reutilizar código y optimizar programas.

[31] Variable de entorno, en la que se encuentran almacenados las rutas (PATH), de aquellos directorios a los que el usuario tiene acceso, de esta forma puede buscar y ejecutar comandos o programas ubicados en ellos sin necesidad de acceder a dicho directorio.

[32] El programa make determina qué partes de un programa necesitan ser compiladas y manda ejecutar los comandos que son necesarios para alcanzar dicho objetivo. Es necesario escribir un archivo denominado Makefile.

### 2.7.6. /usr/src.

Contiene archivos fuente para los paquetes instalados en el sistema.

### 2.7.7. /var/spool.

Este directorio es usado para almacenar grandes volúmenes de archivos temporales asociados con la impresora, correo, grupos de noticias (news), entre otros.

Este es un directorio que se debe cuidar constantemente, ya que si un dispositivo no está trabajando o un volumen grande de correo que ha sido enviado al sistema no se ha depurado, entonces se corre el riesgo de que gran parte del disco duro pueda ser consumido por archivos almacenados en esta ubicación generando una negación de servicio.

### 2.7.8. /etc.

Este es uno de los directorios más importantes del sistema pues, no sólo incluye al archivo de contraseñas (passwd, shadow), sino que contiene además varios archivos de configuración para el sistema (por ejemplo los archivos de red, X windows y otros).

Otros directorios como skel, X11 y rc.d[33] están dentro de /etc; /etc/skel, contiene la estructura de archivos del usuario que son creados en el directorio del usuario, cuando se crea, /etc/X11 contiene archivos de configuración para X Windows.

### 2.7.9. /var/log.

UNIX típicamente mantiene un área particular para colocar las bitácoras (logs), estos archivos contienen registros de eventos que se dan en nuestro sistema, este directorio es generalmente denominado /var/log. Aquí se alojan diversas clases de errores como: errores de acceso, errores de programas como httpd[34], errores de hardware, etc.

### 2.7.10. /proc.

La jerarquía del directorio /proc contiene archivos asociados con la ejecución del kernel. Esta estructura de archivos contiene información acerca del estado del sistema, recursos del sistema (por ejemplo, la cantidad de memoria usada, espacio swap[35] y recursos de CPU), información acerca de cada proceso así como información valiosa del sistema.

### 2.7.11. /dev.

Este es el directorio principal para ubicar a los archivos de dispositivos, como la unidad de cd, impresora, cinta, unidad de zip, etc.

---

[33] En el directorio /etc/rc.d están los shells scripts(servicios y programas de arranque) que son ejecutados por el proceso init siguiendo las instrucciones del archivo /etc/inittab, que para cada nivel de ejecución existe un directorio /etc/rcN.d, donde N es un número que representa su nivel de ejecución del sistema (systems run level) sobre el cual un sistema UNIX trabaja. Por ejemplo en linux en nivel 1 es para re – arranque desde la PROM, el nivel 2 es un ambiente multiusuario sin NFS, el nivel 3 es un ambiente multiusuario con NFS, el nivel 4 es para procesos del usuario (sin uso), el nivel 5 es para trabajar sobre un ambiente gráfico. Por ejemplo en el nivel 3 de arranque se ejecutan los shells scripts que están contenidos en el directorio /etc/rc3.d

[34] Es el programa encargado del servicio Web en un servidor.

[35] El espacio swap o de "intercambio" es conocido como memoria virtual. La diferencia entre la memoria real y la virtual es que está última utiliza espacio en el disco duro, en lugar de un dispositivo (SIMM, DIMM, RIMM) de memoria, de esta manera cuando la memoria real se agota, el sistema copia parte del contenido que está directamente en este espacio Swap a fin de poder realizar otras tareas.

## 2.8. Archivos binarios.

Los archivos que se ubican dentro de "bin" son archivos binarios o ejecutables. Los cuatro directorios bin más comunes en UNIX son:

- /bin
- /sbin
- /usr/bin
- /usr/local/bin

Todos éstos directorios son similares pero con distintos propósitos; la división de estos archivos binarios tiene varios objetivos, entre ellos el de facilitar los respaldos, administración y separación lógica.

### 2.8.1. /bin.

Este directorio debe estar disponible al momento en que el sistema arranca (bootea), ya que contiene programas que son usados durante el arranque del sistema. Este directorio contiene además shells y archivos básicos así como utilidades de texto (ls, pwd, cut, head, tail, ed, etc.). Idealmente, el directorio /bin debe contener pocos archivos como sea posible, de esta manera sería más fácil tomar una copia del directorio completo para recuperar el disco boot/root en caso de alguna contingencia.

### 2.8.2. /sbin.

sbin literalmente significa "Binarios del sistema" (system binaries). Este directorio contiene archivos que generalmente son usados por el super usuario (root), y no son accesibles para los usuarios normales, así la ruta /sbin se agrega al PATH del usuario root y no para los demás usuarios. (excepto aquellos usuarios que tengan privilegios de super usuario)

El directorio /sbin deberá contener scripts y aquellos programas esenciales para el sistema de administración, incluyendo los involucrados con la administración de usuarios, administración de discos, control de eventos del sistema (por ejemplo los programas restart y shutdown), así como los programas de red.

Una regla para una buena administración es: "si los usuarios necesitan ejecutar un programa de /sbin, entonces este programa ya no puede seguir ubicado en /sbin, debe cambiarse a /usr/bin".

### 2.8.3. /usr/bin.

Este directorio contiene los binarios del usuario, (Programas que el usuario ejecuta) esto incluye las aplicaciones comúnmente utilizadas por los usuarios, incluyendo editores (vi, pico, emac), clientes de correo, compiladores, juegos y varias aplicaciones de red. Todos los usuarios tendrán incluido la ruta /usr/bin por cada PATH de usuario.

### 2.8.4. /usr/local/bin.

En este directorio deberán situarse los programas que son instalados por el administrador del sistema. (que son programas adicionales al momento de instalar el sistema operativo.) La principal razón de hacer esta distinción es facilitar el respaldo de los programas durante una actualización del sistema[36], o para restaurar el sistema después de una inconsistencia del mismo.

[36] Cuando un software es obsoleto, el administrador del sistema debe cambiar a un software actual, asimismo cuando un software presenta problemas se debe de sustituirlo por otro más optimo.

## 2.9. Consideraciones en sistemas de archivos.

Cuando un sistema UNIX arranca una de las tareas que obligatoriamente ha de realizar es incorporar diferentes sistemas de archivos a la jerarquía de directorios UNIX; este proceso se llama montaje, y para realizarlo generalmente se utiliza el comando mount. Es obligatorio montar al menos un sistema de archivos durante el arranque, el sistema raíz ("/")[37], del que colgarán todos los demás.

Montar un sistema de archivos significa asociar un determinado nombre de directorio, denominado punto de montaje (mount point), con el sistema en cuestión, de forma que al utilizar dicha ruta se estará trabajando sobre el sistema de archivos que se ha asociado a ella. Para saber qué sistemas de archivos se han de montar en el arranque de la máquina, y bajo qué nombre de directorio, UNIX utiliza un determinado archivo; aunque su nombre depende del clon utilizado (/etc/vfstab en Solaris, /etc/fstab en Linux, etc.), su función – incluso su sintaxis – es siempre equivalente.

```
# mount
/dev/hda3 on / type ext2 (rw)
/dev/hda4 on /home type ext2 (rw)
none on /proc type proc (rw)
```

Cuando el sistema arranque, el archivo /etc/fstab indica que en /dev/hda3 se encuentra el sistema de archivos raíz, de tipo ext2 (el habitual en Linux), y que se ha de montar con las opciones predeterminadas. La segunda línea nos dice que /home es un sistema diferente del anterior, pero del mismo tipo y que se montará con las mismas opciones; finalmente, la tercera línea hace referencia al directorio /proc/, donde se encuentra un sistema de archivos especial que algunas versiones UNIX utilizan como interfaz entre estructuras de datos del núcleo y el espacio de usuario. Si cualquiera de las entradas anteriores fuera errónea, el sistema o bien no arrancaría o bien lo haría incorrectamente. Como una política de seguridad, el archivo /etc/fstab o sus equivalentes ha de ser modificable sólo por el usuario root.

Es importante comprender el sistema de archivos de UNIX, puesto que diferentes problemas radican en una gestión incorrecta del montaje de sistemas de archivos. Por ejemplo, algo muy habitual en un atacante o intruso que consigue privilegios de administrador en una máquina es instalar ciertas utilidades que le permitan seguir gozando de ese privilegio (por ejemplo: un rootkit o un shell setuid); si guarda el archivo setuid en cualquier directorio de nuestro sistema, su localización será muy rápida: un comando como find nos alertará de su presencia. En cambio, ¿qué sucede si el atacante utiliza una parte del sistema de archivos oculta?, cuando montamos un sistema bajo un nombre de directorio (que ya existía, por ejemplo /home/saul/directorio), todo lo que había en ese directorio desaparece de la vista, y es sustituido por el contenido del sistema montado; no volverá a estar accesible hasta que no desmontemos el sistema:

```
# ls /home/saul/directorio
ftp/ Gloria / lost+found/

# umount /home/saul/directorio
# ls /home/saul/directorio
#
```

El atacante puede desmontar una parte de la jerarquía de directorios, guardar ahí ciertos archivos, y volver a montar el sistema que había anteriormente. Localizar esos archivos puede ser complicado, no por motivos técnicos sino porque a muy poca gente se le ocurre hacerlo. El comando ncheck, existente en versiones UNIX antiguos, puede detectar estos archivos ocultos bajo un punto de montaje (mount point), si no disponemos de esta utilidad podemos buscar por

[37] Origen de la distribución del sistema de archivos.

Internet aplicaciones que consiguen lo mismo, o simplemente desmontar manualmente los sistemas (a excepción del raíz) y comprobar que no hay nada oculto bajo ellos.

El tema de desmontar sistemas de archivos también puede ocasionar algún dolor de cabeza a muchos administradores; aunque no se trata de algo estrictamente relativo a la seguridad, se va a comentar un problema típico que se podría considerar como una negación de servicio (no causada por un fallo de UNIX sino por el desconocimiento del administrador). En ocasiones, al intentar desmontar un sistema de archivos, se podrá observar el siguiente resultado:

```
# umount /home/
umount: /home: device is busy
#
```

En este caso existe un determinado proceso haciendo uso de recursos bajo ese nombre de directorio. Hasta que dicho proceso no termine, será imposible desmontar el sistema, es fácil determinar de qué proceso se trata mediante la orden fuser (y posteriormente eliminarlo con el comando kill).

Otro problema clásico de los sistemas de archivos viene de la necesidad de que en muchos entornos se puede permitir a los usuarios (sin privilegios) montar y desmontar sistemas de archivos (típicamente, discos flexibles o CD-ROMs). Por ejemplo, imaginemos un laboratorio de máquinas UNIX donde es deseable que todos los usuarios puedan acceder a la unidad de disco flexible, tanto para copiar prácticas realizadas en casa como para hacer una copia de las que se han hecho en el propio laboratorio (este es uno de los casos más frecuentes en cualquier organización).

UNIX permite dar una solución rápida a este problema, pero esta solución puede convertirse en una amenaza a la seguridad si no es implantada correctamente. Al tratar el tema de /etc/fstab se ha comentado el montaje con ciertas opciones predeterminadas, dichas opciones son "rw" (permite tanto la lectura como la escritura), "suid" (permite la existencia de archivos con atributos setuid), "dev" (permite la existencia de dispositivos), "exec" (permite la ejecución de binarios), "auto" (el sistema se monta automáticamente al arrancar o al utilizar mount -a), "nouse"" (sólo puede ser montado por el root) y "async" (entrada/salida sobre el dispositivo se realiza de forma asíncrona).

Estas opciones son más comunes para sistemas de archivos "normales", pero no para los que puedan montar los usuarios, si es necesario que un usuario sin privilegios pueda montar y desmontar cierto dispositivo, hemos de especificar la opción "user" en la entrada correspondiente de /etc/fstab.

Se puede utilizar "noauto" para que el sistema no se monte automáticamente en el arranque de la máquina (si esto sucediera, el root tendría que desmontar la unidad manualmente para que otros usuarios puedan montarla). Es importante que si permitimos a un usuario montar una unidad, entonces se debe hacer uso de "nodev", de forma que si en el sistema montado existen archivos de tipo dispositivo (por ejemplo, un archivo que haga referencia a nuestros discos duros) ese archivo sea ignorado; en caso contrario, cualquier usuario podría acceder directamente al hardware.

También es importante especificar "nosuid", de forma que se ignore el bit de setuid en cualquier archivo contenido en el sistema que el usuario monta: así se evita que con un simple shell con el bit setuid activo en un disco flexible el usuario consiga privilegios de administrador en el sistema. Incluso puede ser conveniente especificar "noexec", de forma que no se pueda ejecutar nada de lo que está en el dispositivo montado. Todas estas opciones ("noexec", "nosuid" y "nodev") en Linux se asumen al indicar "user", pero en otros sistemas UNIX quizás no, por lo que nunca está de más ponerlas explícitamente (consultar el manual en otros clones de UNIX para asegurarse del efecto de cada opción); de esta forma, si los usuarios pudieran montar por ejemplo la unidad de disco flexible, la línea correcta en /etc/fstab sería la siguiente:

```
# grep fd0 /etc/fstab
/dev/fd0 /floppy ext2 user, noauto, nodev, nosuid, noexec
#
```

Otro aspecto relacionado con el montaje de sistemas de archivos que puede afectar a nuestra seguridad es el uso de sistemas de archivos diferentes de raíz (/) bajo ciertos directorios, una elección incorrecta a la hora de elegir dónde montar sistemas puede causar ciertos problemas, sobre todo negaciones de servicio. Razón por el cuál, es recomendable montar dispositivos diferentes bajo todos y cada uno de los directorios sobre los que los usuarios tienen permiso de escritura; esto incluye el padre de sus SHOME, /tmp/ o /var/tmp/ (que puede ser un enlace a /tmp/). Con esto se consigue que si un usuario llena un disco, esto no afecte al resto del sistema: un disco lleno implica muchos problemas para la computadora (servidor[38]), desde correo electrónico que no se recibe, logs que no se registran, o una negación de servicio contra el resto de usuarios, que no podrán almacenar ninguna información debido a que esta lleno el disco duro.

Una configuración adecuada sería la siguiente:

```
# mount
/dev/hda1       on /        type ext2     (rw)
/dev/hda2       on /tmp     type ext2     (rw)
/dev/hdb1       on /home    type ext2     (rw)
none            on /proc    type proc     (rw)
#
```

Si un usuario lanza un ftp en background[39] desde su directorio SHOME[40], durante la noche - típico proceso que llena gran cantidad de disco -, en todo caso podrá afectar al resto de usuarios, pero nunca al sistema en global (correo, logs, root.); este tipo de problemas no suele ser ataques, sino más bien descuidos de los usuarios que no tienen en cuenta el espacio disponible antes de descargar archivos de forma no interactiva.

## 2.10. Conclusión.

A lo largo del capítulo se habló de que UNIX está formado por archivos, debido a ello se debe proteger dichos archivos en el sistema comenzando con los permisos y después con el uso de algunos programas que UNIX proporciona para la seguridad del servidor.

Si no se toman las correspondientes medidas para proteger los archivos del sistema y tampoco se hace uso de los archivos y programas que ofrece UNIX para dicho objetivo (/etc/passwd, /etc/hosts, /etc/services, /usr/bin/chmod, etc.) se corre el riesgo de que un atacante con el conocimiento de estos problemas podría consultar información privada, modificarla o hacerla inaccesible para el usuario dueño de la misma.

Si se emplean los permisos correctamente en los archivos de UNIX, y se explotan todos los programas que ofrece UNIX, se podrá proteger los archivos para alcanzar la seguridad interna del servidor y de esta manera proteger la información desde su origen, es decir desde el servidor.

La seguridad interna en un servidor UNIX es muy importante, porque mantiene la información con las mejores condiciones para su uso, para alcanzar la seguridad se debe generar los permisos

---

[38] Proporciona servicios a usuarios, por ejemplo: transferencia de archivos, correo electrónico, servidor Web, manejador de base de datos, etc.

[39] Es dejar un proceso ftp ejecutando. Después de lanzar el proceso en background el ftp seguirá ejecutándose sin interrupción hasta finalizar su tarea.

[40] Variable de entorno, que almacena el directorio del usuario, desde que inicia el shell del usuario cuando entra en el sistema.

correctos en los archivos, la mejor configuración y uso de los archivos del sistema, y también con el empleo de todos los programas de seguridad que ofrecen los sistemas UNIX.

En el próximo capítulo se explicará que son los archivos y programas para monitorear el sistema (sulog, messages, etc), cuál es su importancia y se estudiará como poder obtener información de dichos archivos y programas, asimismo se explicará la influencia o relación que tiene el uso de estos archivos para la protección y administración del sistema.

# CAPÍTULO 3.

# MONITOREO DEL SISTEMA.

## 3.1. Introducción.

Después de haber visto el papel que desempeñan los archivos en un servidor UNIX, ahora en este capítulo se estudiará una clase de archivos especiales conocidos como "Logs" que son archivos para registrar (monitorear) actividades de diversa índole en el sistema. Se estudiará según como podremos obtener información de ellos, el objetivo es aprender a usar y obtener los beneficios que ofrecen en cuanto la seguridad del servidor, explicando de igual forma los beneficios y las desventajas que traen consigo.

En este capítulo el objetivo será conocer que UNIX fue diseñado para proporcionar un tipo de archivos cuya función es de mantener informado a los administradores sobre las actividades que ocurren en nuestro servidor, desde cuestiones de Hardware hasta registros de control de:

- Accesos al sistema.
- Errores de programas (por ejemplo errores que registra el demonio[41] de httpd, tcpd[42], etc.).
- Reportes de errores al arrancar el sistema.

Estos archivos, están ubicados en /var/logs y mediante comandos o un editor de texto se puede obtener información váliosa del sistema, ya que con estos archivos permiten conocer como se llevan a cabo las actividades del sistema y como afectan al mismo, incluso mediante una correcta interpretación ayuda a prevenir problemas, o corregirlas.

También se analizará las desventajas a los que estos archivos están expuestos y de que manera podrían afectar al sistema con el objetivo de crear conciencia sobre estos problemas y en todo caso se planteará las medidas para dar solución a esta problemática.

## 3.2. Monitoreo del sistema.

Casi todas las actividades realizadas en un sistema UNIX son susceptibles de ser, en mayor o menor medida, registradas (monitoreo) desde las horas de acceso de cada usuario al sistema hasta las páginas Web más frecuentemente visitadas, pasando por los intentos fallidos de conexión, los programas ejecutados o incluso el tiempo de CPU que cada usuario consume. Ésta facilidad de UNIX para recoger información tiene unas ventajas inmediatas para la seguridad: es posible detectar un intento de ataque, así como también detectar usos indebidos de los recursos o actividades "sospechosas"; sin embargo, existen también desventajas, ya que la gran cantidad de información que potencialmente se registra puede ser aprovechada para crear negaciones de servicio o dicha cantidad de información puede hacer difícil detectar problemas por el volumen de datos a analizar.

Algo muy interesante de los archivos de log en UNIX es que la mayoría de ellos son simples archivos de texto, que se pueden visualizar con un comando cat[43]. Por una parte esto es bastante cómodo para el administrador del sistema, ya que no necesita de herramientas especiales para poder revisar los logs (existen utilidades para hacer esto, como swatch) e incluso puede programar shell scripts para comprobar los informes generados de forma automática, con comandos como awk, grep o sed. No obstante, el hecho de que estos archivos sean texto plano hace que un atacante oculte fácilmente ciertos registros modificando los archivos con cualquier editor de textos; esto implica que un administrador no debe confiar al 100% en lo que los informes de auditoría del sistema le digan. Para minimizar estos riesgos se pueden tomar diversas medidas, desde algunas quizás demasiado complejas para entornos habituales hasta otras más sencillas pero igualmente

[41] Un demonio en UNIX es un proceso que se ejecuta siempre, cuando esta activado, en espera de peticiones para atenderlos.
[42] El demonio de tcpd es el programa que atiende todas las peticiones del servicio de Tcp – Wrappers. Así como éste existen más programas conocidos como demonios que mantienen sus datos en bitácoras especiales.
[43] Comando de UNIX cuya función es desplegar línea por línea un archivo.

efectivas, como utilizar una máquina confiable para registrar información del sistema o incluso enviar los registros más importantes a una impresora.

## 3.3. El sistema de log en UNIX.

Algunos de los archivos de log de los que hablaremos a continuación son comunes en cualquier sistema, su localización, o incluso su formato, pueden variar entre diferentes versiones UNIX.

Dentro de UNIX hay dos grandes familias de sistemas: se trata de System V y BSD[44], la localización de archivos y ciertos comandos relativos a la auditoría de la máquina van a ser diferentes, razón por la cual es recomendable consultar las páginas del manual antes de configurar el sistema de auditoría en un equipo particular. La principal diferencia entre ellos es él denominado *"contabilidad del sistema"* (process accounting o simplemente accounting), cuya función es registrar todos los programas ejecutados por cada usuario; los informes generados en este proceso pueden llegar a ocupar un espacio considerable en disco (dependiendo del número de usuarios en nuestro sistema) por lo que sólo es recomendable en situaciones muy concretas, por ejemplo para detectar actividades sospechosas[45] en una máquina o para analizar el tiempo de CPU consumido. En los sistemas System V el process accounting está desactivado por defecto, se puede iniciar mediante /usr/lib/acct/startup, y para visualizar los informes se utiliza el comando acctcom. En la familia BSD los equivalentes a estas órdenes son accton y lastcomm.

## 3.4. El demonio syslogd.

El demonio syslogd (Syslog Daemon) se lanza automáticamente al arrancar un sistema UNIX, y es el encargado de guardar informes sobre el funcionamiento de la máquina. Recibe mensajes de las diferentes partes del sistema (núcleo, programas, etc.) y los envía y/o almacena en diferentes localizaciones, tanto locales como remotas, siguiendo un criterio definido en el archivo de configuración /etc/syslog.conf, donde se aplican las reglas a seguir para gestionar el almacenamiento de mensajes del sistema. Las líneas de este archivo que comienzan por el carácter "#" son comentarios, con lo cual son ignoradas de la misma forma que las líneas en blanco; si ocurriera un error al interpretar una de las líneas del archivo, se ignoraría la línea completa. Un ejemplo de archivo /etc/syslog.conf es el siguiente:

```
saul:~# cat /etc/syslog.conf
#ident "@(#)syslog.conf 1.4 96/10/11 SMI" /* SunOS 5.7 */
#
# Copyright (c) 19911993, by Sun Microsystems, Inc.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (``) names
# that match m4 reserved words. Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice /dev/console
*.err;kern.debug;daemon.notice;mail.crit /var/adm/messages
*.alert;kern.err;daemon.err operator
*.alert root
*.emerg *

# if a nonloghost machine chooses to have authentication messages
# sent to the loghost machine, uncomment out the following line:
```

[44] Ver cuadro sinóptico de la evolución de UNIX en el apéndice I.

[45] Actividades que no son normales para el sistema, por ejemplo: consumo en exceso de tiempo de procesador.

```
#auth.notice ifdef("LOGHOST", /var/log/authlog, @loghost)

mail.debug ifdef("LOGHOST", /var/log/syslog, @loghost)

#
# nonloghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#

ifdef("LOGHOST", ,
user.err  /dev/console
user.err  /var/adm/messages
user.alert 'root, operator'
user.emerg *
)
saul:~#
```

Cada regla del archivo tiene dos campos: un campo de selección y un campo de acción, separados por espacios o tabuladores. El campo de selección está formado a su vez de dos partes: la primera, del servicio que envía el mensaje y la última de su prioridad, separadas por un punto (".");
Ambas son indiferentes a mayúsculas y minúsculas. La parte del servicio contiene una de las siguientes palabras clave: auth, authpriv, cron, daemon, kern, lpr, mail, mark, news, security (equivalente a auth), syslog, user, uucp y desde local0 hasta local7. Esta parte especifica el "subsistema" que ha generado ese mensaje (por ejemplo, todos los programas relacionados con el correo generarán mensajes ligados al servicio de correo electrónico).

La prioridad está compuesta de uno de los siguientes términos, en orden ascendente: debug, info, notice, warning, warn (equivalente a warning), err, error (equivalente a err), crit, alert, emerg, y panic (equivalente a emerg). La prioridad define la importancia del mensaje almacenado. Todos los mensajes de la prioridad especificada y superiores son almacenados de acuerdo con la acción requerida.

Además de los términos mencionados hasta ahora, el demonio syslogd emplea los siguientes caracteres especiales:

- " * " (asterisco)
  Empleado como comodín para todas las prioridades y servicios anteriores, dependiendo de dónde son usados (antes o después del carácter de separación "."):

  ```
  # Guardar todos los mensajes del servicio mail en /var/adm/mail
  #
      mail.*                                          /var/adm/mail
  ```

- " " (blanco, espacio, nulo)
  Indica que no hay prioridad definida para el servicio de la línea almacenada.

- "," (coma)
  Con este carácter es posible especificar múltiples servicios con el mismo patrón de prioridad en una misma línea; es posible enumerar varios servicios:

  ```
  # Guardar todos los mensajes mail.info y news.info en
  # /var/adm/info
      mail,news.=info                                 /var/adm/info
  ```

- ";" (punto y coma)

Es posible dirigir los mensajes de varios servicios y prioridades a un mismo destino, separándolos por este carácter:

```
# Guardamos los mensajes de prioridad "info" y "notice"
# en el archivo /var/log/messages
    *.=info;*.=notice                                   /var/log/messages
```

- "=" (igual)
  Se almacenan los mensajes con la prioridad exacta especificada y no incluyendo las superiores:

```
# Guardar todos los mensajes críticos en /var/adm/critical
#
    *.=crit                                             /var/adm/critical
```

- "!" (exclamación)
  Preceder el campo de prioridad con un signo de exclamación sirve para ignorar todas las prioridades, teniendo la posibilidad de escoger entre la especificada (!=prioridad) y la especificada más todas las superiores (!prioridad). Cuando se usan conjuntamente los caracteres "=" y "!", el signo de exclamación "!" debe preceder obligatoriamente al signo igual "=", de esta forma: !=.

```
# Guardar mensajes del kernel de prioridad info, pero no de prioridad err y superiores
# Guardar mensajes de mail excepto los de prioridad info
    kern.info;kern.!err                                 /var/adm/kernelinfo
    mail.*;mail.!=info                                  /var/adm/mail
```

El campo de acción describe el destino de los mensajes, que puede ser:

- Un archivo plano. Normalmente los mensajes del sistema son almacenados en archivos planos. Dichos archivos han de estar especificados con la ruta de acceso completa (comenzando con "/").

  Podemos preceder cada entrada con el signo menos, "-", para omitir la sincronización del archivo (vaciado del buffer de memoria a disco). Aunque puede ocurrir que se pierda información si el sistema cae justo después de un intento de escritura en el archivo, utilizando este signo se puede conseguir una mejora importante en la velocidad, especialmente si se están ejecutando programas que mandan muchos mensajes al demonio syslogd.

```
    # Guardamos todos los mensajes de prioridad critica en "critical"
    #
        *.=crit                                         /var/adm/critical
```

- Un terminal (conocido también como consola)
  También tenemos la posibilidad de enviar los mensajes a terminales; de este modo podemos tener uno de los terminales virtuales que muchos sistemas UNIX ofrecen en su consola "dedicado" a listar los mensajes del sistema, que podrán ser consultados con solo cambiar a ese terminal:

```
# Enviar todos los mensajes a tty12 (ALT+F12 en Linux).
# y los mensajes críticos del núcleo a consola
#
    *.*                                                 /dev/tty12
    kern.crit                                           /dev/console
```

- Una tubería con nombre.

Algunas versiones de syslogd permiten enviar registros a archivos de tipo pipe[46] (|) anteponiendo el símbolo " | " al nombre del archivo; dicho archivo ha de ser creado antes de iniciar el demonio syslogd, mediante comandos como: mkfifo o mknod. Esto es útil para procesar los registros utilizando cualquier aplicación de UNIX.

Por ejemplo, la siguiente línea de /etc/syslog.conf enviaría todos los mensajes de cualquier prioridad a uno de estos archivos denominado /var/log/mififo:

```
# Enviamos todos los mensajes a la tubería con nombre /var/log/mififo
#
    *.*                                                    |/var/log/mififo
```

- Una máquina remota
  Se pueden enviar los mensajes del sistema a otra máquina, de manera a que sean almacenados remotamente. Esto es útil si tenemos una máquina segura, en la que se puede confiar, conectada a la red, ya que de esta manera se guardaría allí una copia de los mensajes del sistema y no podrían ser modificados en caso de que comprometan al servidor. Esto es especialmente útil para detectar usuarios "ocultos" en nuestro sistema (usuarios que han conseguido los suficientes privilegios para ocultar sus procesos o su conexión):

```
# Enviamos los mensajes de prioridad warning y superiores al
# archivo "syslog" y todos los mensajes (incluidos los
# anteriores) a la maquina "zelda.nintendo.com.mx"
#
    *.warn                                                 /usr/adm/syslog
    *.*                                                    @zelda.nintendo.com.mx
```

- usuarios del sistema (si están conectados)
  Se especifica la lista de usuarios que deben recibir un tipo de mensajes escribiendo su login, separados por comas:

```
# Enviamos los mensajes con la prioridad "alert" a los usuarios root y saul
#
    *.alert                                                root, saul
```

- Todos los usuarios que estén conectados.
  Los errores con una prioridad de emergencia se suelen enviar a todos los usuarios que estén conectados al sistema, para que tomen sus precauciones:

```
# Mostramos los mensajes urgentes a todos los usuarios conectados, mediante wall
#
    *.=emerg                                               *
```

## 3.5. Algunos archivos de log.

En función de la configuración del sistema de auditoría de cada equipo UNIX los eventos que sucedan en la máquina se registrarán en determinados archivos; aunque se puede accesar en cualquier archivo (incluso a través de la red o en dispositivos), existen ciertos archivos de registro *"habituales"* en los que se almacena información. A continuación se comentará los más comunes y la información que almacenan.

---

[46] Programa de UNIX que realiza algún proceso generado por un comando, el resultado de dicho programa se guarda y lo utiliza como entrada para el siguiente proceso que tomará ese resultado para efectuar su tarea, así puede continuar uniendo más programas, al finalizar la tubería muestra el contenido que se generó por el último comando que se ejecuto.

### 3.5.1. Syslog.

El archivo syslog (guardado en /var/adm/ o /var/log/) es quizás el archivo de log más importante del sistema; en él se guardan, en texto claro, mensajes relativos a la seguridad de la máquina, como los accesos o los intentos de acceso a ciertos servicios. No obstante, este archivo es escrito por el demonio syslogd, por lo que dependiendo del archivo de configuración es como realizará sus actividades. Al estar guardado en formato texto, se puede visualizar su contenido con el comando cat:

```
saul:/# cat /var/log/syslog
Mar 5 04:15 23 saul in.telnetd[11632]  connect from localhost
Mar 5 06:16 52 saul rpcbind  connect from 127 0 0 1 to getport(R )
Mar 5 06.16.53 saul last message repeated 3 times
Mar 5 06:35 08 saul rpcbind  connect from 127 0 0 1 to getport(R )
Mar 5 18.26.56 saul rpcbind  connect from 127.0 0 1 to getport(R )
Mar 5 18:28:47 saul last message repeated 1 time
Mar 5 18.32.43 saul rpcbind  connect from 127.0 0 1 to getport(R )
Mar 6 02:30 26 saul rpcbind  connect from 127 0 0 1 to getport(R )
Mar 6 03.31 37 saul rpcbind  connect from 127 0 0 1 to getport(R )
Mar 6 11:07 04 saul in.telnetd[14847]  connect from gloria
Mar 6 11:40 43 saul in.telnetd[14964]  connect from localhost
```

Fecha  Usuario  Servicio  Desde dónde se conectó.

### 3.5.2. Messages.

En este archivo de texto se almacenan datos "informativos" de ciertos programas, mensajes de baja o media prioridad destinados más a informar que a avisar de sucesos importantes, como información relativa al arranque de la máquina; no obstante, como sucedía con el archivo syslog, en función de /etc/syslog.conf se podrá especificar que todo tipo de datos se debe guardar. Para visualizarlo se utiliza el comando head:

```
saul:/# head /var/adm/messages
Dec 24 18:09:54 saul UNIX: SunOS Release 5.7 Version Generic
[UNIX(R) System V Release 4 0]
Dec 24 18 09:54 saul UNIX: Copyright (c) 19831998, Sun Microsystems, Inc.
Dec 24 18:09:54 saul UNIX: mem = 65152K (0x3fa0000)
Dec 24 18:09:54 saul UNIX: avail mem = 51167232
Dec 24 18:09:54 saul UNIX: root nexus = i86pc
Dec 24 18:09:54 saul UNIX: isa0 at root
Dec 24 18:09:54 saul UNIX: pci0 at root: space 0 offset 0
Dec 24 18:09:54 saul UNIX:      IDE device at targ 0, lun 0 lastlun 0x0
Dec 24 18:09:54 saul UNIX:      model WDC WD84AA, stat 50, err 0
```

Fecha y hora.  Usuario. S.O.  Información de los servicios.

### 3.5.3. Wtmp.

Este archivo es un archivo binario (no se puede leer su contenido directamente con comandos como cat o similares) que almacena información relativa a cada conexión y desconexión del sistema. Para ver su contenido se ejecuta un comando como last:

```
saul:/# last -10
toni     pts/11       localhost        Mon  Mar  6 11:07  -  11:07 (00:00)
toni     pts/11       saul             Sun  Mar  5 04:22  -  04:25 (00:03)
ftp      ftp          andercheran.aiin Sun  Mar  5 02:30     still logged in
ftp      ftp          andercheran.aiin Sun  Mar  5 00:28  -  02:30 (02:01)
ftp      ftp          saul             Thu  Mar  2 03:02  -  00:28 (2+21:25)
ftp      ftp          saul             Thu  Mar  2 03:01  -  03:02 (00:00)
ftp      ftp          localhost        Thu  Mar  2 02:35  -  03:01 (00:26)
root     console                       Thu  Mar  2 00:13     still logged in
reboot   system       boot             Thu  Mar  2 00:12
root     console                       Wed  Mar  1 06:18     down (17:54)
```

| Nombre De usuario | Servicio que solicita | Desde dónde accede | Fecha y hora de acceso. | Estado de la conexión. |
|---|---|---|---|---|

Los registros guardados en este archivo (y también en utmp) tienen el formato de la estructura utmp, que contiene información como el nombre de usuario, la línea por la que accede, el lugar desde donde lo hace y la hora de acceso[47].

### 3.5.4. Utmp.

El archivo utmp es un archivo binario con información de cada usuario que está conectado en un momento dado; el programa /bin/login genera un registro en este archivo cuando un usuario se conecta, mientras que init lo elimina cuando se desconecta. Aunque habitualmente este archivo está situado en /var/adm/, junto a otros archivos de log, es posible encontrar algunas versiones UNIX – los más antiguos – que lo sitúan en /etc/. Para visualizar el contenido de este archivo se ejecuta un comando como last (indicando el nombre de archivo mediante la opción f), w o who:

```
saul:/# who
root     console      Mar  2 00:13
root     pts/2        Mar  3 00:47  (UNIX)
root     pts/3        Mar  2 00:18  (UNIX)
root     pts/5        Mar  2 00:56  (UNIX)
root     pts/6        Mar  2 02:23  (UNIX:0.0)
root     pts/8        Mar  3 00:02  (UNIX:0.0)
root     pts/7        Mar  2 23:43  (UNIX:0.0)
root     pts/9        Mar  3 00:51  (UNIX)
root     pts/10       Mar  6 00:23  (UNIX)
```

| Nombre de usuario. | Puerto. | Fecha y hora. |
|---|---|---|

Como sucedía con wtmp, algunas versiones UNIX utilizan también una versión extendida de utmp (utmpx) con campos adicionales.

[47] se puede consultar la página de manual de funciones como getutent() para ver la estructura concreta en el clon de UNIX en el que trabajemos. Algunas variantes de UNIX (como Solaris o IRIX) utilizan un archivo wtmp extendido denominado wtmpx, con campos adicionales que proporcionan más información sobre cada conexión.

```
saul:/# cat /var/adm/loginlog
cris:        /dev/pts/6:     Thu Jan 6    07:02:53 2000
cris:        /dev/pts/6:     Thu Jan 6    07:03:00 2000
cris:        /dev/pts/6:     Thu Jan 6    07:03:08 2000
cris:        /dev/pts/6:     Thu Jan 6    07:03:37 2000
cris:        /dev/pts/6:     Thu Jan 6    07:03:44 2000
```

Nombre de usuario. Puerto de acceso Fecha. Hora.

### 3.5.5. Lastlog.

El archivo lastlog es un archivo binario (programa ejecutable) guardado generalmente en /var/adm/, y que contiene un registro para cada usuario con la fecha y hora de su última conexión; se puede visualizar estos datos mediante el comando finger:

```
saul:/# finger saul
Login name: saul                        In real life: saul at Saul Fabian Fabian
Directory: /export/home/saul                        Shell: /bin/sh
Last login Mon Mar  6 11:07   on   pts/11   from localhost
No unread mail
No Plan.
saul:/#
```

### 3.5.6. Faillog.

Este archivo es equivalente al anterior, pero en lugar de guardar información sobre la fecha y hora del último acceso al sistema, se guarda información del último intento de acceso de cada usuario; una conexión es fallida si el usuario (o alguien en su lugar) teclea incorrectamente su contraseña. Esta información se muestra la siguiente vez que dicho usuario entra correctamente a la máquina:

```
hosts login: saul
Password:
Linux 2.0.33.
1 failure since last login. Last was 14:39:41 on ttyp9.
Last login: Wed May 13 14:37:46 on ttyp9 from yoshi.nintendo.com.mx.
andercheran:~$
```

### 3.5.7. Loginlog.

Si en algunas versiones de UNIX (como Solaris) creamos el archivo /var/adm/loginlog (que originalmente no existe), se registrarán en él los intentos fallidos de login, siempre y cuando se produzcan cinco o más de ellos seguidos:

### 3.5.8. Btmp.

En algunos clones de UNIX, como Linux o HP-UX, el archivo btmp se utiliza para registrar las conexiones fallidas al sistema, con un formato similar al que wtmp utiliza para las conexiones que han tenido éxito:

```
zelda:~# last -f /var/adm/btmp |head -7
pnvarro      ttyq1      term104.nintendo.com      Wed Feb 9  16:27  15:38  (23:11)
jomonra      ttyq2      deportes.nintendo.com.mx  Fri  Feb 4 14:27   09:37  (9+19:09)
PNAVARRO     ttyq4      term69. nintendo com mx   Wed Feb 2  12:56   13:09  (20+00:12)
panavarr     ttyq2      term180. nintendo.com.mx  Fri   Jan 28 12:45  14:27  (7+01:42)
vbarbera     ttyp0      daind03. nintendo.com.mx  Thu  Jan 27 20:17   still logged in
pangel       ttyq1      agarcia2. nintendo com.mx Thu  Jan 27 18:51   16:27  (12+21:36)
abarra       ttyp0      dtra51. nintendo com.mx   Thu  Jan 27 18:42   20:17  (01:34)
```

Usuario. Puerto. Terminal y dominio. Fecha y hora de acceso fallido.

### 3.5.9. Sulog.

Este es un archivo de texto, donde se registran las ejecuciones del comando *"su"*, indicando fecha, hora, usuario que lanza el programa y usuario cuya identidad adopta, terminal asociada y éxito ("+") o fracaso ("-") de la operación:

```
saul:/# head 4 /var/adm/sulog
SU          12/27   07:41          + console root:cris
SU          12/28   23:42          - vt01 crisroot
SU          12/28   23:43          + vt01 crisroot
SU          12/29   01:09          + vt04 crisroot
```

Programa "su". Fecha y hora de ejecución. Estado de éxito o fallo.

### 3.5.10. Debug.

En este archivo de texto se registra información de depuración (de debug) de los programas que se ejecutan en la máquina; esta información puede ser enviada por las propias aplicaciones o por el núcleo del sistema operativo:

```
saul:~# tail 8 /var/adm/debug
Dec 17 18:51:50 gloria kernel: ISO9660          Extensions: RRIP_1991A
Dec 18 08:15:32 gloria sshd[3951]: debug: sshd version 1.2.21 [i486-unknown-linux]
Dec 18 08:15:32 gloria sshd[3951]: debug: Initializing random number generator.
Dec 18 08:15:32 gloria sshd[3951]: debug: inetd sockets after dupping: 7, 8
Dec 18 08:15:34 gloria sshd[3951]: debug:Client protocol version 1.5; client software version 1.2.21
Dec 18 08:15:34 gloria sshd[3951]: debug: Calling cleanup 0x800cf90(0x0)
Dec 18 16:33:59 gloria kernel: VFS: Disk change detected on device 02:00
Dec 18 23:41:12 gloria identd[2268]: Successful lookup: 1593 , 22 : cris.users
```

Fecha y hora. Usuario. Información de depuración.

## 3.6. Logs remotos.

El demonio syslog permite fácilmente guardar registros en máquinas remotas; de esta forma se pretende que, aunque la seguridad de un sistema se vea comprometida y sus logs sean modificados se puedan seguir registrando las actividades sospechosas en una máquina a priori segura. Esto se consigue definiendo un "LOGHOST" en lugar de un archivo normal en el archivo /etc/syslogd.conf de la máquina en la que se guardará información; por ejemplo, si queremos registrar toda la información de prioridad info y notice en la máquina remota saul, se especificará de la siguiente forma:

```
*.=info;*.=notice                                    @saul
```

Tras modificar /etc/syslogd.conf, el demonio tiene que volver a leer su archivo de configuración enviándole la señal sighup (por ejemplo, con kill). Por otra parte, se debe especificar el host donde se desea almacenar los logs, para ello se debe definir el puerto syslog en /etc/services y ejecutar syslogd con el parámetro "r" para que acepte conexiones a través de la red:

```
saul:~# grep syslog /etc/services
        syslog                                                  514/udp

saul:~# ps xua|grep syslogd
root    41      0.0     0.4     852     304 ?   S       Mar21   0:01    /usr/sbin/syslogd

saul:~# kill TERM 41
saul:~# syslogd r
saul:~#
```

A partir de ese momento todos los mensajes generados en la máquina origen se enviarán a la destino y se registrarán según las reglas de esta, en un archivo, en un dispositivo. Incluso se reenviarán a otra máquina (en este caso hay que tener cuidado con los bucles).

```
saul:~# tail 3 /var/adm/messages
Mar     23      07:43:37        gloria  syslogd         1.3-3: restart.
Mar     23      07:43:46        gloria  in.telnetd[7509]: connect from amparo
Mar     23      07:57:44        gloria  MARK
saul:~#
```

Esto, que en muchas situaciones es muy recomendable, si no se realiza correctamente puede incluso comprometer la seguridad de la máquina que guarda registros en otro equipo: por defecto, el tráfico se realiza en texto claro, por lo que cualquier atacante con un sniffer entre las dos máquinas puede tener acceso a información importante que habría que mantener en secreto, por ejemplo, en una situación muy habitual: un usuario que teclea su password cuando el sistema le pide el login. Evidentemente, esto generará un mensaje de error que syslogd registrará, este mensaje será similar a este:

```
Mar     23      05:56:56        gloria login[6997]: invalid password for "UNKNOWN"\
on 'ttyp5' from 'amparo'
```

Si en lugar de "UNKNOWN" el sistema almacenara el nombre de usuario que se ha introducido, en esta situación el mensaje seria muy parecido al siguiente:

```
Mar     23      05:59:15        saul    login[3582]: FAILED LOGIN 1 FROM amparo FOR\
5k4@b&-, User not known to the underlying authentication module
```

Como se puede observar, se registraría una contraseña de usuario, contraseña que se envía a la máquina remota en texto claro a través de la red; evidentemente, es un riesgo que no se puede correr. Quizás alguien pueda pensar que una clave por sí sola no representa mucho peligro, ya que el atacante no conoce el nombre de usuario en el sistema, lo cierto es que el pirata sólo tiene que esperar unos instantes, porque cuando el usuario teclee su login y su password correctamente (en principio, esto sucederá poco después de equivocarse, porque el usuario trata de acceder a su cuenta) el sistema generará un mensaje indicando que ese usuario (con su nombre) ha entrado al sistema.

Para evitar este problema existen dos soluciones: se puede registrar logs en un equipo directamente conectado al nuestro, sin emitir tráfico al resto de la red, o bien utilizar

comunicaciones capaces de codificar la información (por ejemplo con ssh) para enviar los registros a otra computadora.

- En el primer caso sólo se necesita un equipo con dos tarjetas de red, una por donde enviar el tráfico hacia la red local y la otra para conectar con la máquina donde almacena los logs, que sólo será accesible desde nuestro equipo y que no ha de tener usuarios ni ofrecer servicios, no es necesaria una gran potencia de procesador: se puede aprovechar un procesador Pentium 386 o 486 con un sistema operativo Linux para esta tarea.

- El segundo caso, utilizar comunicaciones cifradas para guardar registros en otro equipo de la red, requiere algo más de trabajo; aquí no es estrictamente necesario que la máquina esté aislada del resto de la red, ya que la transferencia de información se va a realizar de forma codificada, consiguiendo que un potencial atacante no obtenga ningún dato comprometedor analizando el tráfico; evidentemente, aunque no esté aislado, es fundamental que el sistema donde se almacenan los logs sea seguro. Para enviar un log codificado a una máquina remota podemos utilizar ssh, junto con las ventajas que ofrece syslogd, si lo hacemos así, lo único que necesitamos es el servidor sshd en la máquina destino y el cliente ssh en la máquina origen. Por ejemplo, si se utiliza al host *"saul"* para almacenar una copia de los registros generados en el host *"gloria"* conforme se vayan produciendo; en este caso se enviará logs a un fifo con nombre, desde donde se codificará con ssh y se enviarán al sistema remoto a través de la red. Lo primero que necesitamos hacer es crear un archivo de tipo tubería en la máquina origen, por ejemplo con mknod o mkfifo:

  ```
  gloria:~# mknod /var/run/cifra p
  gloria:~#
  ```

Este es el archivo al que se enviará desde syslogd los registros que sean de interés, por ejemplo los de prioridad warn; hemos de modificar /etc/syslog.conf para añadirle una línea como la siguiente:

```
gloria:~# tail -1 /etc/syslog.conf
        *.warn                                          |/var/run/cifra
gloria:~#
```

A continuación se hará que syslog relea su nueva configuración mediante la señal sighup:

```
gloria:~# ps xua|grep syslog |grep v grep
root     7978   0.0    0.2     1372   156 ?   S        03:01 0:00      syslogd  -m    0
gloria:~# kill HUP 7978
gloria:~#
```

Una vez realizados estos pasos, se consigue que se registren los eventos de interés en el archivo /var/run/cifra; este archivo es una tubería con nombre, de forma que los datos que se envían no se graban en el disco realmente, sino que sólo esperan a que un proceso lector los recoja. Ese proceso lector será justamente el cliente ssh, encargado de codificarlos y enviarlos al sistema remoto; para ello se debe ejecutar el siguiente comando:

```
gloria:~# cat /var/run/cifra | ssh x saul 'cat >>/var/log/gloria'
```

Si se tiene configurado ssh para que autentique sin clave, se podrá lanzar el proceso directamente en background, si se tiene que introducir la clave del root, una vez tecleada se puede parar el proceso y relanzarlo también en segundo plano (esto es simplemente por comodidad, realmente no es necesario). Con este mecanismo logramos codificar lo que llega al fifo y enviarlo al sistema remoto, en el que se descodificará y se guardará en el archivo /var/log/gloria.

Quizás podría ser interesante añadir unas líneas en los scripts de arranque del sistema operativo para que este proceso se lance automáticamente al iniciar el sistema, se ser así se tiene que tener cuidado con la autenticación, ya que si ssh requiere una clave para conectar con el sistema remoto es probable que la máquina tarde más de lo normal en arrancar si un operador no está en la consola: Justamente el tiempo necesario hasta que ssh produzca un timeout por no teclear el password de root en el sistema remoto.

Si al producirse el timeout el programa ssh no devuelve el control al shell, el sistema ni siquiera arrancará, de cualquier forma, si ese timeout se produce no se registrará ningún evento en la otra máquina. Por supuesto, también hay que prestar atención a otros problemas con la máquina destino que eviten que la conexión se produzca, con un número máximo de usuarios sobrepasado o que ese sistema esté apagado.

## 3.7. Registros físicos.

Para asegurar que la información que se registra de las actividades en nuestro sistema es fiable ( se ha explicado cómo almacenarla), ya sea en un archivo de la propia máquina, o en un equipo remoto a través de la red; la idea es poder comparar los registros de ambos sistemas para detectar posibles modificaciones en una de ellas.

Si el atacante consigue también control sobre el segundo equipo, y modifica también ahí los archivos de log, en algunos casos (por ejemplo si sospechamos que el intruso ha conseguido el control de ambos equipos) es conveniente recurrir a registros físicos, mucho más difíciles de alterar. No siempre se guarda información en un archivo plano, ya sea local o remoto. UNIX permite almacenar mensajes en archivos especiales – dispositivos –, como terminales o impresoras; son estas últimas las más habituales por la seguridad que ofrecen, ya que mientras que un intruso con el privilegio suficiente puede modificar un archivo de log local, o acceder a un sistema donde se almacenen registros remotos, no puede eliminar una información extraída por impresora sin tener acceso físico a la misma.

El demonio syslog de cualquier sistema UNIX permite especificar uno de estos archivos especiales como destinatario de ciertos registros de una forma muy simple: agregar una línea en /etc/syslog.conf indicando el dispositivo y la clase de eventos a registrar en él; por ejemplo, para enviar todos los mensajes de prioridad warn a una impresora (como /dev/lp1) se debe añadir en el archivo de configuración la línea siguiente:

```
*.warn                                  /dev/lp1
```

Como siempre, tras modificar el archivo de configuración se debe hacer que el demonio vuelva a leer el archivo, bien enviándole la señal sighup o deteniéndolo y volviéndolo a lanzar; por último, si se decide utilizar una impresora para generar registros físicos se recomienda que se trate de un modelo de agujas, ya que dispositivos más modernos utilizan buffers que no se llegan a imprimir hasta que están llenos, por lo que sería posible para un atacante hacer que se pierda cierta información.

Se debe evitar especialmente algunos modelos nuevos de impresoras que tienen incluso sistema de red y dirección IP para control remoto, ya que en este caso puede suceder que un pirata llegue a controlar el dispositivo igual que controla la máquina que envía los registros.

Otro tipo de registro físico, pero por desgracia no se suele utilizar mucho, son las propias anotaciones que todo administrador debería realizar en una especie de "cuaderno de bitácora" de cada equipo. Evidentemente la única persona con acceso a dicho cuaderno debería ser el administrador, y debería guardarse en un lugar seguro, aplicando las mismas políticas que por ejemplo se aplican a las cintas de backup (alejadas del entorno de operaciones para prevenir la pérdida ante un desastre físico, y almacenadas bajo llave).

## 3.8. Conclusión.

Como ya se analizó a lo largo del capítulo UNIX proporciona archivos cuya función es monitorear todas las actividades que llevan a cabo en un servidor, mediante el uso de estos archivos se puede saber que ocurre en nuestro sistema por lo cuál ayudará a tomar las precausiones necesarias para enfrentar sucesos que puedan afectar al sistema.

Si no se hace uso de estos archivos de monitoreo no se podrá saber qué sucesos ocurren en el sistema y en caso de sufrir un ataque no se tendrá ningún elemento para identificar el ataque, ni tampoco se podrá justificar porque detenemos el sistema al solucionar un problema relacionado con la seguridad del mismo.

Si se revisa frecuentemente los archivos de monitoreo del sistema, entonces se tendrá al menos la oportunidad de identificar un suceso anormal en el sistema y así implementar inmediatamente todas las medidas necesarias para enfrentar cualquier contingencia que pueda sufrir la información.

El uso de los archivos de monitoreo son de gran ayuda porque mediante la información que proporcionan permite tomar las medidas necesarias para el buen funcionamiento del sistema y mantener segura la información, asimismo los archivos de monitoreo proporcionan los elementos suficientes para identificar si el sistema atraviesa por un ataque o es un problema normal del sistema.

En el próximo capítulo se explicarán los algoritmos para codificar la información, cuál es la importancia de usarlos y su relación con la seguridad de la información dentro del servidor así como en canales inseguros.

# CAPÍTULO 4.

# CODIFICACIÓN DE LA INFORMACIÓN (CRIPTOGRAFÍA).

## 4.1. Introducción.

Una vez que se ha explicado las ventajas y desventajas para un buen mantenimiento del sistema de archivos y conscientes de que la información que contienen dichos archivos, puede ser vistas por terceras personas, aún sin el consentimiento de su dueño, se debe comprender algunos conceptos de algoritmos para codificar información, los cuales se explican en este capítulo.

Una realidad es que existen varios mecanismos para que alguna persona pueda apoderarse de información de otros usuarios, esto significa que cualquier otro usuario con el conocimiento suficiente puede romper las medidas de seguridad y así obtener información dentro o fuera de la red local. En virtud de estas necesidades se han desarrollado técnicas para codificar la información y aunque un atacante puede obtener la información, le sea más difícil entenderlos incluso en el caso de que quisiera romper el algoritmo con que se codificó la información le costaría más trabajo y tiempo.

Por ello en este capítulo se estudiarán los algoritmos más usuales para codificar información, explicando las ventajas y desventajas que representan su uso, para ello se tratarán cuestiones como:

- Conceptos sobre la codificación de la Información (conocida como Criptografía).
- Los métodos, y en algunos casos los procedimientos, que utilizan al codificar la información.
- El costo sobre el rendimiento del equipo que implica ejecutar estos algoritmos.
- Las fortalezas y debilidades que tienen los algoritmos.
- La evolución y cambios que han experimentado estos algoritmos.

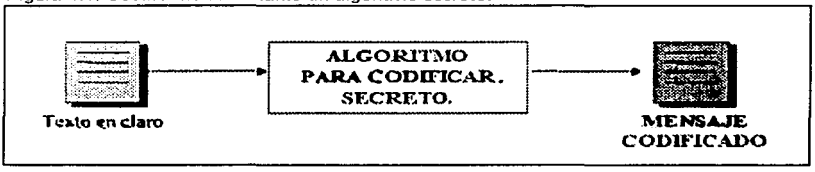
El objetivo es fundamentar los conceptos para codificar la información y en algunos casos con ejemplos prácticos y sencillos que ayudan a entender como operan dichos algoritmos; puesto que, en capítulos posteriores se estudiarán las herramientas y estas hacen uso de los algoritmos y nos proporcionarán una mejor idea para saber como trabajan las herramientas de seguridad.

## 4.2. CONCEPTOS BÁSICOS. (Criptología, Criptografía y Criptoanálisis).

Criptología quiere decir "escritura secreta"[48]. Actualmente tiene el significado de ciencia de la comunicación segura; su objetivo es codificar la información y de esta forma sólo quién conoce la clave podrá descodificar dicha información, así se puede asegurar la confiabilidad de la información dentro de un servidor o que dos partes puedan intercambiar información sin que una tercera parte no autorizada, a pesar de que capte los datos, sea capaz de descodificar la información.

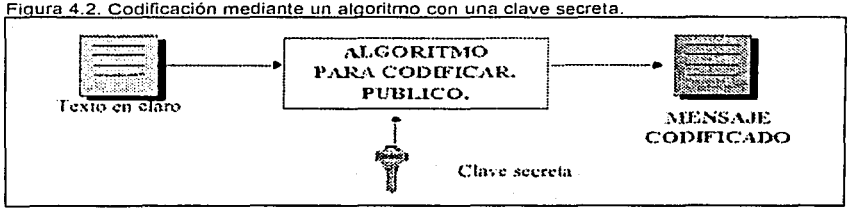
Para codificar (cifrar) se debe transformar un texto mediante un método, cuya función inversa sólo lo conocen las personas autorizadas. Así se puede utilizar un algoritmo secreto (Figura 4.1) o un algoritmo público que utiliza una palabra, llamada clave, sólo conocida por las personas autorizadas, esta clave debe ser imprescindible para la codificación y descodificación (Figura 4.2).

Figura 4.1. Codificación mediante un algoritmo secreto.




[48] En este trabajo manejaremos el término "codificación de la información" por "Cifrar la información".

Figura 4.2. Codificación mediante un algoritmo con una clave secreta.




La criptografía actúa mediante criptosistemas; un criptosistema o sistema de codificación es un sistema que permite codificar los mensajes de tal forma que una persona no autorizada no pueda descodificar el mensaje. La criptografía es la ciencia de diseñar criptosistemas. El criptoanálisis trata de romper los criptosistemas para apoderarse de la información codificada.

## 4.3. Criptografía llave privada.

El ser humano siempre ha tenido secretos de muy diversa índole, y ha buscado mecanismos para mantenerlos fuera del alcance de personas ajenas. Julio Cesar empleaba un sencillo algoritmo para evitar que sus comunicaciones militares fueran interceptadas. Leonardo Da Vinci escribía las anotaciones sobre sus trabajos de derecha a izquierda y con la mano zurda.

En este tema se realizará un breve repaso de los mecanismos criptográficos considerados clásicos. Se puede llamar así a todos los sistemas de codificación anteriores a la II Guerra Mundial, o lo que es lo mismo, al nacimiento de las computadoras. Estas técnicas tienen en común que pueden ser empleadas usando simplemente lápiz y papel, y que pueden ser criptoanalizadas casi de la misma forma. De hecho, con la ayuda de las computadoras, los mensajes codificados (cifrado) empleando estos códigos son fácilmente descodificados (descifrado), por lo que cayeron rápidamente en desuso.

La transición desde la criptografía clásica a la moderna se da precisamente durante la II Guerra Mundial, cuando el servicio de Inteligencia aliado[49] rompe la máquina para codificar mensajes del ejército alemán, llamada ENIGMA. Todos los algoritmos criptográficos clásicos son simétricos, ya que hasta mediados de los años setenta nació la criptografía asimétrica.

### 4.3.1. Algoritmos de codificación clásicos.

En esta sección se estudiarán algunos criptosistemas que en la actualidad han perdido su eficacia, debido a que son fácilmente criptoanalizables empleando cualquier computadora doméstica, pero que fueron empleados con éxito hasta principios del siglo XX. Algunos se remontan incluso, como el algoritmo de César, a la Roma Imperial. Sin embargo mantienen un interés teórico, permitiendo explotar algunas de sus propiedades para entender mejor los algoritmos modernos.

#### 4.3.1.1. Codificados monoalfabéticos.

Se engloban dentro de este apartado todos los algoritmos criptográficos que, sin desordenar los símbolos dentro del mensaje, establecen una correspondencia única para todos ellos en todo el texto. Es decir, si al símbolo *"A"* le corresponde el símbolo *"D"*, y se mantiene a lo largo de todo el mensaje.

[49] El servicio de inteligencia aliado desarrolló un mecanismo para descodificar los mensajes que la máquina enigma enviaba al ejercito alemán, de esta manera el ejército aliado podía saber que planes tenían los alemanes.

### 4.3.1.2. Algoritmo de cesar.

El algoritmo de Cesar, llamado así porque es el que empleaba Julio Cesar para enviar mensajes secretos, es uno de los algoritmos criptográficos más simples. Consiste en sumar 3 al número de orden de cada letra. De esta forma a *"la letra A"* le corresponde *"la letra D"*, a *"la letra B"* le corresponde *"la letra E"*, y así sucesivamente. Si asignamos a cada letra un número (A = 0, B = 1, etc.), y consideramos un alfabeto de 27 letras[50], la transformación para codificar sería:

C = (M + 3) mod 27.

Observe que este algoritmo ni siquiera posee clave, puesto que la transformación siempre es la misma. Para descodificar basta con restar 3 al número de orden de las letras del Criptograma.

Es el caso general del algoritmo de Cesar. Su transformación sería:

$$E_{(a,b)}(M) = (aM + b) \bmod N$$

Cuadro 4.1. Codificación de Cesar.

| | | | | | |
|---|---|---|---|---|---|
| A=0 | B=1 | C=2 | D=3 | E=4 | F=5 |
| G=6 | H=7 | I=8 | J=9 | K=10 | L=11 |
| M=12 | N=13 | N=14 | O=15 | P=16 | Q=17 |
| R=18 | S=19 | T=20 | U=21 | V=22 | W=23 |
| X=24 | Y=25 | Z=26 | | | |

En este ejemplo le asignamos un número a cada letra del alfabeto, luego aplicamos la función f(x) = a+b
Consiste en sumar el número asignado a una letra (a) y sumar b (la clave). Así tenemos por ejemplo:

Una clave = 3.
Un mensaje = "CESAR"

Asignamos los valores correspondientes del mensaje según muestra tabla, obteniendo:

| | | | | |
|---|---|---|---|---|
| 2 | 4 | 19 | 0 | 18 |

Aplicamos a cada número la función f(x) = a + 3 y obtenemos:

| | | | | |
|---|---|---|---|---|
| 5 | 7 | 22 | 3 | 21 |

Posteriormente sustituyendo cada valor por su equivalente sobre el alfabeto, obtenemos: FHVDU

Siendo a y b dos números enteros menores, que el cardinal N del alfabeto, y cumpliendo que mcd(a, N) = 1. La clave de codificado k viene entonces dada por el par (a, b). El algoritmo de Cesar será pues una transformación afín con k = (1, 3).

## 4.3.2. Algoritmos simétricos de codificación.

### 4.3.2.1. Codificación de producto.

La gran mayoría de los algoritmos de codificación simétricos se apoyan en los conceptos de confusión y difusión inicialmente propuestos por Shannon, que se combinan para dar lugar a la codificación de producto.

[50] Éste alfabeto es tomado por las letras que reconoce el teclado QWERTY.

La confusión consiste en tratar de ocultar la relación que existe entre el texto plano, el texto codificado y la clave; utilizando tablas para hacer sustituciones. Un buen mecanismo de confusión hará demasiado complicado extraer relaciones estadísticas entre las tres cosas. Por su parte la difusión trata de repartir la influencia de cada bit del mensaje original lo más posible entre el mensaje codificado.

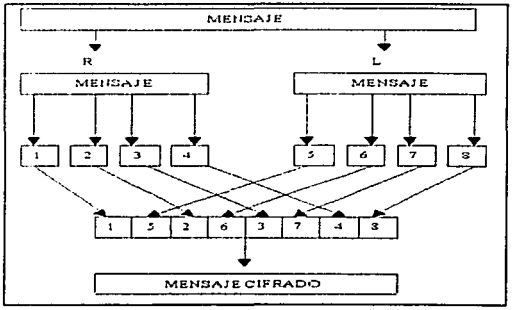
Lo que en realidad se hace para conseguir algoritmos fuertes sin necesidad de almacenar tablas enormes es intercalar la confusión (sustituciones simples, con tablas pequeñas) y la difusión (permutaciones); esta combinación se conoce como codificación de producto. La mayoría de los algoritmos se basan en diferentes capas de sustituciones y permutaciones, estructura que se denomina *"red de sustitución – permutación"*. En muchos casos el criptosistema es un paso simple de Sustitución–Permutación repetido *n* veces, como ocurre con el algoritmo DES.

### 4.3.2.2. Redes de feistel.

Las codificaciones de producto tienen en común que dividen un bloque de longitud *n* en dos mitades, *L* y *R*. Se define entonces una codificación de producto iterativo en el que la salida de cada ronda se usa como entrada para la siguiente.

Este tipo de estructura se denomina red de feistel, y es empleada en multitud de algoritmos, como des, lucifer, feal, cast, blowfish, etcétera. Tiene la propiedad de ser reversible, para ello basta con aplicar de nuevo el algoritmo al resultado, entonces nos va a permitir emplear el mismo mecanismo tanto para codificar como para descodificar.

Cuadro 4.2. Ejemplo del algoritmo de la Red de Feistel.



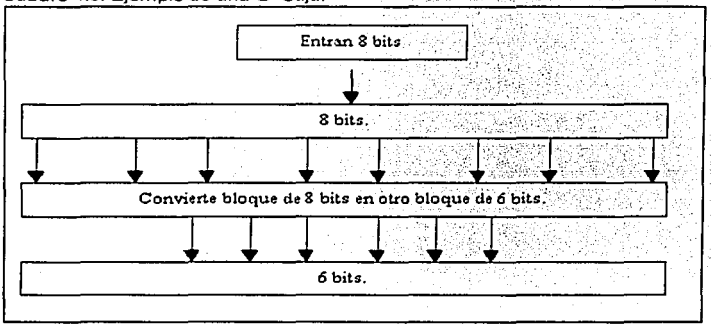

### 4 3.2.3. S-Cajas.

Para poder construir buenos algoritmos de producto, se intercala sustituciones sencillas (confusión), con tablas pequeñas, y permutaciones (difusión). Estas tablas pequeñas de sustitución se denominan de forma genérica S–Cajas.

Una S–Caja de m*n bits (ver cuadro 4.3) es una tabla de sustitución que toma como entrada cadenas de *m* bits y da como salida cadenas de *n* bits. El algoritmo DES, por ejemplo, emplea ocho S–Cajas de 6*4 bits. La utilización de las S–Cajas es la siguiente: se divide el bloque original en trozos de *m* bits y cada uno de ellos se sustituye por otro de *n* bits, obteniendo los valores correspondientes de la S–Caja correspondiente. Cuanto más grandes sean las S–Cajas, más resistente será el algoritmo resultante, aunque la elección de los valores de salida para que den

lugar a un buen algoritmo no es en absoluto sencillo porque depende de factores, por ejemplo, como el costo en procesador.

Cuadro 4.3. Ejemplo de una S–Caja.




Existe un algoritmo criptográfico, llamado CAST, que emplea seis S–Cajas de 8*32 bits. CAST codifica bloques de 64 bits, empleando claves de 64 bits, consta de ocho rondas y deposita prácticamente toda su fuerza en las S–Cajas. Existen muchas variedades de CAST, cada una con sus S–Cajas correspondientes "algunas de ellas secretas": este algoritmo se ha demostrado resistente a las técnicas habituales de criptoanálisis, y sólo se conoce la fuerza bruta como mecanismo para atacarlo.

### 4.3.3. El algoritmo DES.

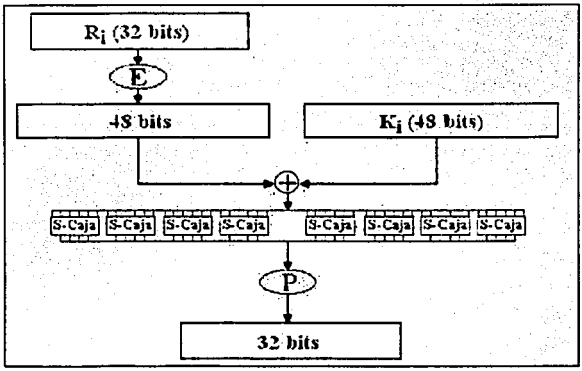
Es el algoritmo simétrico más extendido mundialmente. Data de mediados de los setenta, cuando fue adoptado como estándar para las comunicaciones seguras por el Gobierno de los EE.UU. La NSA[51] lo diseñó para ser implementado por hardware, con la intención de mantenerlo en secreto, pero al parecer por un malentendido entre ellos y la Oficina Nacional de Estandarización, su especificación se hizo pública con suficiente detalle para implementarlo por software.

A mediados de 1998, se demostró que un ataque por la fuerza bruta a DES era viable, debido a la escasa longitud que emplea en su clave. No obstante, el algoritmo aún no ha demostrado ninguna debilidad grave desde el punto de vista teórico, por lo que su estudio sigue siendo interesante. El algoritmo DES codifica bloques de 64 bits empleando claves de 56 bits. Es una Red de Feistel de 16 rondas, más dos permutaciones, una que se aplica al principio ($P_i$) y otra que se aplica al final ($P_f$).

[51] Agencia de Seguridad Nacional en los Estados Unidos de Norteamerica.

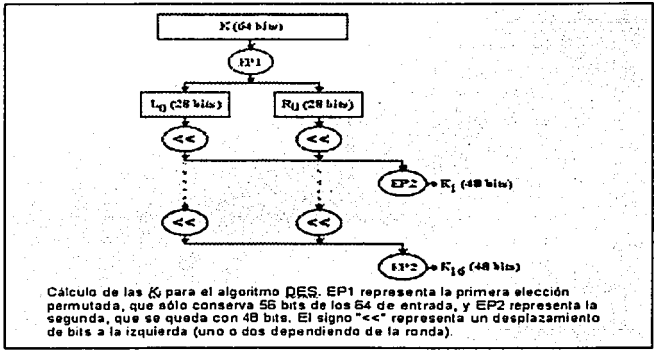
Se compone de una permutación de expansión*(E)*, que convierte el bloque de 32 bits correspondiente en uno de 48. Después realiza un *or-exclusivo* con el valor $K_i$, también de 48 bits, aplica ocho S-Cajas de 6 bits de entrada y 4 bits de salida, y efectúa una nueva permutación *P*.

Figura 4.3. Permutación de expansión.




Se calcula un total de 16 valores de $K_i$; uno para cada ronda, efectuando primero una permutación inicial EP1 sobre la clave de 64 bits, llevando a cabo desplazamientos a la izquierda de cada una de las dos mitades "de 28 bits" resultantes, y realizando finalmente una elección permutada (EP2) de 48 bits en cada ronda, que será la $K_i$. Los desplazamientos a la izquierda son de dos bits, salvo para las rondas 1, 2, 9 y 16, en las que se desplaza sólo un bit.

Figura 4.4. Algoritmo para calcular las 16 valores de ki.




Cálculo de las $K_i$ para el algoritmo DES. EP1 representa la primera elección permutada, que sólo conserva 56 bits de los 64 de entrada, y EP2 representa la segunda, que se queda con 48 bits. El signo "<<" representa un desplazamiento de bits a la izquierda (uno o dos dependiendo de la ronda).

Observe que aunque la clave para el algoritmo DES tiene en principio 64 bits, se ignoran ocho de ellos "un bit de paridad por cada byte de la clave", por lo que en la práctica se usan sólo 56 bits.

Para descodificar basta con usar el mismo algoritmo empleando las $K_i$ en orden inverso.

### 4.3.3.1. Variantes de DES.

A pesar de su caída, DES sigue siendo ampliamente utilizado en varias aplicaciones, como por ejemplo las transacciones de los cajeros automáticos. De todas formas, el problema real de DES no radica en su diseño, sino que emplea una clave muy corta (56 bits), lo cual hace que con el avance actual de las computadoras los ataques por la fuerza bruta comiencen a ser opciones realistas. Mucha gente se resiste a abandonar este algoritmo, precisamente porque ha sido capaz de sobrevivir durante veinte años sin mostrar ninguna debilidad en su diseño, prefieren proponer variantes como triple DES[52], que por un lado evitarían el riesgo de tener que confiar en algoritmos nuevos, y por el otro permitirían aprovechar gran parte de las implementaciones por hardware existentes de DES.

### 4.3.4. El algoritmo IDEA.

El algoritmo IDEA (International Data Encryption Algorithm) data del año de 1992. Para muchos constituye el mejor y más seguro algoritmo simétrico disponible en la actualidad. Trabaja con bloques de 64 bits de longitud y emplea una clave de 128 bits. Como en el caso de DES, se usa el mismo algoritmo tanto para codificar como para descodificar.

IDEA es un algoritmo bastante seguro, y hasta ahora se ha mostrado resistente a multitud de ataques, entre ellos el criptoanálisis diferencial. No presenta claves débiles[53], y su longitud de clave hace imposible en la práctica un ataque por la fuerza bruta.

Como sucede con todos los algoritmos simétricos de codificación por bloques, el algoritmo IDEA se basa en los conceptos de confusión y difusión, haciendo uso de los algoritmos para codificar información por bloques.

### 4.3.5. Modos de operación para algoritmos de codificación por bloques.

Estos métodos se usan para aplicar codificación por bloques a mensajes de gran longitud. En primer lugar, independientemente del método empleado para codificar, se debe tener en cuenta lo que ocurre cuando la longitud de la cadena que se quiere codificar no es un múltiplo exacto del tamaño de bloque. Entonces se tendrá que añadir información al final para que sí lo sea. El mecanismo más sencillo consiste en rellenar con ceros (o algún otro patrón) el último bloque que se codifica. El problema ahora consiste en saber cuando se descodifica, por dónde hay que cortar. Lo que se suele hacer es añadir como último byte del último bloque el número de bytes que se han añadido. Esto tiene el inconveniente de que si el tamaño original es múltiplo del bloque, por lo que hay que alargarlo con otro bloque entero.

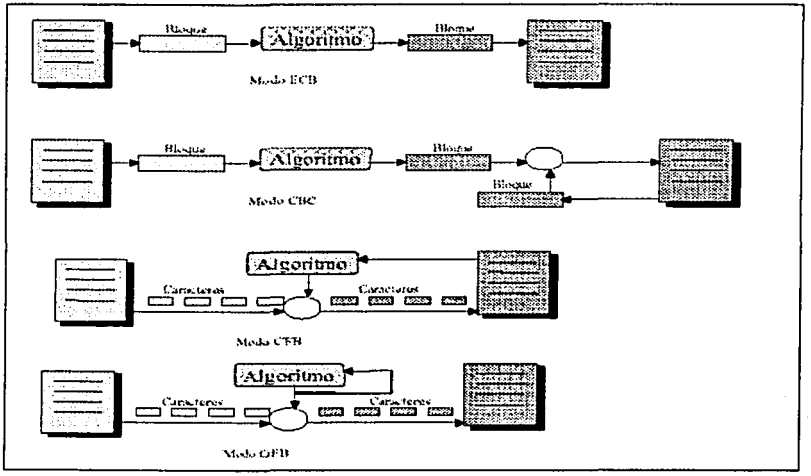
Los algoritmos simétricos codifican bloques de texto, el tamaño de los bloques puede ser constante o variable según el tipo de algoritmo. Tienen 4 modelos de algoritmos.

---

[52] El algoritmo "triple DES" es el mismo que el algoritmo "DES", solo que realiza 3 veces el algoritmo DES.

[53] En realidad, IDEA tiene un pequeñísimo subconjunto de claves que pueden dar ciertas ventajas a un criptoanalista, pero la probabilidad de encontrarnos con una de ellas es de 1 entre 2 a la potencia 96 , por lo que no representan un peligro real.

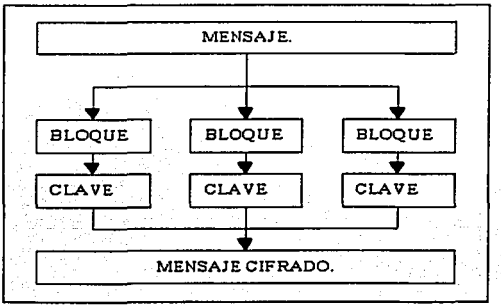
Figura 4.5. Modelos de algoritmos simétricos.




4.3.5.1. Modo ECB.

El modo ECB (Electronic Codebook) es el método más sencillo de aplicar un algoritmo de codificación por bloques. Se subdivide la cadena que se quiere codificar en bloques del tamaño adecuado y se codifican todos ellos empleando la misma clave. Una ventaja de este método es que permite codificar los bloques independientemente de su orden, lo cual es adecuado para codificar bases de datos o archivos en los que se requiera un acceso aleatorio. También es resistente a errores, pues si uno de los bloques sufriera una alteración, el resto quedaría intacto.

Figura 4.6. Ejemplo de ECB.



En caso contrario, si el mensaje presenta patrones repetitivos, el texto codificado también los presentará, y eso es peligroso, sobre todo cuando se codifica información muy redundante (como archivos de texto), o con patrones comunes al inicio y final (como el correo electrónico). Un contrincante puede en estos casos efectuar un ataque estadístico y extraer bastante información.

Otro riesgo bastante importante que presenta el modo ECB es el de la sustitución de bloques; el atacante puede cambiar un bloque sin mayores problemas, y alterar los mensajes incluso, si no conoce la clave y el algoritmo empleados.

#### 4.3.5.2. Modo CBC.

El modo CBC (Cipher Book Chaining Mode) incorpora un mecanismo de retroalimentación en la codificación por bloques, esto significa que la codificación de bloques anteriores condiciona la codificación del bloque actual, por lo que será imposible sustituir un bloque individual en el mensaje codificado. Se obtiene efectuando una operación XOR entre el bloque del mensaje que se quiere codificar y el último criptograma obtenido.

En cualquier caso, dos mensajes idénticos se codificarán de la misma forma usando el modo CBC, más aún, dos mensajes que empiecen igual, se codificarán similarmente hasta llegar a la primera diferencia entre ellos. Para evitar esto se emplea un vector de iniciación, que puede ser un bloque aleatorio, como bloque inicial de la transmisión, éste vector será descartado en destino, pero garantiza que siempre los mensajes se codifiquen de manera diferente, aunque tengan partes comunes.

#### 4.3.5.3. Modo CFB.

El modo CBC no inicia la codificación (o descodificación) hasta que no se tiene que transmitir (o se ha recibido) un bloque completo de información. Esta circunstancia puede convertirse en un serio inconveniente, por ejemplo en el caso de terminales, que deberían poder transmitir cada carácter que pulsa el usuario de manera individual. Una posible solución sería emplear un bloque completo para transmitir cada byte y rellenar el resto con ceros, pero esto hará que tengamos únicamente 256 mensajes diferentes en nuestra transmisión y que un atacante pueda efectuar un sencillo análisis estadístico para comprometerla. Otra opción sería rellenar el bloque con información aleatoria, aunque seguimos desperdiciando gran parte del ancho de banda de la transmisión. El modo de operación CFB (Cipher-Feedback Mode) permitirá codificar la información en unidades inferiores al tamaño del bloque, lo cual permite aprovechar totalmente la capacidad de transmisión del canal de comunicaciones, manteniendo además un nivel de seguridad adecuado.

#### 4.3.5.4. Otros modos.

Existen protocolos para codificar (protocolos criptográficos) que no solo se basan en la transmisión de bloques, sino también lo hacen en un mecanismo secuencial de codificación de streams de tamaño variable. Estos algoritmos permiten codificar un mensaje bit a bit de forma continua y enviar cada bit antes que el siguiente sea codificado. Funcionan a partir de lo que se llama un generador de secuencia de clave (keystream generator), un algoritmo que genera una clave continua de longitud muy grande, bit a bit. Lo que hace es aplicar una operación XOR[54] entre cada bit del texto claro y cada bit de la clave. En el destino existe otro generador idéntico sincronizado para llevar a cabo la descodificación. El problema fundamental es mantener ambos generadores sincronizados, para evitar errores si se pierde algún bit de la transmisión.

Los algoritmos de codificación por bloques pueden ser empleados como generadores de secuencia de clave. Existen para ello otros modos de operación de estos algoritmos, como el OFB (Output–

[54] Se puede ver un ejemplo de la operación XOR en el apéndice 2.

Feedback), que incorporan mecanismos para mantener la sincronía entre los generadores de secuencia origen y destino.

### 4.3.6. Criptoanálisis de algoritmos simétricos.

Se podría decir que el criptoanálisis se comenzó a estudiar seriamente con la aparición de DES. Mucha gente desconfiaba (y aún desconfía) del algoritmo propuesto por la NSA. Se dice que existen estructuras extrañas, que muchos consideran puertas traseras(bugs) colocadas por la Agencia para facilitarles la descodificación de los mensajes. Nadie ha podido aún demostrar ni desmentir este punto. Lo único cierto es que el interés por buscar posibles debilidades en él ha llevado a desarrollar técnicas que posteriormente han tenido éxito con otros algoritmos.

## 4.4. Criptografía de llave pública.

### 4.4.1. Algoritmos asimétricos de codificación.

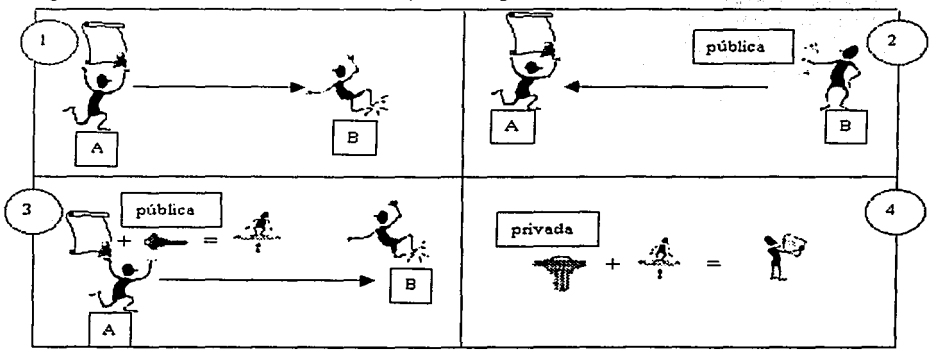
Los algoritmos de llave pública, o algoritmos asimétricos, han demostrado su interés para ser empleados en redes de comunicación inseguras (Internet). Introducidos por Whitéld Dié y Martin Hellman a mediados de los años 70, su novedad fundamental con respecto a la codificación de la información simétrica es que las claves no son únicas, sino que forman pares. Hasta la fecha han aparecido multitud de algoritmos asimétricos, la mayoría de los cuáles son inseguros. Otros son poco prácticos, porque el criptograma es considerablemente mayor que el mensaje original, porque la longitud de la clave es enorme. Se basan por general en plantear al atacante problemas matemáticos difíciles de resolver. En la práctica muy pocos algoritmos son realmente útiles. El más popular por su sencillez es RSA, que ha sobrevivido a multitud de ataques. Otros algoritmos son los de el Gamal y Rabin.

Los algoritmos asimétricos emplean generalmente longitudes de clave mucho mayores que los simétricos. Por ejemplo, mientras que para algoritmos simétricos se considera segura una clave de 128 bits, para algoritmos asimétricos se recomiendan claves de al menos 1024 bits. Además, la complejidad de cálculo que comportan estos últimos los hace considerablemente más lentos que los algoritmos de codificación por bloques. En la práctica los métodos asimétricos se emplean únicamente para codificar la clave de sesión (simétrica) de cada mensaje.

### 4.4.2. Aplicaciones de los algoritmos asimétricos.

Los algoritmos asimétricos poseen dos claves: $K_p$ y $K_P$, denominadas clave privada y clave pública. Una de ellas se emplea para codificar, mientras que la otra se usa para descodificar. Dependiendo de la aplicación que le demos al algoritmo, la clave pública será para codificar o viceversa. Para que estos criptosistemas sean seguros también ha de cumplirse que a partir de una de las claves resulte extremadamente difícil calcular la otra.

Figura 4.7. Transmisión de información empleando algoritmos asimétricos.




1. *"A"* tiene el mensaje *"m"* y quiere enviárselo a *"B"*.
2. *"B"* envía a *"A"* su clave pública, $K_P$.
3. *"A"* codifica el mensaje *"m"* y envía a *"B"* el criptograma.
4. *"B"* descodifica el criptograma empleando la clave privada $K_p$.

### 4.4.2.1. Protección de la información.

Una de las aplicaciones inmediatas de los algoritmos asimétricos es la codificación de la información sin tener que transmitir la clave de descodificación, lo cual permite su uso en canales inseguros, por ejemplo *"A"* quiere enviar un mensaje a *"B"*. Para ello solicita a *"B"* su clave pública $K_P$. *"A"* genera entonces el mensaje codificado *"E"*. Una vez hecho esto únicamente quien posea la clave privada $K_p$ (para el ejemplo es *"B"*) podrá recuperar el mensaje original *"m"*. Para este tipo de aplicación, la llave que se hace pública es aquella que permite codificar los mensajes, mientras que la llave privada es aquella que permite descodificarlos.

### 4.4.2.2. Autenticación.

La segunda aplicación de los algoritmos asimétricos es la autenticación de mensajes, con ayuda de funciones resumen, que nos permiten obtener una firma a partir de un mensaje. Dicha firma es mucho más pequeña que el mensaje original, y es muy difícil encontrar otro mensaje que tenga la misma firma. Por ejemplo *"A"* recibe un mensaje *"m"* de *"B"* y quiere comprobar su autenticidad. Para ello *"B"* genera un resumen del mensaje.

Muchos de los algoritmos asimétricos presentan claves duales, esto quiere decir que si empleamos una para codificar, la otra permitirá descodificar y viceversa. Esto ocurre con el algoritmo RSA, por lo cual un único par de claves es suficiente para codificar y autentificar.

### 4.4.3. Algoritmo RSA.

De todos los algoritmos asimétricos, quizá RSA[55] sea el más sencillo de comprender e implementar. Sus pares de claves son duales, por lo que sirve tanto para codificar como para autentificar. Su nombre proviene de sus tres inventores: Ron Rivest, Adi Shamir y Leonard Adleman. Desde su nacimiento nadie ha conseguido probar o rebatir su seguridad, lo cual lo tiene como uno de los algoritmos asimétricos más seguros.

RSA se basa en la dificultad para factorizar grandes números. Las claves públicas y privadas se calculan a partir de un número que se obtiene como producto de dos números primos grandes. El atacante se enfrentará, si quiere recuperar un mensaje a partir del criptograma y la llave pública, a un problema de factorización.

Para generar un par de llaves ($K_P$, $K_p$), en primer lugar se escogen aleatoriamente dos números primos grandes, $p$ y $q$. Después se calcula el producto $n = pq$.

En la práctica, se toma a dos números primos *"p* y *q"* con un número grande de bits, por ejemplo 200, con lo que $n$ tendrá 400 bits.

[55] En el apéndice número 3 de este trabajo, se explica un ejemplo sencillo para codificar y descodificar un mensaje, así como de los cálculos que ello implica.

Ejemplo del Funcionamiento de RSA.

1. Elegimos dos números primos a quienes llamaremos p y q. Al producto de ambos lo llamaremos n.
2. Elegimos un número e tal que sea relativamente primo a (p-1) * (q-1).
3. Encontramos ahora un número d tal que (e*d) –1 sea divisible entre (p-1)*(q-1).
4. Se puede probar utilizando los teoremas de Fermat y Euler, que para cualquier número m menor que n,
   $M^{e*d}$ mod n = m mod n = m.

   En nuestro ejemplo m sería el mensaje a codificar. Luego entonces:

   $m^e$ mod n = c.

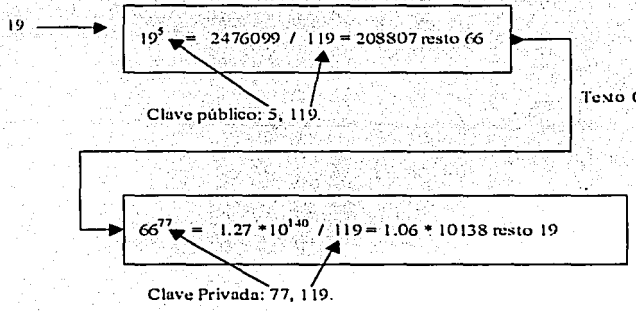
   Donde c es el mensaje codificado.

   C puede ser descifrado utilizando d de la siguiente forma:

   $c^d$ mod n = $m^{e*d}$ mod n = m mod n = m

   De esta forma tenemos un mecanismo para codificar y descodificar en el que:

   - e y d conforman la clave pública.
   - d y n conforman la clave privada.




El atacante, si quiere recuperar la clave privada a partir de la pública, debe conocer los factores *"p"* y *"q"* de *"n"*, y esto representa un problema computacionalmente intratable, siempre que *"p"* y *"q"* y, por lo tanto, *"n"* sean lo suficientemente grandes.

4.4.4. Algoritmo el gamal.

Fue diseñado en un principio para producir firmas digitales, pero posteriormente se extendió también para codificar mensajes. Se basa en el problema de los logaritmos discretos, que está

íntimamente relacionado con el de la factorización de números enteros. Para generar un par de llaves, se escoge un número primo *"p"* y dos números aleatorios *"g"* y *"x"* menores que *"p"*.

### 4.4.5. Algoritmo de rabin.

El sistema de llave asimétrica de Rabin se basa en el problema de calcular raíces cuadradas módulo un número compuesto. Este problema se ha demostrado que es equivalente al de la factorización de dicho número. En primer lugar escogemos dos números primos, $p$ y $q$, ambos congruentes con 3 módulo 4 (los dos últimos bits a 1). Estos primos son la clave privada. La clave pública es su producto $n = pq$.

### 4.4.6. Firmas digitales.

La Codificación de la información asimétrica permite autentificar información, es decir, poder asegurar que un mensaje *"m"* proviene de un emisor *"A"* y no de cualquier otro. Así mismo la autenticación debe hacerse empleando una función resumen y no codificando el mensaje completo. En esté capítulo analizaremos dichas funciones, que nos van a permitir crear firmas digitales.

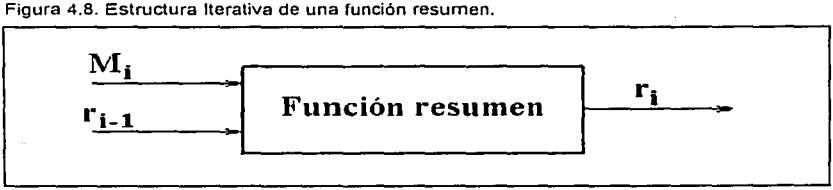
- Sabemos que un mensaje $m$ puede ser autentificado codificando con la llave privada $K_p$. El resultado de aplicarle una función resumen es generar información adicional (que se denomina firma o signatura del mensaje *"m"*) y sólo puede ser generada por el poseedor de la clave privada $K_p$. Cualquiera que tenga la llave pública correspondiente estará en condiciones de descodificar y verificar la firma.

Actualmente se recomienda emplear signaturas de al menos 128 bits, siendo 160 bits el valor más usado.

#### 4.4.6.1. Estructura de una función resumen.

En general, las funciones resumen se basan en la idea de funciones de compresión, que dan como resultado bloques de longitud $n$ a partir de bloques de longitud $m$. Estas funciones se encadenan de forma iterativa, haciendo que la entrada en el paso $i$ sea función del *i-ésimo* bloque del mensaje y de la salida del paso $i$ (figura 4.8). En general, se suele incluir en alguno de los bloques del mensaje *m "al principio o al final"*, información sobre la longitud total del mensaje. De esta forma se reducen las probabilidades de que dos mensajes con diferentes longitudes generen el mismo valor en su resumen.

Figura 4.8. Estructura Iterativa de una función resumen.




#### 4.4.6.2. Algoritmo MD5.

Se trata de uno de los más populares algoritmos de generación de firmas, debido a su inclusión en las primeras versiones de PGP. Resultado de una serie de mejoras sobre el algoritmo MD4, diseñado por Ron Rivest, procesa los mensajes de entrada en bloques de 512 bits, y produce una salida de 128 bits.

En los últimos tiempos el algoritmo MD5 ha mostrado ciertas debilidades, aunque sin implicaciones prácticas reales, por lo que se sigue considerando en la actualidad un algoritmo seguro, aunque su uso tiende a disminuir.

### 4.4.6.3. El algoritmo SHA.

El algoritmo SHA fue desarrollado por la NSA, para ser incluido en el estándar DSS (Digital Signature Standard). Al contrario de los algoritmos de codificación propuestos por esta organización, SHA se considera seguro y libre de puertas traseras, ya que favorece a los propios intereses de la NSA que el algoritmo sea totalmente seguro. Produce firmas de 160 bits, a partir de bloques de 512 bits del mensaje original.

## 4.4.7. Certificados X.509.

Un certificado es esencialmente una clave pública y un identificador, firmados digitalmente por una autoridad de certificación, y su utilidad es demostrar que una clave pública pertenece a un usuario concreto. El formato de certificados X.509 (Recomendación X.509 de CCITT, "The Directory Autentication Framework", 1988) es el más común y extendido en la actualidad. El estándar X.509 sólo define la sintaxis de los certificados, por lo que no está atado a ningún algoritmo en particular, y contempla los siguientes campos:

- Versión.
- Número de serie.
- Identificador del algoritmo empleado para la firma digital.
- Nombre del certificador.
- Periodo de validez.
- Nombre del sujeto.
- Clave pública del sujeto.
- Identificador único de certificador.
- Identificador único de sujeto.
- Extensiones.
- Firma digital de todo lo anterior generada por el certificador.

Estos certificados se estructuran de forma jerárquica, de tal forma que nosotros podemos verificar la autenticidad de un certificado comprobando la firma de la autoridad que lo emitió, que a su vez tendrá otro certificado expedido por otra autoridad de rango superior. De esta forma se escala en la jerarquía hasta llegar al nivel más alto, que deberá estar ocupado por un certificador que goce de la confianza de toda la comunidad[56]. Normalmente las claves públicas de los certificadores de mayor nivel se suelen publicar incluso en papel para que cualquiera pueda verificarlas.

El mecanismo que debe emplearse para conseguir un certificado X.509 es enviar nuestra clave pública "¡nunca la privada!" a la autoridad de certificación, después de habernos identificado positivamente frente a ella. Existen autoridades de certificación que, frente a una solicitud, generan un par llaves: *"pública – privada"* y lo envían al usuario. Hemos de hacer notar que en este caso, si bien tendremos un certificado válido, nuestro certificador podrá descodificar todos nuestros mensajes.

[56] La autoridad máxima es verisign en los Estados Unidos, tomando como punto de partida este dato, podemos mencionar un ejemplo de la siguiente manera: verising de Estados Unidos tiene todas las facultades para certificar a otro organismo como al CERT, y este organismo a su vez certifica a la Universidad Nacional Autónoma de México, esta a su vez certifica, finalmente, a los becarios de DGSCA. Estos últimos, mediante la certificación garantizan ser quién dicen ser.

## 4.5. PGP.

El nombre PGP responde a las siglas pretty good privacy (privacidad bastante buena), y se trata de un proyecto iniciado a principios de los 90 por Phill Zimmerman. Con el paso de los años, PGP se ha convertido en uno de los mecanismos más populares y fiables para mantener la seguridad y privacidad en las comunicaciones, especialmente a través del correo electrónico, tanto para pequeños usuarios como para grandes empresas. Hasta la fecha la política de distribución de PGP ha consistido en permitir su uso gratuito para usos no comerciales y en publicar el código fuente en su integridad, con el objetivo de que pueda ser estudiado.

Actualmente PGP se ha convertido en un estándar internacional (RFC 2440), lo cual está dando lugar a la aparición de múltiples productos PGP, que permiten desde codificar correo electrónico hasta codificar particiones enteras del disco duro (PGPDisk), pasando por la codificación automática y transparente de todo el tráfico TCP/IP (PGPnet).

### 4.5.1. Fundamentos e historia de PGP.

PGP trabaja con codificación de la información asimétrica, y por ello tal vez su punto más fuerte sea precisamente la gran facilidad que ofrece al usuario a la hora de gestionar sus claves públicas y privadas. Si uno emplea algoritmos asimétricos, debe poseer las claves públicas de todos sus interlocutores, además de la clave privada propia. Con PGP surge el concepto de anillo de claves (o llavero), que es el lugar que este programa proporciona para que el usuario guarde todas las claves que posee. El anillo de claves es un único archivo en el que se pueden efectuar operaciones de extracción e inserción de claves de manera sencilla, y que además proporciona un mecanismo de identificación y autenticación de llaves completo y simple de utilizar. Esta facilidad en la gestión de claves es una de las causas fundamentales que han hecho a PGP tan popular. Los PGP 2.x.x emplean los algoritmos IDEA, RSA y MD5.

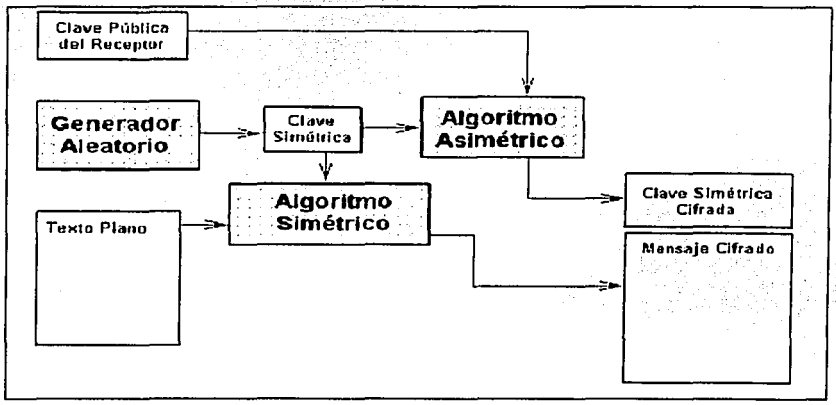
En algún momento una versión de PGP atravesó las fronteras de EE.UU. y nació la primera versión internacional de PGP, denominada PGPi, lo que le supuso a Phill Zimmermann una investigación de más de tres años por parte del FBI, ya que había violado las restrictivas leyes de exportación de material para codificación que poseen los Estados Unidos. Para la versión 5 de PGP se mejoró este problema exportando una versión impresa del código fuente, que luego era reconstruida y compilada en Europa (más información en http://www.pgpi.com).

### 4.5.2. Estructura de PGP.

#### 4.5.2.1. Codificación de mensajes.

Los algoritmos simétricos de codificación son considerablemente más rápidos que los asimétricos. Por esta razón PGP cifra primero el mensaje empleando un algoritmo simétrico (ver figura 4.9) con una clave generada aleatoriamente (clave de sesión) y posteriormente codifica la clave haciendo uso de la llave pública del destinatario. Dicha clave es extraída convenientemente del anillo de claves públicas a partir del identificador suministrado por el usuario, todo ello de forma transparente, por lo que únicamente se tiene que indicar el mensaje a codificar y la lista de identificadores de los destinatarios. Observe que para que el mensaje pueda ser leído por múltiples destinatarios basta que se incluya en la cabecera la clave de sesión codificada con cada una de las claves públicas correspondientes.

Figura 4.9. Codificación de un mensaje PGP.




Cuando se descodifica el mensaje, PGP busca en la cabecera las claves públicas con las que está codificado y pide una contraseña. La contraseña servirá para que PGP abra el anillo de claves privadas del propietario y compruebe si tiene una clave que permita descodificar el mensaje. En caso afirmativo, PGP descodificará el mensaje. Siempre se necesite hacer uso de una clave privada, se tiene que suministrar a PGP la contraseña correspondiente, por lo que si el anillo de claves privadas quedara comprometido, un atacante aún tendría que averiguar la contraseña para descodificar los mensajes. No obstante, si el anillo de claves privadas quedara comprometido, lo mejor será revocar todas las claves que tuviera almacenadas y generar otras nuevas.

Gran parte de la seguridad de PGP reside en la calidad del generador aleatorio que se emplea para generar las claves de sesión, puesto que si alguien logra predecir la secuencia de claves que se están usando, podrá descodificar todos los mensajes independientemente de los destinatarios a los que vayan dirigidos; por está razón PGP utiliza un método de generación de números pseudo – aleatorios muy seguro.

### 4.5.2.2. Firma digital.

En lo que se refiere a la firma digital, las primeras versiones de PGP obtienen en primer lugar la signatura MD5, que posteriormente se codifica empleando la clave privada RSA correspondiente. Las versiones actuales implementan el algoritmo DSA.

La firma digital o signatura puede ser añadida al archivo u obtenida en otro archivo aparte. Esta opción es muy útil, por ejemplo, si se quiere firmar un archivo ejecutable.

### 4.5.2.3. Armaduras ascii.

Una de las funcionalidades más útiles de PGP consiste en la posibilidad de generar una armadura ASCII para cualquiera de sus salidas. Todas las salidas de PGP (mensajes codificados, claves públicas extraídas de algún anillo, firmas digitales, etc.) consisten en secuencias binarias, que pueden ser almacenadas en archivos. Sin embargo, nos puede interesar enviar la información mediante correo electrónico, o almacenarla en archivos de texto.

Recordemos que el código ASCII es de 7 bits, eso quiere decir que los caracteres situados por encima del valor ASCII 127 no están definidos, de hecho diferentes computadoras y sistemas operativos los interpretan de manera distinta. Debemos considerar que entre los 128 caracteres ASCII se encuentran muchos que representan códigos de control, como el retorno de carro, el fin de archivo, el tabulador, etc.

La idea es elegir 64 caracteres imprimibles[57] (que no sean de control) dentro de esos 128 caracteres. Con los 64 caracteres escogidos podremos representar exactamente 6 bits, de esta manera una secuencia de tres bytes (24 bits) podrá representarse mediante cuatro de estos caracteres.

#### 4.5.2.4. Gestión de claves.

Como se menciono anteriormente, PGP almacena las claves en unas estructuras denominadas anillos. Un anillo es una colección de claves, almacenadas en un archivo. Cada usuario tendrá dos anillos, uno para las claves públicas (PUBRING.PKR) y otro para las privadas (SECRING.SKR).

Cada una de las claves, posee una serie de datos, como son el identificador del usuario que la emitió, la fecha de expiración, la versión de PGP con que fue generada, y la denominada huella digital (finger print). Este último campo es bastante útil, pues se trata de una secuencia hexadecimal lo suficientemente larga que lo hace ser única, y lo suficientemente corta como para que pueda ser escrita en un papel. La huella digital se emplea para asegurar la autenticidad de una clave. Por ejemplo, una huella digital de una clave pública es:

9E2B 9D14 CBCE FE12 16A8 C103 48B2 5161 69AB 5784

Si alguien quisiera asegurarse de la autenticidad de dicha clave, llamara por teléfono al autor, y le pedirá que lea su huella digital para verificar.

#### 4.5.2.5. Distribución de claves y redes de confianza.

PGP es susceptible a ataques de intermediario. Esto nos obliga a establecer mecanismos para asegurar que una clave procede realmente de quien creemos que es. Uno de los mecanismos que permite hacer esto es la huella digital.

PGP permite a un usuario firmar claves, y de esta forma se puede confiar en la autenticidad de una clave siempre que ésta venga firmada por una persona de confianza. Hay que distinguir entonces dos tipos de confianza: aquella que permite creer en la validez de una clave, y aquella que garantiza confiar de una persona como certificador de claves.

- La primera se puede calcular automáticamente, en función de que las firmas que contenga una clave pertenezcan a personas de confianza.

- la segunda ha de ser establecida manualmente. El hecho de que una clave sea auténtica no garantiza nada acerca de la persona que la emitió. Por ejemplo, se puede tener la seguridad de que una clave pertenece a una persona, pero esa persona puede dedicarse a firmar todas las claves que recibe, sin asegurarse de su autenticidad, razón por la cual ya no es confiable.

Cuando una clave queda comprometida, puede ser revocada por su autor. Para ello basta con generar y distribuir un certificado de revocación que informará a todos los usuarios de que esa clave ya no es válida. Para generarlo es necesaria la clave privada, razón por lo cual se

[57] Los únicos símbolos empleados son las letras mayúsculas y minúsculas, los números, y los signos "/" y "+"; el resto de símbolos y caracteres de control serán ignorados.

recomienda generar con cada clave su certificado de sustitución y guardarlo en lugar seguro, de forma que si perdemos la clave privada podremos sustituirla.

### 4.5.3. Otros PGP.

La rápida popularización de PGP entre ciertos sectores de la comunidad de Internet, y la disponibilidad del código fuente, han hecho posible la proliferación de variantes complejas de PGP. Muchas de ellas implementaban claves de mayor longitud (como PGPg), y otras corresponden a proyectos tan ambiciosos como GNU-PG (GNU Privacy Guard), que se basa únicamente en algoritmos de libre distribución.

### 4.5.4. Vulnerabilidades de PGP.

Como cualquier herramienta, PGP proporcionará un gran rendimiento si se emplea correctamente, pero su uso inadecuado lo dejaría inútil. Por ello retomamos los elementos más útiles de PGP.

- Elección de contraseñas adecuadas ya que un ataque por diccionario afecta a PGP.
- Protección de los archivos sensibles de PGP. Estos archivos son, nuestros llaveros (anillos de claves) y un archivo que alberga la semilla aleatoria, dicha protección debe llevarse a cabo frente al acceso de usuarios ajenos, como a una posible pérdida de los datos (si pierde el archivo con su clave privada no podrá descodificar jamás ningún mensaje).
- Emitir sustituciones de las claves al generarlas y guardarlas en lugar seguro. Serán el único mecanismo válido para sustituir una clave en caso de pérdida del anillo privado. La versión 6 de PGP permite nombrar sustitutos para las claves, de forma que éstos podrán invalidarla en cualquier momento sin necesidad de nuestra clave privada.
- Firmar sólo las claves de cuya autenticidad sea segura. Es la única manera de que las redes de confianza puedan funcionar, ya que si todos se dedican a firmar claves, se podría estar certificando claves falsas.

## 4.6. Implantación de algoritmos para codificar en hardware.

El incremento de la conectividad a nivel global hace que sea un deseo lógico el disponer de medios de comunicación entre la empresa y sus clientes y proveedores, aunque las herramientas disponibles en ambos puntos de la comunicación (Cliente – Servidor) no siempre son compatibles.

Ahora bien, el hecho de que Internet sea el medio para transportar datos por excelencia, también es la causa de que alguna de las ventajas de la red se conviertan en problemas. Porque Internet está accesible a todo el mundo, a la vez que ha promovido un crecimiento explosivo de las capacidades de comunicación personal y ha proporcionado un método fácil de acceso a recursos informáticos públicos pero distantes, también se ha convertido en un método de fácil acceso a recursos informáticos distantes. Es decir, la empresa que para aumentar su productividad opte por el establecimiento de una red privada virtual entre sus usuarios locales, remotos y móviles, a la vez que pone sus recursos a disposición de los usuarios autorizados, corre el riesgo de ponerlos al alcance de usuarios no autorizados con los riesgos que ello conlleva.

Es lógico pues que la evolución de las redes privadas virtuales (VPN[58]) se dirijan hacia las redes privadas virtuales seguras (SVPN[59]). Para proporcionar comunicaciones seguras, un sistema debe ser capaz de proporcionar siete puntos principales, como son:

---

[58] El concepto de VPN es: un conjunto de redes locales, cuyo propósito es compartir sus recursos, conectadas entre sí, semejante a una Intranet.

- Confidencialidad: Debe proteger el valor de la información.
- Integridad: Debe proporcionar protección de la veracidad de la información.
- Autenticación: Debe proporcionar confianza en el origen de la información.
- Viabilidad: Debe asegurar la recepción de la información.

Los tipos básicos de ataques se pueden dividir en cinco apartados, como son:

- Los ataques a la viabilidad, confidencialidad, integridad y autenticación.
- Interrupción del servicio, que ataca la viabilidad.
- Interceptación del flujo de datos, que ataca la confidencialidad.
- Modificación de la información, que ataca la integridad.
- Generación de la información, que ataca la autenticación.

Para conseguir todos estos puntos, existe un amplio abanico de mecanismos, que pueden ir desde los tokens para acceso personal hasta los sistemas seguros, pasando por la educación de los usuarios, los firewall o los sistemas para codificar datos. El planteamiento es: que mecanismos que se encargan de proporcionar seguridad en las fases en que los datos se encuentran en el canal de transmisión, es decir, desde que salen del equipo informático origen hasta que llegan al equipo informático de destino.

Las estrategias básicas para enfrentarse, mediante la codificación de los datos, a los tres tipos de ataques son los siguientes:

- Contra la *"interceptación del flujo de datos"*, el concepto de la defensa, para este caso, se basa en que los datos interceptados no contienen información accesible para el atacante porque está codificada, para descodificarlo tendría que disponer de una capacidad computacional enormemente elevada para poder obtener la información en tránsito.

- Contra la *"modificación del flujo de información"*, la codificación proporciona la capacidad de detección de dicha modificación, alertando de la intrusión y permitiendo que se rechace la veracidad de la información suplantada.

- Contra la *"generación de la información"*, la ausencia de las herramientas para codificar por parte del atacante o de las claves, permiten inhabilitar el ataque mediante la capacidad de descartar la información que no sigue los estándares de codificación que se hayan decidido necesarios para considerar fiable la información recibida.

Una desventaja por el hecho de efectuar un procesamiento sobre la información, sobre todo si dicho proceso es gravoso en cuanto a los recursos computacionales, es que el rendimiento global del sistema disminuye.

Para el caso de las comunicaciones, este inconveniente, no es demasiado relevante en el caso de que se pueda trabajar con sistemas de comunicaciones en diferido, como la transferencia de imágenes o archivos, pero se vuelve una desventaja muy notable en el caso de las comunicaciones en tiempo real, como la ejecución remota de aplicaciones. Puesto que en una red privada virtual segura es necesario poder acceder tanto a la ejecución de aplicaciones como a la compartición de recursos, es necesario mantener un mínimo ancho de banda para que la utilización de la red sea efectiva, ya que la codificación de la información afecta al ancho de banda.

Con estas ideas podemos pensar que si bien la codificación de los datos es una manera aceptable de asegurar la fiabilidad de la transmisión, debemos proveernos de un sistema para codificar que

[59] El concepto de una SVPN es: un conjunto de redes locales conectadas entre si con una serie de medidas para mantener segura la información que viaja en ellas, una de las medidas es codificar la información.

proporcione una buena velocidad de proceso al sistema, y no perjudique el ancho de banda de la red.

Esta nueva necesidad de velocidad se puede solucionar a través de aceleradores de hardware, que descarguen todas las operaciones para codificar. Es posible codificar los datos mediante equipos dedicados, para ello existen dos enfoques principales:

1. Codificación de los datos en la computadora de origen.

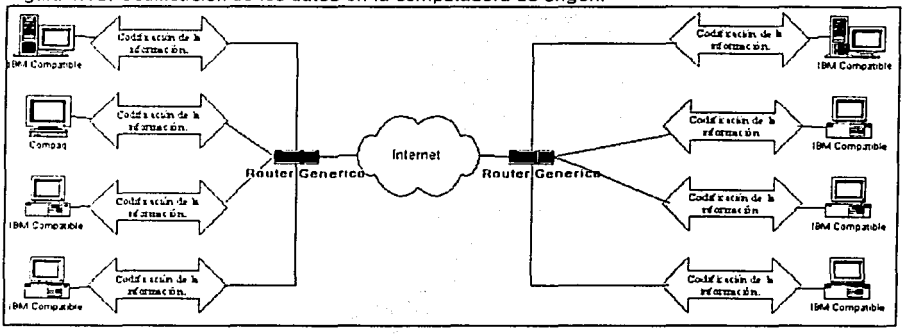
2. Codificación de los datos en equipos específicos (como firewall, gateways dedicadas).

### 4.6.1. Codificación de los datos en la computadora de origen.

La codificación del flujo de los datos en la computadora de origen consiste en proveer a los equipos de la red de tarjetas de red con capacidades para codificar (Figura 4.10). La ventaja de este sistema radica en que no es necesario efectuar ninguna operación adicional al momento de codificar y transmitir los datos a través de la red local/Internet, con lo cual el servidor ya no tiene que hacer la codificación y el ancho de banda efectivo de la red se mantiene en un óptimo estado.

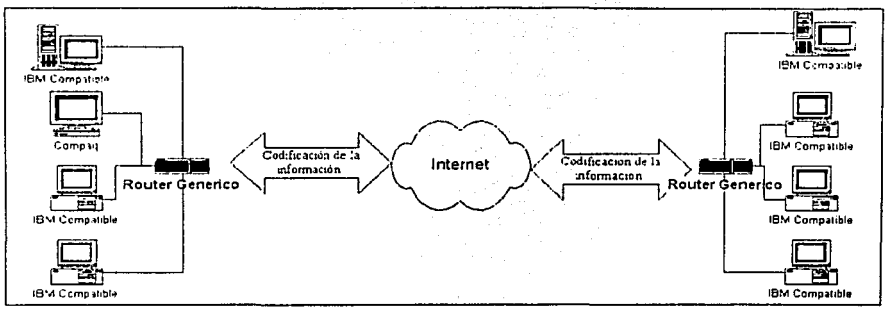
El inconveniente principal de esta solución es que los equipos para codificar deben estar presentes en todos y cada uno de los equipos que forman la red generando un costo que se eleva según el número de computadoras.

Figura 4.10. Codificación de los datos en la computadora de origen.




Visto el inconveniente que proporciona el proveer de periféricos con capacidades para codificar a todos y cada uno de los ordenadores que forman la VPN, la solución que más frecuentemente se adopta es la de unificar en un solo punto la codificación de los datos, dicho punto es un dispositivo dedicado que tendrá un dispositivo en hardware dedicado a codificar la información, dicho dispositivo puede ser, por ejemplo, un ruteador como se muestra en la siguiente figura.

Figura 4.11. Codificación de los datos en dispositivos dedicados, para este caso en un router.




En la figura 4.11, los ruteadores codifican la información que se generan por computadoras de la red local y los envía a otra red destino, entonces ahorramos dinero porque ya no necesitamos una tarjeta dedicada a codificar información en cada maquina de la red local.

De esta manera, el tráfico se mantiene en modo claro en el interior de cada una de las subredes que componen la red, y solo es codificado cuando el flujo de información se prevé que deba ser dirigido al exterior.

Mediante esta arquitectura se disminuye el costo de implantación para esta solución, si bien no se proporciona protección alguna ante ataques al interior de la red, del estilo de sniffers, o de vampiros en la instalación de cableado, por lo cual se debe pensar en la seguridad interna de la red, para ello se pueden implantar medidas que controlen estas variables sin perjudicar el rendimiento del sistema ni de la velocidad en transmisión (se pude instalar Secure Shell para cada subred).

### 4.6.2. Tecnologías para codificar.

En función de cuál sea la filosofía adoptada como solución para proporcionar la codificación al flujo de información se dispone de dos métodos, que ya comentamos, de implementación en el proceso de codificación. Para la solución de instalar una tarjeta por computadora, la codificación que se realiza a través de dichas tarjetas codificadoras locales es la que realmente presenta una codificación por hardware, a cargo de procesadores específicos.

En el caso de los equipos dedicados, por ejemplo, gateways la codificación suele estar a cargo de coprocesadores matemáticos con subsistemas de memorias situados en el equipo. Están construidos alrededor de procesadores Pentium Pro © o de procesadores Pentium II © o SPARC ©, por lo cual estos equipos al disponer de capacidad computacional remanente, pueden efectuar en un solo nodo de red otras tareas como pueden ser las de autenticación de usuarios y de firewall, disminuyendo así el costo total de implantación y explotación.

### 4.6.3. Estándares soportados.

En la siguiente tabla se puede observar un resumen de cuáles son las capacidades más usuales a nivel de intercambio seguro de datos en lo que es el general de los sistemas aceleradores.

Tabla 4.1. Estándares para aceleradores en Hardware

| | |
|---|---|
| *PROTOCOLOS.* | *PKCS#11.*<br>IKE.<br>IPSec. |
| FUNCIONES PARA CODIFICAR. | Algoritmos de clave publica<br>RSA (512-2048)<br>DSA (1024) |
| INTERCAMBIO DE CLAVES. | Diffie-Hellman (512-1024) |
| FIRMA DIGITAL. | RSA (2048)<br>DSA (1024 bits) |
| ALGORITMOS DE CLAVE SECRETA. | DES.<br>3DES.<br>RC2.<br>RC4.<br>RC5.<br>CAST-128. |
| ALGORITMOS DE HASH. | SHA-1.<br>MD-2.<br>MD-5. |
| CÓDIGOS DE AUTENTICACIÓN DE MENSAJES. | HMAC-MD5.<br>HMAC-SHA-1.<br>SSL3-MD5-MAC.<br>SSL3-SHA-1-MAC. |
| GENERACIÓN DE NUMEROS PSEUDOALEATORIOS. | Por ruido blanco<br>Anexos C de ANSI X9.17<br>Validados FIPS 140-1 |

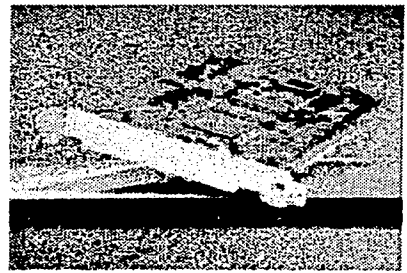
Retomando todas estas ideas, Se sabe que ya se disponen de sistemas hardware para la codificación de los datos, esto trae un beneficio para reducir la carga de trabajo para cualquier servidor y la velocidad de transmisión es muy buena, se recomienda sobre todo para redes que constantemente intercambian información a través de Internet por lo cual necesitan asegurar que la información que envían y reciben es la correcta.

El costo de adquirir el hardware se compensa con los beneficios que proporciona, principalmente la velocidad de ancho de banda y quitarle la carga de trabajo al servidor, sobre todo para subredes que interactúan dentro de Intranets, o de empresas que necesitan intercambiar información, por ejemplo pagos, cobranzas, con sus clientes, proveedores, etc.

Finalmente el uso del hardware para codificar facilita el uso de las técnicas para la codificación asimétrico y simétrico, el intercambio de claves, la codificación en flujo, la codificación en bloque y todas las capacidades que hoy soportan estos aceleradores en hardware (ver figura 4.12).

Figura 4.12. Ejemplo de una tarjeta para codificar información basada en hardware.

La tarjeta aceleradora para instalar en una computadora, proporciona servicios de transmisión de la información, así como codificación de la información, manteniendo un rendimiento adecuado para el acceso remoto y también para ejecutar aplicaciones de subred a subred. Este acelerador basado en hardware se ha optimizado para gestionar las repetitivas, pero voluminosas funciones matemáticas necesarias para brindar servicios para codificar y/o descodificar la información de manteniendo un nivel de seguridad óptimo. Algunos de los servicios que proporciona son autenticación, control de acceso, integridad y confidencialidad.



La tarjeta aceleradora, se coloca en una ranura PCI del chasis, códifica los datos utilizando los algoritmos Data Encryption Standard (DES) de 56 bits o 3DES de 168 bits a velocidades que llegan hasta los 100 Millones de bits por segundo. Además de la codificación de la información, la tarjeta gestiona una gran variedad de tareas relacionadas con Ipsec, hashing, intercambio de llaves que liberan al procesador central para que puedan llevar a cabo otras tareas

## 4.7. Conclusión.

Como ya se dijo en el presente capítulo la codificación de la información es un método para transformar un texto comprensible, mediante un algoritmo en software o hardware, a un estado que sólo es posible entender mediante un método para descodificar el texto y en algunos casos quién podrá descodificar será él que conozca una clave sobre el mismo texto con que se codifico dicha información.

Si no se comprende y no se hace uso de herramientas que emplean algoritmos para codificar la información, la red local se expone a que un atacante o cualquier usuario del sistema, pueda obtener información considerada como confidencial dañando la privacidad de la información.

Si se comprende y se usan las herramientas que utilizan algoritmos para codificar, entonces se crean las condiciones para garantizar a los integrantes del sistema de que la información la puede ver sólo quién tenga la autorización de hacerlo y de esta manera asegurar la confidencialidad de la información.

Los algoritmos que codifican la información, son de gran utilidad para mantener la privacidad y la integridad de la información, brindando la tranquilidad a todos los usuarios porque saben que la información se encuentra en un estado que no es fácil de comprender y aún cuando un atacante obtenga el texto no podrá comprenderlo.

En el próximo capítulo se explicarán las herramientas para proteger la información en un servidor y en canales inseguros, estas herramientas usan los algoritmos vistos en este capítulo, estudiaremos la importancia de usarlos así como la relación con la seguridad del sistema.

# CAPÍTULO 5.

# HERRAMIENTAS DE SEGURIDAD.

## 5.1. Introducción.

Después de haber estudiado los algoritmos para codificar la información más comunes, por su uso, ahora se estudiarán algunas herramientas de seguridad informática con base a estos algoritmos, explicando sus ventajas y desventajas, el costo que representan para el rendimiento del servidor, y para aquellos casos donde el rendimiento del sistema es vital pero de igual forma es necesario mantener la información codificada, es conveniente considerar una solución basada en hardware que estudiamos en el capítulo anterior, la cual ayuda a proteger la información manteniendo el rendimiento esperado del sistema.

La mayoría de las herramientas explicadas han sido probadas en un servidor Linux Red Hat 6.1 y en un servidor Solaris 5.7, pero pueden ser aplicadas para cualquier plataforma que emplee el uso del sistema operativo UNIX.

Cabe hacer notar que estas herramientas pueden ser de gran ayuda, pero un mal uso de las mismas podrá afectar al sistema, es por ello que es muy importante documentarse bien y conocer su modo de operación de cada una, en la sección de fuentes de documentación hay direcciones de Internet las cuales resultaron de gran ayuda al configurar estás herramientas en los sistemas mencionados. Para obtener mayor ventaja es recomendable ver los sitios Web en su idioma de origen (ingles), para el caso de personas que no puedan leer ingles, existe la posibilidad de documentarse en la página http://www.asc.unam.mx o la de http://www.seguridad.unam.mx, ya que están explicados en idioma español y son muy buena ayuda cuando se empieza a conocer sobre estos temas.

## 5.2. Utilidad de las herramientas de seguridad.

Ningún sistema operativo se puede considerar seguro y menos los que vienen preinstalados en los equipos[60]. Normalmente, cualquier distribución de un sistema se instala pensando en proporcionar los mínimos problemas a un administrador que desee poner la máquina a trabajar inmediatamente, sin tener que preocuparse en configurar el servidor. Por ejemplo si quisiera un sistema UNIX instalado en su modo más restrictivo en cuanto a seguridad, entonces el administrador deberá conocer muy bien al sistema, ya que ha de dar explícitamente los permisos necesarios para realizar cada tarea, con la consiguiente pérdida de tiempo, por esta razón es mucho más productivo para cualquier empresa desarrolladora de sistemas, proporcionarlos de forma preinstalada, de manera que el administrador no tenga que preocuparse mucho de cómo funciona cada parte del sistema que acaba de instalar: simplemente inserta el CDROM original, el software se instala, y todo funciona a la primera, aparentemente sin problemas. Esta política, que lamentablemente siguen casi todas las empresas desarrolladoras, convierte a un sistema UNIX que no se haya configurado mínimamente en un fácil objetivo para cualquier atacante.

Un administrador piensa que para aumentar la seguridad de su sistema, necesitará cerrar ciertos servicios de red o detener algunos demonios, haciendo esto obtiene una sensación de seguridad. Esta persona va a pensar equivocadamente que su sistema es seguro por negar servicios. La mejor alternativa es implementar el uso de herramientas que brinden una seguridad óptima aún con los servicios que son una función principal que un administrador debe proporcionar.

## 5.3. Descripción de algunas herramientas de control y seguimiento de accesos.

En esta sección se encuentran aquellas herramientas que permitirán tener información, mediante archivos, de todos los intentos de conexión que se han producido sobre nuestro sistema, permitir o negar accesos de paquetes IP a la red, así como de ataques a puertos de TCP incluyendo a UDP (herramientas de tipo SATAN).

---

[60] Una instalación de forma Preinstalada es aquella que se instala con el software más común o una selección de software específico.

Este tipo de herramientas nos permite tener un control sobre todos los paquetes que entran por la interfaz de red de la máquina: IP (TCP, UDP) e ICMP, o analizando los paquetes a nivel de aplicaciones (telnet, ftp, smtp, login, shell, etc.). Estas herramientas pueden ser utilizadas junto con otras que nos permitan definir desde qué máquinas permitimos ciertas conexiones y cuales prohibiremos.

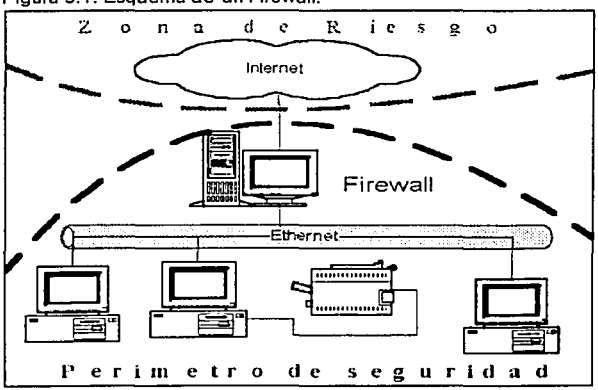
Algunas de las herramientas descritas en este apartado pueden tener un doble uso. Es decir, nos permiten protegernos ante posibles ataques, pero también podrían ser utilizadas para intentar comprometer los sistemas. Por eso es importante que el uso de estas herramientas esté restringido, en la manera que se pueda, para que no todos los usuarios las usen.

Las herramientas que estudiaremos para este apartado son: firewall, nmap, satan, tcp – wrapper, nessus. También existen más herramientas que tienen funciones muy similares a las que aquí trataremos y aunque no serán estudiadas, mencionaremos algunos como netlog, argus, tcpdump, ISS, courtney, gabriel, nocol, tcplist, etc.

### 5.3.1. Firewall.

Muchas redes privadas no están directamente conectadas con Internet por razones de seguridad. La máquina que juega un papel de barrera entre la red pública y la red privada se llama firewall. Es un sistema o grupo de sistemas que hace cumplir una regla para controlar el acceso entre dos redes o una red con Internet, dicho sistema (desde un router hasta varias redes en serie) es utilizado para separar una máquina o subred del resto de redes (Internet), protegiéndola así de servicios y protocolos que desde redes exteriores puedan representar una amenaza a la seguridad del sistema.

Figura 5.1. Esquema de un Firewall.




Con esta idea es posible identificar un espacio protegido, denominado perímetro de seguridad la cual suele ser propiedad de la misma organización, y la protección se realiza contra una o varias redes externas consideradas como no confiable, llamada zona de riesgo.

De esta manera mediante un firewall es posible aislar una red local de otras redes exteriores.

Un firewall se puede configurar de las siguientes dos maneras.

- Como filtro de paquetes IP: define si se permite o se niega el acceso de un paquete IP a nuestro sistema de acuerdo a determinadas reglas de acceso.
- Como compuerta: En esta clase de configuración los paquetes IP no son rechazados, sino enviados a una máquina especial que es quien realmente realiza las solicitudes de los usuarios y recibe la respuesta desde el exterior.

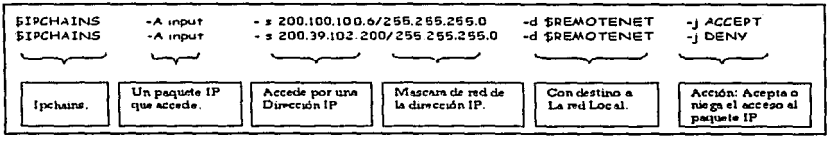
Filtros de Paquetes IP.

- Normalmente, los filtros están constituidos por un conjunto de reglas, en la configuración del firewall, cuya función es examinar un paquete IP con las reglas definidas en el sistema para buscar una similitud con algún parámetro del paquete IP.
- Si el firewall encuentra alguna coincidencia del paquete IP con la regla, entonces aplica dicha regla; por ejemplo, si la dirección de origen del paquete es 200.100.100.6 y existe una regla en el firewall cuya indicación sea permitir el acceso a todos los paquetes de la dirección IP 200.100.100.6 entonces el firewall dejará pasar dichos paquetes IP.
- Si al final de las comparaciones no se encuentran similitudes, se aplica la regla predeterminada del sistema (una regla por default).

Ejemplos de criterios para Filtrar paquetes:

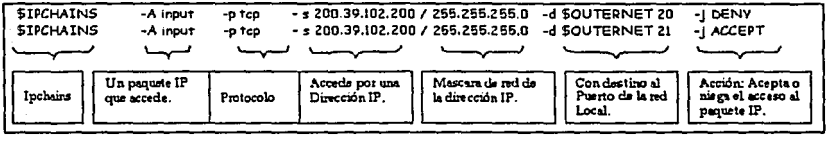
- Por dirección de origen o dirección de destino. En el siguiente ejemplo se permite el acceso a todos los paquetes que pertenezcan a la dirección IP 200.100.100.6 y se negará el acceso a los paquetes con la dirección IP 200.39.102.200.

  Ejemplo:



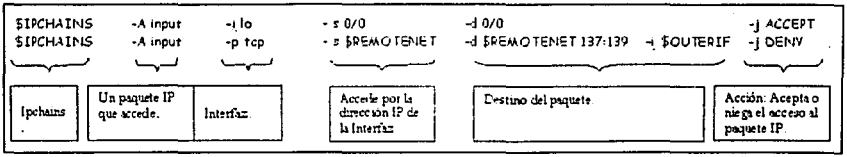

- Por puerto de origen o puerto de destino. En el siguiente ejemplo se permite el acceso a todos los paquetes que provengan del puerto 21 y se negará el acceso a los paquetes del puerto 20.

  Ejemplo:




- Por la interfaz de los paquetes. En el siguiente ejemplo se negará el acceso a los paquetes con la interfaz netbios y se aceptarán todas las de loopback.

Ejemplo:




- Existen más criterios para filtrar paquetes, las cuales no serán tratadas en este tema, dichas reglas se aplican según las necesidades del sistema lo exija, con base a los servicios que necesita proporcionar. Para mayor información revisar la documentación que viene con el software del firewall ya que tienen mas ejemplos con explicaciones más detalladas.

Ventajas de la Filtración de Paquetes IP.

1. Permite controlar desde un solo punto (computadora) todo el acceso de paquetes IP a nuestra red.
2. No necesita que el usuario invierta mucho tiempo vigilando al firewall durante el tiempo en que esta trabajando.
3. Los routers tienen la capacidad de filtrar paquetes IP mediante la implementación de un firewall en hardware.

Desventajas de la filtración de paquetes.

1. Las reglas de acceso manipulan sólo acciones de permitir ó rechazar paquetes IP a nuestro servidor.
2. Las reglas de acceso son, por lo general, difíciles de comprender así como también para depurar.
3. Si la aplicación de una regla para filtrar paquetes es incorrecta, entonces dejará un hueco de seguridad en el sistema.

### 5.3.2. Nmap.

El nmap es una herramienta de escaneo más reciente y con más características. Tiene técnicas avanzadas, como las huellas tcp-ip, un método por el cual se examinan los paquetes tcp-ip devueltos, y deduce el sistema operativo del host basándose en diferentes peculiaridades presentes en todas las pilas tcp-ip. El nmap soporta un número de métodos de escaneo, desde los escaneos normales de tcp (tratar de abrir una conexión como es habitual) hasta escaneos clandestinos (stealth scanning) y escaneos syn semi–abiertos. Este es indiscutiblemente uno de los mejores programas de escaneo de puertos disponibles, ya sea comercial o de cualquier otro tipo.

Nmap se encuentra disponible en http://www.insecure.org/nmap/index.html. También existe un interesante artículo disponible en http://raven.genome.washington.edu/security/nmap.txt sobre nmap y acerca del uso de sus características más avanzadas.

### 5.3.3. SATAN (Security Administrator Tool for Analyzing Networks).

Es un software de dominio público creado por Dan Farmer que revisa máquinas conectadas en red y genera información sobre el tipo de máquina, qué servicios da cada máquina y avisa de algunos fallos de seguridad que tengan dichas máquinas. Una de las ventajas de SATAN frente a otros paquetes es que utiliza una interfaz de WWW (como Mosaic, Netscape, etc.) y va creando una base de datos de todas las máquinas revisadas y las va relacionando entre ellas (de forma que si encuentra una máquina insegura, y revisa otra máquina que está relacionada con ésta,

automáticamente quedará marcada esta segunda también como insegura), además tiene la posibilidad de poder revisar las máquinas con tres niveles.

1. Light. (alta)
2. Normal. (normal)
3. heavy. (bajo)

Una vez realizado la revisión de la máquina se genera una salida en formato html, y en el caso de encontrar fallos da una breve explicación sobre el fallo en concreto y si existe algún documento sobre ese fallo recogido en el CERT, ya que cuenta con un enlace a ese documento, para que pueda ser consultado. Además en el caso que el fallo de seguridad sea ocasionado por versiones antiguas de software da la posibilidad (mediante un enlace) de instalar una versión de ese software.

Algunos de los servicios revisados por SATAN son: finger, nfs, nis, ftp, dns, rexd, así como tipo de sistema operativo, versión de sendmail, etc. La base de datos generada por satan puede ser consultada por varios campos: por tipo de sistema operativo, tipo de servicio (nis, ftp, nfs, etc.). SATAN ha sido diseñado como una herramienta de seguridad para ayudar a administradores de sistemas y redes, pero también puede ser utilizada para atacar a sistemas y descubrir la topología de la red de una organización (SATAN es capaz de revisar máquinas por subredes, con lo que quedan al descubierto todas las máquinas que se encuentran conectadas en dicha subred)

Para poder compilar y ejecutar SATAN basta con poseer la versión 5 de perl y un visualizador de WWW. Algunos de los fallos de seguridad que SATAN es capaz de detectar son:

- Acceso vía rexec.
- Vulnerabilidad en el sendmail.
- Acceso vía tftp.
- Accesos vía rsh.
- Acceso a servidores sin restricción alguna.
- Exportar sistemas de archivos no restringido.
- Acceso a archivos de password vía nis.

### 5.3.4. TCP Wrappers.

Hay una serie de servicios como telnet o ftp que habitualmente no vamos a poder cerrar, ya que los usuarios necesitarán conectarse al servidor para trabajar en él o para transferir archivos, en estos casos es peligroso permitir que cualquier máquina de Internet tenga la posibilidad de acceder a nuestros recursos, por lo que se suele utilizar un programa denominado Tcp Wrappers para definir una serie de redes o máquinas autorizadas a conectarse con nosotros.

Actualmente, cualquier administrador que desee un mínimo de seguridad ha de instalar TCP Wrappers en sus equipos, incluso algunas versiones UNIX como Linux o BSD lo ofrecen como un software predeterminado al instalar el sistema operativo. Cabe mencionar que la configuración del programa puede ser muy elaborada y con muchas opciones razón por la cual se recomienda documentarse muy bien.

### 5.3.5. Nessus.

Sin duda una de las herramientas de seguridad más utilizadas durante años en todo tipo de entornos UNIX ha sido SATAN (Security Analysis Tool for Auditing Networks), desarrollada por Dan Farmer y Wietse Venema. La tarea de SATAN (o SANTA) era detectar vulnerabilidades de seguridad en sistemas UNIX y redes, desde fallos conocidos en el software hasta políticas incorrectas, el resultado de su ejecución se mostraba en formato HTML, de forma que cualquier administrador podía analizar esa información de una forma muy cómoda.

Sin embargo, todo esto sucedía en abril de 1995, y SATAN no se ha actualizado mucho desde entonces. Evidentemente, para una herramienta de seguridad este tiempo sin nuevas versiones es demasiado, por lo que en 1998 surgió Nessus, un analizador de vulnerabilidades gratuito, de código fuente libre, y lo más importante: igual de fácil – o más – de utilizar que su predecesor (SATAN).

La distribución de Nessus consta de cuatro archivos básicos: las librerías del programa, las librerías (Nessus Attack Scripting Language), el núcleo de la aplicación y sus plugins, es necesario compilar en este orden cada una de esas partes. Además, el programa requiere para funcionar correctamente pequeñas aplicaciones adicionales, como la librería gmp, necesaria para las operaciones de codificación. La compilación sobre diferentes plataformas UNIX no ofrece ningún problema siempre que se realice en el orden adecuado, y se suele limitar a un "./configure, make y make install" para cada una de las cuatro partes de Nessus.

## 5.4. Herramientas que revisan la integridad del sistema:

Se estudiará a continuación una serie de herramientas que ayudan a proteger el sistema. Para lograr esto hay dos clases de herramientas. Las primeras se basan en revisiones de los archivos y las segundas nos alertan de posibles modificaciones de archivos, y de programas "sospechosos" que puedan estar ejecutándose en la máquina de forma camuflageada. Se estudiarán primero las que revisan la integridad de los sistemas de archivos.

### 5.4.1. COPS (Computer Oracle and Password System).

Cops es un conjunto de programas diseñados por la Universidad de Purdue que revisan ciertos aspectos del sistema operativo UNIX relacionados con la seguridad. Existen dos versiones de este paquete: una versión escrita en "sh" y "C" y otra versión escrita en "perl", aunque su funcionalidad es similar. Este programa es fácil de instalar y configurar, se ejecuta en gran cantidad de plataformas UNIX. En el primer caso se necesita un compilador de lenguaje C y un shell estándar (sh), en el segundo nos bastará con tener instalado un intérprete de perl (versión 3.18 o superior). Entre las funcionalidades que tiene cops podemos destacar:

- Revisión de modos y permisos de los archivos, directorios y dispositivos.
- Palabras de passwd débiles (en el caso de que se tenga una herramienta como crack, se puede comentar la línea de *"revisión de palabras de paso"*).
- Revisión de contenido, formato y seguridad de los archivos de *"password"* y *"group"*.
- Revisión de programas con root-SUID.
- Permisos de escritura sobre algunos archivos de usuario como *".profile"* y *".cshrc"*.
- Configuración de *"ftp anonymous"*.
- Revisión de algunos archivos del sistema como *"hosts.equiv"*, montajes de NFS sin restricciones, *"ftpusers"*, etc.

### 5.4.2. Tiger.

Es un software desarrollado por la Universidad de Texas que está formado por un conjunto de shell scripts y código C que analizan el sistema para detectar problemas de seguridad de forma parecida a COPS. Una vez revisado el sistema, se genera un archivo con toda la información recogida por el programa. Tiger dispone de una herramienta (tigexp) que recibe como parámetro dicho archivo y da una serie de explicaciones adicionales de cada línea que generó el programa anterior. El programa viene con un archivo de configuración donde es posible decirle qué tipo de revisión se quiere realizar. Entre la información que analiza el programa tenemos:

- Configuración del sistema.
- Sistemas de archivos.
- Archivos de configuración de usuario.
- Revisión de cuentas.
- Revisión de alias[61].
- Comprueba la configuración de ftp "anonymous".
- Revisión en scripts de cron.
- NFS.
- Análisis de servicios en el archivo /etc/inetd.conf
- Análisis de algunos archivos de usuario (*".netrc"*, *".rhosts"*, *".profile"*, etc.).
- Comprobación de archivos binarios (firmas). Para poder revisar éstos es necesario disponer de un archivo de firmas.

### 5.4.3 Crack.

Crack, desarrollado por Alec Muffet, es el "adivinador" de contraseñas más utilizado en entornos UNIX, actualmente se encuentra en su versión 5, que funciona correctamente en la mayoría de clones del sistema operativo (Linux, Solaris, SCO, etc.). Ejecutar periódicamente Crack sobre el archivo de contraseñas de sus sistemas es algo muy recomendable para cualquier administrador mínimamente preocupado por la seguridad, sin importar que se utilicen mecanismos para obligar a los usuarios a elegir passwords aceptables.

Este programa realiza una primera pasada sobre el archivo de claves intentando romper contraseñas con base en la información de cada usuario almacenada en el archivo; se trata de unas comprobaciones rápidas pero efectivas, ya que aunque la cantidad de datos del archivo no es muy grande, se trata de información frecuentemente utilizada como password. Tras esta pasada, entran en juego los diccionarios. Para seguir adivinando contraseñas (diccionario no es más que un archivo con posibles passwords en él, generalmente uno por línea). El propio programa se distribuye con algunos de estos archivos, pero es recomendable que se complementen con más diccionarios (existen multitud de ellos disponibles a través de Internet), especialmente con aquellos que contengan palabras que por las características del sistema sean susceptibles de ser usadas como claves; por ejemplo, si estamos en México seguramente convendrá utilizar un diccionario con palabras en español, si administramos una máquina de un departamento de biología, otro con términos de esta ciencia, incluso si nuestros usuarios son aficionados al deporte, o a la literatura griega antigua, podemos encontrar diccionarios adecuados a estos campos.

Con todos estos diccionarios – los propios y los que cada administrador puede añadir – Crack construye una base de datos con la que empieza a trabajar; la primera pasada utilizando diccionarios consiste simplemente en probar palabras con todas las letras en minúsculas. Posteriormente, se mezclan mayúsculas y minúsculas, y de esta forma se van combinando caracteres hasta añadir números y caracteres alfanuméricos a cada palabra de los diccionarios para comprobar que dicha combinación no es utilizada como contraseña en el sistema. Habitualmente las primeras pasadas son las que más claves son capaces de romper, pero una vez adivinados los passwords más débiles quizás interese seguir ejecutando Crack para obtener contraseñas más elaboradas: recordemos que un atacante puede aprovechar la potencia de servidores en los que ha penetrado para ejecutar Crack sobre el archivo de contraseñas durante mucho tiempo, por lo que es posible que "adivine" claves que a priori no son seguras.

---

[61] Un alias es un nombre y a este nombre se asocia uno o varios comandos los cuales serán ejecutados al ejecutar el nombre que hemos definido como alias. Veamos un ejemplo: *alias r=`rm –i`*, con esto hemos definido que al teclear el nombre de alias *"r"* ejecutara el comando *"rm"* para borrar archivos con la opción *"i"*, es decir de manera interactiva.

Tal y como se explica en su documentación, la forma en que Crack trata de adivinar contraseñas es seguramente la que consigue mayor velocidad; en primer lugar se ordenan y se agrupan las entradas del archivo de passwords en base a su salt. Una vez clasificadas, para cada grupo de salts diferente se selecciona una entrada de diccionario convenientemente tratada (mayúsculas, números, etc.), se cifra utilizando el salt (esto es lo que consume mayor tiempo de CPU) y se compara con la contraseña cifrada de cada miembro del grupo; si coinciden, se ha adivinado un nuevo password.

Con esta idea, hay que volver a insistir sobre el uso regular de Crack en cada una de las máquinas, aunque muchos administradores consideran que utilizar este tipo de programas puede equipar a un intruso, hemos de pensar siempre que es mejor que las contraseñas débiles las encuentre el administrador antes que un atacante. Y si el administrador no utiliza un programa de este estilo, existe la posibilidad de que un atacante podrá hacerlo.

### 5.4.4. Tripwire.

La herramienta Tripwire es un comprobador de integridad para archivos y directorios de sistemas UNIX: compara un conjunto de estos objetos con la información sobre los mismos almacenada previamente en una base de datos, y alerta al administrador en caso de que algo haya cambiado.

La idea consiste en: crear un resumen de cada archivo o directorio importante para nuestra seguridad por ejemplo /etc/passwd al instalar el sistema, y esos resúmenes se almacenan en un medio seguro (un CDROM o un disco protegido contra escritura), de forma que si alguno de los archivos es modificado (por ejemplo, por un atacante que sustituye un programa por una versión troyana o añade una entrada en nuestro archivo de contraseñas) Tripwire podrá alertar la próxima vez que realicemos la comprobación. Para generar esos resúmenes se utilizan funciones hash, de forma que es casi imposible que dos archivos generen el mismo resumen; concretamente Tripwire implementa md2, md4, md5, Snefru, crc16 y crc32.

Es muy útil para encontrar un programa con un Caballo de Troya, por mencionar un ejemplo: un comando de UNIX como last[62] puede haber sido cambiado por otro programa con el mismo nombre cuya función ya no es completamente la de last, sino realiza otro tipo de procesos, en este caso tripwire nos alertaría que este programa ha sido alterado. Esto le ayuda al administrador a tomar las medidas correspondientes.

Una vez que se ha compilado el código fuente de Tripwire se debe inicializar la base de datos; para ello se necesita en primer lugar crear el archivo tw.config en la localización indicada en include/config.h, donde se especifican los directorios a analizar. A continuación se muestra un ejemplo para iniciar la base de datos con el comando: *tripwire -initialize* (otra opcion es: -init):

```
saul:/tmp/tripwire1.2/src# ./tripwire -init
### Phase 1: Reading configuration file.15.5. TRIPWIRE 251
### Phase 2: Generating file list
### Phase 3: Creating file information database
###
### Warning: Database file placed in ./databases/tw.db_saul.
###
### Make sure to move this file file and the configuration
### to secure media!
###
### (Tripwire expects to find it in "/usr/local/tw".)
saul:/tmp/tripwire1.2/src#
```

[62] El comando last nos informa quienes han sido los últimos 10 usuarios que han entrado al sistema.

En el archivo ./databases/tw.db_saul se encuentran las funciones resumen de los archivos y directorios especificados en tw.config, los datos de ese archivo se asumen como confiables, por lo que es recomendable generarlo antes de proporcionar el servicio a los usuarios. Además, si un usuario lo consigue modificar toda la seguridad de Tripwire se rompe, por lo cual se debe almacenarlo en un medio seguro (por ejemplo, un CD de sólo lectura), incluso imprimir en papel una copia para realizar comprobaciones si hay sospechas de un ataque.

Con la base de datos inicial ya generada, se puede ejecutar regularmente Tripwire (una buena opción sería programar un cron todos los viernes a las 24 horas y que nos envíe por correo cualquier advertencia de tripwire) para verificar que no ha cambiado ningún resumen de nuestros archivos; para ello es necesario utilizar dicha base de datos desde una fuente segura.[63]

```
saul:/usr/bin/tripwire1.2/src# ./tripwire &>resultados
saul:/tmp/tripwire1.2/src# head 17 resultados
### Phase 1: Reading configuration file
### Phase 2: Generating file list
### Phase 3: Creating file information database
### Phase 4: Searching for inconsistencies
###
###                          Total files scanned:          4821
###                                Files added:            2
###                                Files deleted:          0
###                                Files changed:          4413
###
###                          After applying rules:
###                                Changes discarded:      3959
###                                Changes remaining:      458
###
added:    -rw- --- ---  root      0 May  5  03:46:06  2001    /var/tmp/test
changed:  -rw- r-- r--  root     72 May  5  03:49:53  2001    /var/adm/utmp
changed:  -rw- r-- r--  root  10044 May  5  03:49:53  2001    /var/adm/utmpx
saul:/usr/bin/tripwire1.2/src#
```

Finalmente, es valido pensar que existirán archivos o directorios que van a cambiar habitualmente (por ejemplo, el archivo de contraseñas cada vez que agreguemos a un usuario al sistema); por tanto, es lógico que Tripwire ofrezca un mecanismo de actualización de la base de datos. Es más, este programa posee dos: o bien el modo interactivo o el modo actualización. En el primero, cada vez que Tripwire detecte un archivo con modificaciones nos consultará si deseamos actualizar nuestra base de datos, mientras que en el modo update se utiliza para la actualización o bien un nombre de archivo (si es lo único modificado) o bien un directorio pasado como parámetro al ejecutable.

El modo interactivo es invocado mediante la opción interactive:

```
saul:/usr/bin/tripwire1.2/src# ./tripwire -interactive
### Phase 1: Reading configuration file
### Phase 2: Generating file list
### Phase 3: Creating file information database
### Phase 4: Searching for inconsistencies
###
###                          Total files scanned:          4820
###                                Files added:            1
###                                Files deleted:          0
###                                Files changed:          4413
```

[63] En un CD-ROM o en alguna cinta guardar nuestra base de datos inicial de información del sistema.

```
###
###                          After applying rules:
###                                Changes discarded:      3958
###                                Changes remaining:      457
###
added: rw victor 32768 May 5 03:55:29 2000 /var/tmp/Rx0000755
> File: "/var/tmp/Rx0000755"
> Update entry? [YN(y)nh?]
```

Mientras que el modo update se consigue mediante el parámetro *-update*; por ejemplo, si se añadió a un usuario (y por tanto modificado los archivos */etc/passwd* y */etc/shadow*), actualizaremos la base de datos de Tripwire con el siguiente comando:

```
saul:/usr/bin/tripwire1.2/src# ./tripwire -update /etc/passwd /etc/shadow
### Phase 1: Reading configuration file
### Phase 2: Generating file list
Updating: update file: /etc/passwd
Updating: update file: /etc/shadow
### Phase 3: Updating file information database
###
### Old database file will be moved to "tw.db_saul.old"
### in ./databases.
###
### Updated database will be stored in "./databases/tw.db_saul"
### (Tripwire expects it to be moved to "/usr/local/tw".)
###
saul:/tmp/tripwire1.2/src#
```

Tripwire es una herramienta muy útil como sistema de detección de intrusos en máquinas UNIX; ejecutarlo periódicamente, y mantener segura la base de datos de resúmenes *"donde reside toda la fiabilidad del producto"* y puede ayudar a detectar programas troyanos, accesos no autorizados al sistema y lo más importante, modificaciones que un intruso haya podido realizar en el sistema para garantizarse un futuro acceso.

### 5.4.5. Titan.

Es un programa que realiza una auditoría del sistema, por ejemplo analiza si los passwd son factibles de romper, escanea puertos, revisa si los permisos son los más factibles o hay alguno que debería cambiar. Estas labores se hacen con el objetivo de corroborar la inseguridad de los sistemas UNIX instalados tal y como se distribuyen, o mínimamente configurados, cerrado la mayoría de servicios ofrecidos (en /etc/inetd.conf), y controlado el acceso a otros (telnet, finger, ftp, etc.) mediante TCP Wrappers: justo lo que la mayor parte de administradores harían antes de poner el sistema a funcionar. Tras estos pasos, hemos ejecutado el programa de auditoría automática Titan, que detecta problemas de seguridad en la máquina local.

## 5.5. Mecanismos de seguridad en las comunicaciones.

Es especialmente importante para la seguridad del sistema proteger su integridad y la privacidad de los datos cuando se transmiten a través de la red. Para garantizar esta seguridad en las comunicaciones, se puede usar varios mecanismos, la mayoría de los cuales se basan en la criptografía: codificación de clave pública, de clave privada, firmas digitales, etc. Aunque cada vez se utilizan más los protocolos seguros (como SSH o Kerberos), aún es frecuente encontrar conexiones en texto claro ya no sólo entre máquinas de una misma subred, sino entre redes diferentes. Una de las mayores amenazas a la integridad de las redes es este tráfico sin codificar, que hace extremadamente fáciles ataques encaminados a robar contraseñas o falsificar la

identidad de máquinas de la red. En este capítulo estudiaremos las herramientas más comunes para solucionar estos problemas.

### 5.5.1. SSH.

Tradicionalmente el intercambio de datos entre sistemas UNIX (desde la transferencia de archivos o compartir archivos vía NFS hasta el acceso remoto) se ha realizado utilizando mecanismos en los que la seguridad era un factor poco importante frente a otros como la velocidad o la disponibilidad. Sin embargo, conforme ha ido aumentando la calidad de los medios de transmisión (actualmente una organización tiene al menos una red Fast Ethernet capaz de alcanzar velocidades de 100 Mbps), y también conforme ha ido aumentando la peligrosidad de las redes, especialmente de Internet, se ha ido considerando más el grave problema que implica una transmisión de datos en texto claro, hecha con un telnet, un ftp o incluso la transmisión de datos que tiene lugar al utilizar sistemas de archivos en red.

Casi todos los mecanismos clásicos se han reemplazado por protocolos que incorporan procedimientos para codificar, en mayor o menor medida, la información de forma que al intruso que captura datos transmitidos entre sistemas ya le es más difícil para conseguir información importante, como por ejemplo una clave de usuario. Algunos protocolos que incorporan la criptografía son: SSL (Secure Socket Layer) o TCFS (Transparent Cryptographic File System).

Secure Shell (SSH) es uno de los modelos considerados más seguros, un software cuya principal función es permitir la conexión remota segura a sistemas a través de canales inseguros, aunque también se utiliza para la ejecución de comandos en ese sistema remoto o transferir archivos desde o hacia él de manera fiable; por lo cual, es el sustituto ideal de programas como telnet, ftp de UNIX.

La siguiente lista muestra los tipos de ataques que se puede evitar al utilizar ssh:

- IP spoofing, donde un host remoto envía paquetes a otro host aparentando ser un host conocido por este.
- Ruteo de IP spoofing, donde un host puede pretender que un paquete de IP proviene de otro hosts valido.
- DNS spoofing, cuando un ataque provoca que los registros de servidor de nombre se pierdan.
- Intersección de password en claro, u otra información intercambiada entre dos hosts.
- Manipulación de datos que se encuentran en intercambio entre hosts.

SSH también soporta procedimientos de codificación automática en sesiones XWindows o modelos de seguridad más avanzados, como la codificación en NFS o la construcción de redes privadas virtuales; su código fuente es libre para uso no comercial (existe otro software casi completamente compatible con ssh y completamente libre, denominado OpenSSH). En la actualidad, ssh funciona sobre la mayoría de clones de UNIX (también existen versiones para Windows y MacOS), y es ampliamente utilizado en todo tipo de entornos, desde universidades a bancos pasando por empresas de cualquier sector.

SSH está formado por un programa servidor, sshd, varios programas cliente (ssh y scp principalmente) y pequeñas aplicaciones para su configuración, como ssh-add, ssh-keygen o ssh-agent.

El programa demonio (sshd) se ejecuta en la máquina que tiene el servicio, mientras que los clientes han de ejecutar ssh desde sus computadoras personales, de esta manera se puede iniciar una sesión en la máquina remota con un comando como el siguiente:

```
saul:~# ssh -l ffsli zelda
ffsli's password:
```

```
Last login: Thu Apr 6 03:58:32 2000 from saul
Linux 2.2.6
"A witty saying proves nothing."
 Voltaire
zelda:~S
```

El parámetro "-l" permite indicar el nombre de usuario en el sistema remoto (en caso contrario, se utilizará el mismo nombre que se posee en la máquina local), ssh también permite especificar desde línea de comandos un comando a ejecutar en la máquina en que nos conectamos, de forma que cuando este comando finalice se cerrará la conexión entre ambos sistemas:

```
saul:~# ssh -l ffsli zelda w
ffsli's password:
3:15pm up 5 days, 1:30, 5 users, load average: 1.12, 1.04, 1.01
USER   TTY    FROM      LOGIN@   IDLE   JCPU   PCPU  WHAT
root   tty1   -         Sat12am  5days  0.02s  0.02s bash
cris   ttyp1  :0.0      Sun 3pm  1:02   0.18s  0.13s telnet rosita
cris   ttyp2  :0.0      Sun 4am  2.00s  2.40s  2.04s vi juntalinea.ksh
cris   ttyp4  saul      Tue 1am  0.00s  1.31s  0.02s w
saul:~#
```

SSH se utiliza básicamente para iniciar sesiones o ejecutar comandos en un sistema remoto; el otro programa cliente, scp, es utilizado para transferir archivos entre máquinas, de una forma similar a rcp, lo que permite sustituir el ftp tradicional por este mecanismo; por ejemplo, deseamos copiar todos los archivos del directorio /export/home/ffsli/ conectando al sistema remoto bajo el nombre de usuario ffsli en el directorio /tmp/ de la máquina local, lo podemos conseguir con un comando como este:

```
saul:~# scp -r ffsli@zelda:/export/home/ffsli  /tmp/
ffsli's password:
saul:~#
```

Como podemos ver, se indica el nombre de usuario y el del sistema remoto separados por "@", y separados a su vez de la ruta origen por el signo ":".

Lo que hacen estos clientes con el servidor sshd es lo siguiente: con la opción "-p", el cliente conecta al puerto 22 de la máquina servidora y verifica que esta máquina es realmente con la que queremos conectar, intercambia las claves de codificación entre sistemas (codificadas a su vez, para evitar que un atacante pueda obtener la información) y autentica utilizando ".rhosts" y /etc/hosts.equiv (como los protocolos r-*), RSA o claves de usuario; si todo es correcto, el servidor asigna una terminal virtual (generalmente) a la conexión y lanza un shell interactivo. Se puede ver con detalle este proceso utilizando la opción "-v" del cliente:

```
zelda:~# ssh -v -l ffsli zelda
SSH Version 1.2.21 [i486unknownlinux], protocol version 1.5.
Standard version. Does not use RSAREF.
zelda: Reading configuration data /etc/ssh_config
zelda: ssh_connect: getuid 0 geteuid 0 anon 0
zelda: Connecting to zelda [200.39.102.10] port 22.
zelda: Allocated local port 1023.
zelda: Connection established.
zelda: Remote protocol version 1.5, remote software version 1.2.21
zelda: Waiting for server public key.
zelda: Received server public key (768 bits) and host key (1024 bits).
zelda: Host "zelda" is known and matches the host key.
zelda: Initializing random; seed file /root/.ssh/random_seed
```

```
zelda: Encryption type: idea
zelda: Sent encrypted session key.
zelda: Received encrypted confirmation.
zelda: Trying rhosts or /etc/hosts.equiv with RSA host authentication.
zelda: Remote: Rhosts/hosts.equiv authentication refused:\
        client user "root", server user "ffsli", client host "zelda".
zelda: Server refused our rhosts authentication or host key.
zelda: No agent.
zelda: Doing password authentication.
ffsli's password:
zelda: Requesting pty.
zelda: Failed to get local xauth data.
zelda: Requesting X11 forwarding with authentication spoofing.
zelda: Requesting shell.
zelda: Entering interactive session.
Last login: Thu Apr 6 04:13:41 2000 from zelda
Linux 2.2.6
If you want divine justice, die.
                        -- Nick Seldon

zelda:~$ exit
logout
Connection to zelda closed.
zelda: Transferred: stdin 5, stdout 491, stderr 29 bytes in 2.6 seconds
zelda: Bytes per second: stdin 1.9, stdout 189.0, stderr 11.2
zelda: Exit status 0
zelda:~#
```

Como sucede en cualquier programa cliente–servidor, la configuración de la parte cliente es mucho más sencilla que la de la parte servidora: ni siquiera es necesario el archivo de configuración general /etc/ssh_config, donde se definen parámetros por defecto (que cada usuario puede modificar para sí mismo en sus propios archivos o en línea de comandos). Solamente es necesario el ejecutable (por ejemplo, ssh) el cual genera en el directorio $HOME/.ssh (de quien lo ejecute) varios archivos necesarios para su funcionamiento, quizá el más importante sea known_hosts, donde se almacenan las claves públicas de los diferentes sistemas a los que se conecta. Estas claves, una por línea, se guardan la primera vez que se conecta a una determinada máquina, algo que el cliente indica con un mensaje de esta forma:

```
saul:~# ssh -l ffsli zelda
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host "zelda" added to the list of known hosts.
ffsli's password:
Last login:     Thu Apr 6     23:20:42     2001     from :0.0
Linux 2.2.6
Drive defensively. Buy a tank.
zelda:~$
```

Por su parte, la configuración del servidor es algo más compleja; en el archivo /etc/sshd_config, archivo de configuración del demonio sshd, se especifican todos los parámetros necesarios para su funcionamiento. Algunos de estos parámetros, quizás los más útiles, son AllowHosts y DenyHosts, donde como su nombre indica, se hace referencia a los sistemas desde los que la conexión a nuestro demonio se permite o se cancela; al contrario de lo que mucha gente sigue pensando, utilizar ssh no implica tener disponible el servicio para todo el mundo, y es aquí donde se indica los sistemas desde donde permitimos conexiones. Además, podemos servir sshd desde inetd modificando convenientemente /etc/inetd.conf en lugar de hacerlo como demonio independiente,

```
zelda: Encryption type: idea
zelda: Sent encrypted session key.
zelda: Received encrypted confirmation.
zelda: Trying rhosts or /etc/hosts.equiv with RSA host authentication.
zelda: Remote: Rhosts/hosts.equiv authentication refused:\
       client user "root", server user "ffsli", client host "zelda".
zelda: Server refused our rhosts authentication or host key.
zelda: No agent.
zelda: Doing password authentication.
ffsli's password:
zelda: Requesting pty.
zelda: Failed to get local xauth data.
zelda: Requesting X11 forwarding with authentication spoofing.
zelda: Requesting shell.
zelda: Entering interactive session.
Last login: Thu Apr 6 04:13:41 2000 from zelda
Linux 2.2.6
If you want divine justice, die.
                        -- Nick Seldon

zelda:~$ exit
logout
Connection to zelda closed.
zelda: Transferred: stdin 5, stdout 491, stderr 29 bytes in 2.6 seconds
zelda: Bytes per second: stdin 1.9, stdout 189.0, stderr 11.2
zelda: Exit status 0
zelda:~#
```

Como sucede en cualquier programa cliente–servidor, la configuración de la parte cliente es mucho más sencilla que la de la parte servidora: ni siquiera es necesario el archivo de configuración general /etc/ssh_config, donde se definen parámetros por defecto (que cada usuario puede modificar para sí mismo en sus propios archivos o en línea de comandos). Solamente es necesario el ejecutable (por ejemplo, ssh) el cual genera en el directorio $HOME/.ssh (de quien lo ejecute) varios archivos necesarios para su funcionamiento, quizá el más importante sea known_hosts, donde se almacenan las claves públicas de los diferentes sistemas a los que se conecta. Estas claves, una por línea, se guardan la primera vez que se conecta a una determinada máquina, algo que el cliente indica con un mensaje de esta forma:

```
saul:~# ssh -l ffsli zelda
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host "zelda" added to the list of known hosts.
ffsli's password:
Last login:      Thu Apr 6     23:20:42      2001     from :0.0
Linux 2.2.6
Drive defensively. Buy a tank.
zelda:~$
```

Por su parte, la configuración del servidor es algo más compleja; en el archivo /etc/sshd_config, archivo de configuración del demonio sshd, se especifican todos los parámetros necesarios para su funcionamiento. Algunos de estos parámetros, quizás los más útiles, son AllowHosts y DenyHosts, donde como su nombre indica, se hace referencia a los sistemas desde los que la conexión a nuestro demonio se permite o se cancela; al contrario de lo que mucha gente sigue pensando, utilizar ssh no implica tener disponible el servicio para todo el mundo, y es aquí donde se indica los sistemas desde donde permitimos conexiones. Además, podemos servir sshd desde inetd modificando convenientemente /etc/inetd.conf en lugar de hacerlo como demonio independiente,

de forma que podemos aprovechar un software como TCP Wrappers para restringir conexiones; el único inconveniente de este modelo es que cada vez que alguien conecta al demonio, éste tiene que generar una clave RSA para esa conexión, lo que en determinadas situaciones puede sobrecargar demasiado al sistema. Si de cualquier forma queremos seguir este mecanismo, hemos de modificar /etc/services para añadir una línea como la siguiente:

```
ssh                 22/tcp
```

Y también modificar /etc/inetd.conf añadiendo la configuración del nuevo servicio:

```
ssh     stream tcp      nowait root     /usr/sbin/tcpd  /usr/local/sbin/sshd    -i
```

Tras lo cual, como cada vez que modificamos este archivo, hemos de conseguir que inetd lo vuelva a leer enviándole al demonio la señal sighup.

### 5.5.2. Kerberos.

Kerberos fue desarrollado en el MIT por el proyecto Athena. Es un protocolo de autenticación, diseñada para redes TCP/IP. Un servicio Kerberos, sobre una red, actúa como un árbitro confiable, ofrece autenticación segura en la red, permitiendo a una persona acceder a diferentes máquinas sobre la red. Está basada en la codificación simétrica.

Kerberos es un servicio de autenticación. El problema que intenta resolver es el siguiente: restringir los accesos sólo a usuarios autorizados y poder autenticar los requerimientos a servicios, asumiendo un entorno distribuido abierto, en el cual usuarios en workstations acceden a servicios en servidores distribuidos a través de una red. En el entorno de trabajo que considera Kerberos, una workstation no puede ser confiable para identificar a sus usuarios correctamente para servicios de red. En particular, puede suceder lo siguiente:

- Un usuario puede ganar el acceso a una workstation en particular y simular ser otro usuario operando desde otra workstation.
- Un usuario puede alterar la dirección de red de una workstation para que los requerimientos que envía simulen llegar desde otra workstation.
- Un usuario puede "escuchar disimuladamente" los intercambios y atacar para ganar la entrada a un servidor o para interrumpir operaciones.

En muchos de esos casos, un usuario podría ganar acceso a servicios y datos que no está autorizado a acceder. En lugar de construir protocolos de autenticación en cada servidor, Kerberos proporciona un servidor de autenticación centralizado, cuya función es autenticar usuarios frente a servidores y servidores frente a usuarios. Kerberos usa codificación simétrica (también llamada codificación convencional), método en el cual la codificación y descodificación se realizan usando una única clave.

En el modelo Kerberos hay entidades (clientes y servidores) sobre una red, los clientes pueden ser usuarios, programas independientes de software que necesiten hacer algo (bajar archivos, enviar mensajes, acceso a Bases de Datos, obtener privilegios de administrador). Este protocolo mantiene una base de datos de clientes y sus llaves secretas. Dado que conoce todas sus llaves secretas, puede crear mensajes para convencer a una entidad que la otra entidad es quien dice ser. También puede crear llaves de sesión para que el cliente y servidor se comuniquen con seguridad.

Versiones de Kerberos.

Actualmente se encuentran en uso común dos versiones de Kerberos:

- Versión 4: que es la más ampliamente usada, y

- Versión 5: que corrige algunas deficiencias de seguridad de la versión 4.

Diferencias entre las Versiones 4 y 5

La Versión 5 intenta solucionar las limitaciones de la Versión 4 en dos áreas: fallas del entorno y deficiencias técnicas.

### 5.5.3. El Protocolo SSL.

El protocolo SSL (Secure Socket Layer) permite establecer conexiones seguras a través de Internet, de forma sencilla y transparente. La idea consiste en interponer una fase de codificación de los mensajes antes de enviarlos por la red. Una vez que se ha establecido la comunicación, cuando una aplicación quiere enviar información a otra computadora, la capa SSL la recoge y la codifica, para luego enviarla a su destino a través de la red. Análogamente, el módulo SSL de la otra computadora se encarga de descodificar los mensajes y se los pasa como texto claro a la aplicación destino.

Una comunicación SSL consta fundamentalmente de dos fases:

1. Fase de saludo (handshaking). Consiste básicamente en una identificación mutua de los interlocutores, para la cual se emplean habitualmente los certificados X.509. Tras el intercambio de claves públicas, los dos sistemas escogen una clave de sesión, de tipo simétrico.

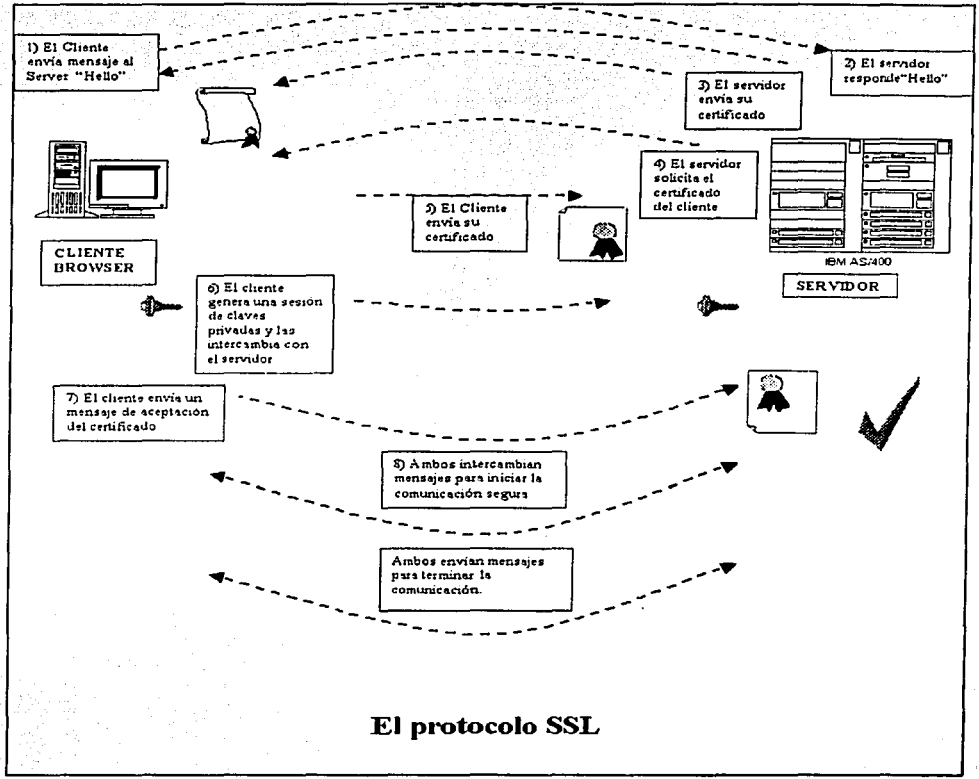
2. Fase de comunicación. En esta fase se produce el intercambio de información, que se codifica mediante la clave de sesión acordada en la fase de saludo.

Cada sesión SSL lleva asociado un identificador único que evita la posibilidad de que un atacante escuche la red y repita exactamente lo mismo que ha recibido, aún sin saber lo que significa, para engañar a uno de los interlocutores.

Entre otras ventajas de este protocolo se puede mencionar que liberan a las aplicaciones de llevar a cabo las operaciones de codificación antes de enviar la información, y su transparencia permite usarlo de manera inmediata sin modificar los programas ya existentes. Desde hace tiempo los principales navegadores de Internet incorporan un módulo SSL, que se activa de forma automática cuando es necesario. Desgraciadamente, las versiones de exportación tanto de Netscape como de Internet Explorer trabajan con claves de sesión de 40 bits, que pueden ser descifradas en cuestión de pocas horas por una computadora óptima[64].

[64] Una computadora con recursos suficientes, por ejemplo un procesador pentium III, 500 megahertz de velocidad, 128 megas en RAM son suficientes para que un equipo pueda descodificar información.

Figura 5.2. Ejemplo del protocolo SSL.



## 5.6. Conclusión.

Como ya se habló a lo largo de este capitulo existen herramientas que ayudan a mantener la información segura dentro de un servidor y en canales inseguros fuera del mismo, estas herramientas fueron diseñadas para dar solución a una clase especial de problemas relacionados con la seguridad de la información y de un servidor, dichas herramientas proporcionan un margen de seguridad adicional para un servidor UNIX.

Si no se usan las herramientas de seguridad, se limitan las posibilidades de:

- Recuperar la información, lo más rápidamente posible después de que éste ha sufrido un ataque.
- Advertir los posibles hoyos de seguridad que existen en nuestro servidor.

Si se usan las herramientas, se generan las condiciones de recuperar la información por cualquier contingencia en el menor tiempo posible, además de seguir confiando en la información así como detectar los posibles problemas dentro y fuera del servidor.

Al hacer uso de las herramientas de seguridad, se agrega un nivel mayor en la protección de la información en el sistema, debido a que hay una variedad de herramientas que por su uso las distinguen unas con otras, el beneficio de estas herramientas es la generación de confianza a todos los usuarios del sistema, además que ayuda al administrador para enfrentar cualquier contingencia en el futuro.

En el próximo capítulo se explicarán las herramientas para proteger la información en el servidor y en canales inseguros, estas herramientas utilizan los algoritmos vistos en este capítulo. Se estudiará la importancia de usarlos y la relación con la seguridad en el sistema.

# CAPÍTULO 6.

# RECOMENDACIONES.

## 6.1. Introducción.

Una vez que se ha estudiado todas las posibilidades que nos ayudan a mantener un nivel de seguridad informática optima para la protección de la información, ahora se estudiarán 5 esquemas, teniendo en cuenta de que estos esquemas responden a ciertas necesidades particulares de la empresa Gamela México S.A. de C.V., y que pueden existir otras necesidades distintas, para las cuales una combinación de estos 5 esquemas podrán solucionar los problemas a situaciones distintas a las planteadas en este capítulo.

Como se ha comentado anteriormente, las recomendaciones parten desde la aplicación de 1 o combinación de más esquemas, según las necesidades le una organización así lo exija, agregado a una serie de medidas (*"Recomendaciones Generales"*) y un plan de contingencias permitirán mantener protegida la información desde el servidor, incluyendo cuando viaje a través de canales inseguros.

El objetivo de este capítulo es que el lector distinga que cada herramienta fue diseñada para solucionar necesidades específicas aplicables a un sistema de Información, por lo cual una buena solución dependerá en gran medida de las necesidades de la organización y una vez identificando estas necesidades ubicarlas al esquema que más acorde se encuentre. Esta es la parte dinámica de las soluciones, porque la decisión de utilizar dichas herramientas dependerá de las necesidades que así lo exija el sistema de información.

En el tema llamado "Recomendaciones generales" y el plan de contingencia se deben aplicar siempre, en cualquier esquema de seguridad informática, porque son los problemas más comunes que se han observado, y es lo que ayudará a mantener una cultura de seguridad informática. No es recomendable dejarle todo el trabajo a las herramientas de seguridad, sino equilibrar una serie de reglas y políticas para todos los usuarios, incluyendo a los administradores del sistema para hacer un buen uso de los equipos, y no debemos olvidar que:

*"La seguridad la hacemos todos".*

## 6.2. Aplicación de las herramientas de seguridad.

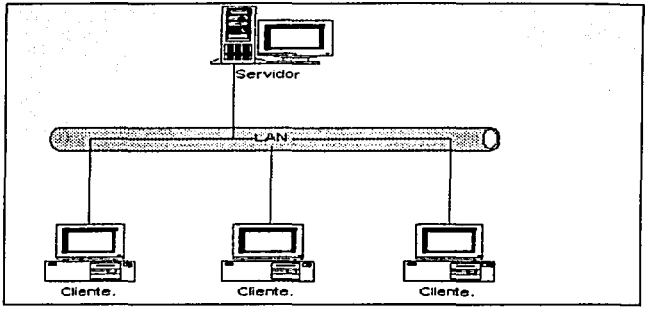
Las medidas que deben tomarse para que una organización tenga un nivel de seguridad Informática óptima está en función de variables como: el tipo y la cantidad de servicios que ofrece el servidor en la red o redes, el nivel de conocimiento de los usuarios de la red, así como la índole del giro de la empresa y el costo que representan dichas medidas[65]. Son factores que en varios casos definen el nivel de seguridad Informática en las distintas compañías.

No es posible implantar una solución Informática igual para todas las empresas para satisfacer sus necesidades de seguridad, ya que cada empresa puede atravesar por una problemática distinta. Es por ello que analizaremos cinco esquemas; surgen desde el esquema más sencillo que es un Servidor UNIX con pocos servicios y necesidades y que puede coexistir en una red LAN hasta llegar a un esquema mucho más complejo donde los servicios del servidor aumentan exigiendo una serie de medidas más sofisticadas.

[65] El costo esta enfocado en cuanto al rendimiento del sistema frente a las herramientas que apliquemos, por ejemplo implantar mecanismos para codificar representa un costo para el procesador porque éste ejecuta tareas para codificar la información.

### 6.2.1. CASO 1.

Figura 6.1. Servidor que recibe las peticiones de una red LAN, donde no hay servicios de Internet.




En este caso debemos proteger a la red de:

a) Los accesos ilegales al sistema. Que pueden ser de los miembros de la red LAN.
b) Ataques de interceptación del tráfico de la red local.
c) Ataques en la integridad del servidor, por ejemplo mediante caballos de Troya.
d) Hoyos de seguridad en nuestro servidor.

Las herramientas recomendadas para que el administrador de la red implemente son:

- Secure Shell (Secure Shell). Al usar SSH, cifra el tráfico y puede hacer los ataques "hombre en el medio en la red LAN" casi imposible. También lo protege de DNS y IP spoofing. Como plus, ofrece la posibilidad de comprimir el tráfico y por lo tanto hacer las transferencias más rápidas. SSH es una herramienta muy versátil, no sólo reemplaza a telnet, puede también "abrir un túnel" a servicios como ftp, pop e incluso ppp a través de él. Es muy útil, porque permite utilizar algoritmos para codificar como DES, 3DES, IDEA, Blowfish etc., proporcionando los mecanismos necesarios para lograr una comunicación segura sobre un canal abierto, de esta forma toda la información que fluye de computadora a computadora viaja siempre codificada brindando un nivel de seguridad óptimo a la información que viaja a través de ella.
- Tripwire. Esta Herramienta permitirá monitorear el estado del sistema en determinadas fechas comparando la información actual del sistema contra la información del sistema original o en la "información del sistema confiable[66]" y si ésta información es la esperada por el administrador del sistema entonces la información generada será almacenada, preferentemente, en un dispositivo externo (es recomendado que se guarde en una cinta o CD-ROM) o en un archivo en el servidor. ¿Cada cuánto tiempo debemos aplicar esta herramienta? Esto depende de factores como: disponibilidad de tiempo, así como del presupuesto con que contemos. Una

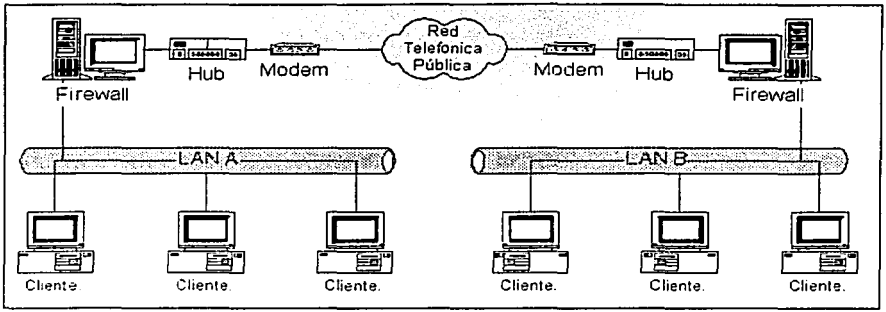
[66] Al referirnos a la información del sistema confiable, nos referimos a la información original del sistema al momento de instalar el Sistema Operativo, que por lo general se guardan en un CD-ROM, o también la información que en una fecha posterior a la instalación del sistema operativo ya hemos comparado contra la información original del sistema cuyos cambios del mismo, en este lapso de tiempo, son los esperados por los administradores del sistema, razón por la cual, la información que comparamos contra el original se puede considerar como "información confiable" y así sucesivamente.

buena alternativa es programar un cron todos los viernes a las 24 horas y si encuentra un cambio no esperado en archivos del sistema, que nos notifique a nuestro correo.
- El administrador de la red debe aplicar las Recomendaciones Generales (Estas medidas se explican en el tema: Recomendaciones Generales[67]).

Nota: Esta solución no representa un costo considerable para el sistema.

### 6.2.2. Caso 2.

Figura 6.2. Modelo donde dos redes LAN A y LAN B, comparten sus recursos, y sus servidores deben atender las peticiones de sus clientes así como intercambiar información.




En este esquema, vemos la red LAN A, donde un servidor debe atender las peticiones de su propia red, así mismo debe compartir recursos con la otra red LAN B. Debemos proteger a la red de ataques como:

a) Se aplican los incisos a,b, c y d del caso1.
b) Ataques de miembros de las redes LAN A y LAN B (insiders), o fuera de la redes locales (outsiders).
c) Ataques de intermediario, donde la información que fluye entre ambas subredes puede ser interceptada por una persona ajena.

Las herramientas recomendadas para que el administrador de la red implemente son:

- Secure Shell.
- Tripwire.
- PGP. Esta herramienta ayuda a codificar la información que es considerada como delicada, y sólo quién(es) posea(n) la clave con que se codifico podrá ver el contenido de la información.
- Firewall. No es muy usual en estos casos, sin embargo es la opción que permite que un administrador de la red pueda confiar en la información que recibe, mediante paquetes de información, de las otras máquinas. Este tipo de casos se presenta muy frecuente cuando un cliente, proveedor, generan una comunicación como para transferir información financiera a la empresa.

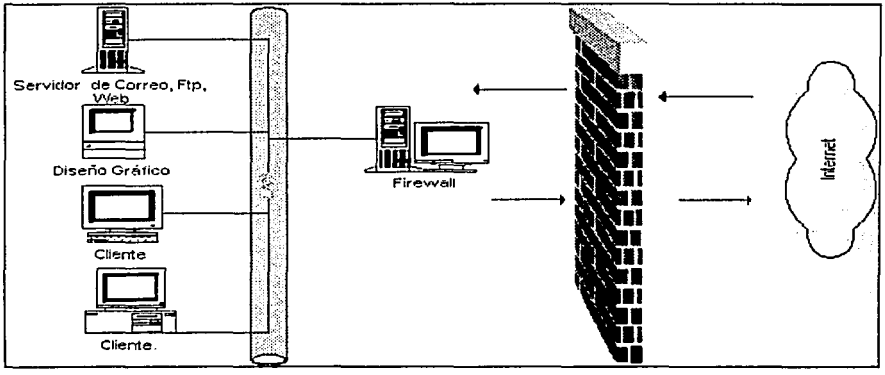
[67] Las recomendaciones generales se aplican para cualquier esquema de seguridad, se explican en este mismo capítulo en el tema llamado "Recomendaciones Generales".

- Nessus. Este escaneador de puertos es útil para saber que puertos abiertos tiene el servidor, con base en esta información se puede cerrar los puertos que no sean necesarios para el servidor en el archivo /etc/services.
- Otra muy buena recomendación es configurar al servidor o al firewall que solo acepte conexiones de las direcciones IP que interactúan con nuestra red, para el caso de Gamela México sus direcciones IP tienen un rango desde la 200.39.102.1 a la 200.39.102.255, por lo cual en el archivo /etc/hosts.allow se edita una línea como la siguiente: ***ALLOW:200.39.102.*** con esto se le dice al servidor que acepte las conexiones que provengan del rango de direcciones IP de Gamela México, y después se negarán todos los intentos de conexión que no pertenecen al rango de Gamela México, para ello se edita el archivo /etc/host.deny agregando una línea como la siguiente: ***ALLOW:ALLOW***. De esta manera se le indica al servidor que no acepte ninguna conexión de ninguna dirección IP que no pertenezca a las especificadas en /etc/hosts.allow. Aplicando estas medidas se puede controlar los accesos en la red local o en un conjunto de redes locales.

El administrador de la red debe aplicar las Recomendaciones Generales (Estas medidas se explican en el tema: Recomendaciones Generales).

### 6.2.3. Caso 3.

Figura 6.3. Esquema donde una red LAN necesita de servicios de Internet, además de satisfacer las necesidades de sus propios clientes (correo electrónico, transferencia de archivos, servidor Web).




Para este modelo se debe protegernos de ataques como:

a) Los ataques de los incisos a, b, c y d del caso 1 así como los incisos b y c del caso2.
b) SPOOFING (Pueden hacer uso de nuestro servidor para atacar otros servidores).
c) FLOODING (Negaciones de servicio, por ejemplo mediante la saturación de la red).

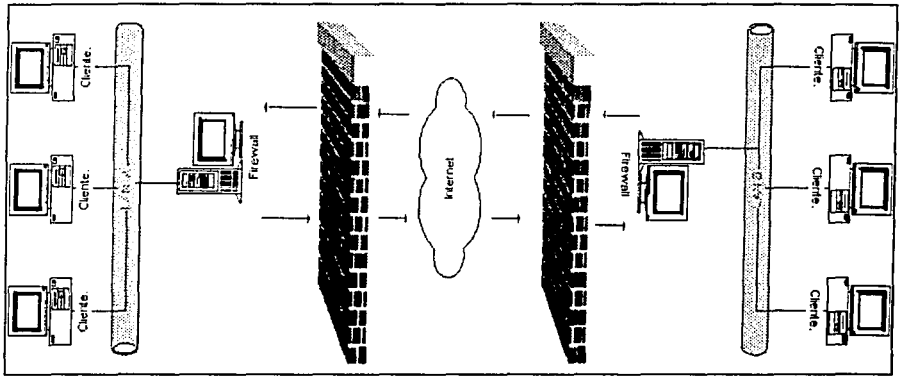
Para solucionar ésta problemática podemos aplicar las siguientes técnicas:

- Secure Shell.
- Tripwire.

- PGP.
- Delimitación los rangos de accesos para las direcciones IP a nuestra(s) red(es) mediante los archivos /etc/hosts.allow y /etc/hosts.deny.
- Firewall. El Firewall en estos esquemas es muy importante. El firewall emplea reglas de acceso para todos los paquetes que ingresan a la red, rechazando aquéllos paquetes que los administradores consideren que son un riesgo para la Red Local.
- Debemos utilizar escaneadores[68] hacia nuestra Red Local. Estos escaneos son de gran utilidad, porque muestran la vulnerabilidad de la red. Los escaneadores como SATAN, NMAP, etc., tienen entre otras funciones hacer peticiones a todos los puertos y con base a la respuesta de los puertos determina cuán tan vulnerable es, y si determina que puede ser causa de problemas, muestran los resultados para que el administrador analice la situación, incluso SATAN muestra páginas del CERT en Internet donde se han detectado problemas similares y sus respectivas soluciones para que el administrador pueda estudiar el caso. Los resultados de estas herramientas son excelentes. Las Herramientas recomendadas son: NESSUS, NMAP, SATAN.
- Crack. La principal función es la de romper los password del sistema y determinar un nivel de seguridad en los passwords, para ello utiliza un barrido por la fuerza bruta con sus diccionarios, para definir los passwords débiles. Hay que estar conscientes de que estas herramientas se pueden utilizar tanto para beneficio del administrador de la red como para perjudicar al mismo; porque puede ser usado por cualquier usuario; para encontrar debilidades en el sistema y atacar esas debilidades. De hecho es así como el 60% de los ataques son generados (publicación hecha por el CERT).
- El administrador de la red debe aplicar las Recomendaciones Generales (Estas medidas se explican en el tema: Recomendaciones Generales).

### 6.2.4. Caso 4.

Figura 6.4. Este esquema es más complejo. La red LAN A. Requiere de servicios de Internet (páginas Web[WWW]), transferencia de archivos [Ftp], correo electrónico [Mail], Secciones remotas [Telnet]) Así como transferencia de información a través de Internet hasta la red LAN B. También cada servidor atiende las peticiones de sus clientes.




[68] Las herramientas de seguridad en servidores son por ejemplo, Nmap, Portsentry, Nessus, etc.

Para este modelo debemos protegernos de ataques como:

a) Se aplican los incisos a,b,c y d del caso 1, los incisos b y c del caso 2, los incisos b y c del caso 3.
b) La integridad de la información a través de Internet.
c) La confidencialidad de la información.
d) La autenticidad de la información.

Observe que la necesidad brindar más servicios cambia también algunos aspectos del servidor, por ejemplo, nos obliga a abrir más puertos, para el servicio Web con Perl se necesita permitir ejecutar programas desde cualquier cliente en Internet al servidor y se debe permitir dicho acceso, y estas situaciones generan una mayor vulnerabilidad al servidor.

Contra estos problemas podemos usar las siguientes herramientas y medidas:

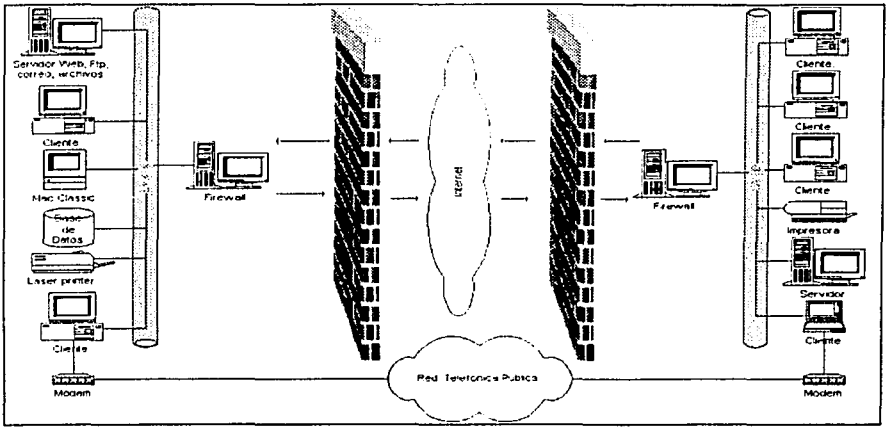
- Secure Shell.
- Tripwire.
- Firewall.
- PGP.
- Delimitación los rangos de accesos para las direcciones IP a nuestra(s) red(es) mediante los archivos /etc/hosts.allow y /etc/hosts.deny.
- Escaneadores como NESUS, NMAP o SATAN.
- Secure Socket Layer. Este programa codifica todos los mensajes que son generados en el servidor y posteriormente lo envía de la red LAN A hacia otra red LAN B a través de un medio como Internet. Este programa emplea algoritmos para codificar como RSA, DES, 3DES, IDEA, (SSL).
- También es recomendable usar firmas digitales, su principal función es informar si un mensaje ha sido interceptado en Internet. Esta herramienta usa el algoritmo de Hash y consiste en generar una clave, de esta forma si el mensaje fue interceptado y trataron de verlo, la llave del Hash cambia y con ello cambian todos los valores del Hash. De esta manera un usuario final puede darse cuenta que el mensaje fue alterada o no. Un clásico riesgo que resuelve el uso de esta herramienta es evitar los caballos de Troya puesto que existe la certeza de que no ha sido modificada la llave.
- El administrador de la red debe aplicar las Recomendaciones Generales (Estas medidas se explican en el tema: Recomendaciones Generales).

### 6.2.5. Caso 5.

Cada esquema representa un tipo de problemática especial, son modelos donde la complejidad aumenta según las necesidades de proporcionar más servicios. El quinto caso es aquel que un administrador ajusta a sus necesidades, donde uno o más modelos de los propuestos coexisten (Caso específico de Gamela México S.A. de C.V.).

Los ataques que tenemos que proteger son todos los incisos vistos en los casos 1, 2, 3 y 4, porque se necesita respaldar la información de todas las computadoras de la compañía en un servidor de respaldo de archivos, este servidor debe estar aislado de Internet y sólo debe atender las peticiones de la red LAN de la compañía (Caso 1), la compañía también necesita realizar operaciones financieras con el banco, clientes, proveedores, para ello requiere de una comunicación exclusiva entre la red de la compañía con las redes involucradas, esto se logra mediante una comunicación entre dos Módem (Caso 2), necesita brindar a sus empleados servicios de correo electrónico, mantener un sitio Web, ofrecer servicio de Internet (Caso 3), finalmente la compañía necesita compartir información y sus recursos con otra red en Panamá y en Estados Unidos (caso 4).

Figura 6.5. Esquema para Gamela México.




Las herramientas que dan solución a estas necesidades son:

- Secure Shell.
- Tripwire.
- Firewall.
- Escaneadores como SATAN, COPS o NESSUS.
- Secure Socket Layer.
- Tcp-wrappers.
- Delimitación los rangos de accesos para las direcciones IP a nuestra(s) red(es) mediante los archivos /etc/hosts.allow y /etc/hosts.deny.
- También es bueno usar Firmas digitales.
- El administrador de la red debe aplicar las Recomendaciones Generales (Estas medidas se explican en el tema: Recomendaciones Generales).
- PGP.

Como se puede observar, las necesidades que exige la compañía son cubiertas por las herramientas que ya estudiamos, agregados a las recomendaciones generales, dando solución a toda una serie de problemas que se generan dentro y fuera de la red.

Para concluir este tema llamado aplicación de las herramientas de seguridad, haremos notar que pudimos emplear hardware para codificar la información, pero en realidad no hubo necesidad por las siguientes razones:

- Recursos suficientes para el servidor. Dos procesadores Pentium III Xeon, a 1024 megabytes de memoria RAM y velocidad de 500 Megahertz, con un disco duro de 40 gigabytes.
- Medios de transmisión. Se cuenta con fibra óptica, 1 medio E4, brindando una capacidad de transmitir gran cantidad de información a una velocidad muy rápida.
- Cantidad de información. La red no codifica grandes volúmenes de información para enviarla fuera de la red.

## 6.3. Recomendaciones generales.

A continuación se recomendará una serie de pasos para la prevención futura de intrusiones en cualquier sistema, de igual forma se sugiere aplicar las que más se adecuen a las necesidades del sistema y requerimientos de los usuarios.

- Actualización del software del sistema (parches de seguridad). Mantener actualizado el sistema (para el caso de Linux mediante los rpm[69]), la mejor recomendación es mantener las penúltimas versiones ya que las nuevas versiones están en versión beta y son susceptibles de fallos. (En Linux para actualizar un programa hay que aplicar el comando: rpm –i nombre_del_programa)

- Usar Tripwire y guardar en un medio seguro, en un CD-ROM, el estado original de algunos archivos importantes, por ejemplo los archivos binarios ubicados en /bin y otros del sistema.

- Bloqueo de puertos. (La versión de Linux de Mandrake 8.1 y OpenBSD ya se instalan con los puertos predeterminadamente cerrados y conforme el sistema necesite de más puertos, el administrador los tendrá que abrir.)

- Realización de respaldos periódicamente. Esto es de gran ayuda pues nos permiten recuperar el sistema en caso de alguna contingencia. Esta decisión depende en gran parte de la información que se genera, por ejemplo la información de la base de datos por lo general se respalda diario, existen casos donde la información se genera cada cierta fecha, aquí se debe aplicar un calendario y con base a este realizar los respaldos..

- Consideración de la instalación de un firewall.

- Educación de usuarios y administradores. (Información de los beneficios de usar las herramientas instaladas)

- Instalación y configuración de herramientas de seguridad tales como SSH, SSL, PGP, TRIPWIRE, COPS, TCP-WRAPPERS, NMAP, etc.

- Reducción en el uso de programas como telnet y ftp y cambiarlos paulatinamente por ssh.

- Eliminar el uso de programas como rtelnet, rftp, rlogin y los programas que comienzan con "r" en /etc/services (Esto se logra comentando estos servicios en el archivo /etc/services).

- Usar contraseñas seguras (por lo menos 6 caracteres alfanuméricos y no máximo de ocho caracteres). En el cuadro 6.1 hay una contraseña incorrecta (Esmeralda8) porque es una contraseña basada en una palabra de diccionario. Si aplicamos el programa de CRACK, este será capaz de encontrar la contraseña.

  El algoritmo para hacer una contraseña más segura es:

  1. Hacer una frase, dicha frase debe representar un recuerdo agradable, que sea fácil de recordar ("Recuerdo mi primer empleo como programador, año 2000").
  2. Tomar la primera letra de cada palabra (R, m, p, e, c, p, a, 2.).
  3. Juntar todas las letras recopiladas (Rmpecpa2).
  4. La contraseña generada es: Rmpecpa2[70].

[69] En Linux un RPM es un programa, por ejemplo un firewall, disponible en Internet el cuál sirve para instalar dicho programa en el sistema.
[70] En UNIX las mayúsculas y las minúsculas son letras distintas.

Las ventajas que presenta esta contraseña son:

A) No es una palabra basada en diccionario, por lo que programas como CRACK no podrán encontrar esta contraseña.

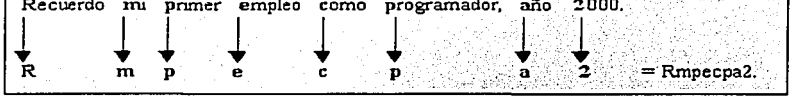
B) Cuando accedemos al sistema en presencia de más usuarios, si éstos la llegan a ver, les será más difícil retener la contraseña, ya que no es una palabra común.

C) Con este tipo de algoritmos no es necesario escribir en papel o algún otro medio la contraseña, porque al recordar nuestra frase preferida (que nunca olvidamos) obtenemos nuestra contraseña.

Cuadro 6.1. Algoritmo para construir una contraseña segura.

Contraseña mal elaborada (passwd): esmeralda8

Contraseña bien elaborada:




- Utilizar sistemas académicos o comerciales para la detección de intrusos (como AAFID) o (como Network Flight Recorder).
- Recompilar syslogd para leer un archivo de configuración diferente a /etc/syslog.conf y enviar las bitácoras a un servidor remoto (de preferencia que este servidor sea dedicado)
- Implementación de políticas de seguridad (Sistema de cuotas en disco, desconectar una sección en un tiempo máximo de 15 minutos en los que una terminal no tenga actividad, etc.).
- Desinstalación de compiladores y otros programas que no se necesiten.
- Los puertos, cuentas y servicios de red realmente necesarios serán activados y configurados adecuadamente.
- En su caso, configurar de manera segura el servidor de FTP anónimo.
- Asegurarse de tener las últimas versiones de sendmail, ftpd, BIND, httpd, fingerd, herramientas de seguridad, majordomo[71], etc.
- Revisar constantemente y de manera impredecible las bitácoras del sistema y las generadas por las herramientas de seguridad instaladas (por lo menos cada semana).
- Validar el acceso del sistema, que sólo permita tres oportunidades para accesar al sistema, si después de estos tres intentos no logra entrar (logearse) al sistema, la cuenta deberá suspenderse y deberá agregarse a las bitácoras de revisiones de conexión.
- Investigar y probar software diferente a los tradicionales, por ejemplo el servidor de correo Qmail es mucho más eficiente que sendmail, incluso la organización propietaria otorga una

[71] Estos programas ofrecen un servicio en especial, por ejemplo, majordomo es una lista de correo.

recompensa de 10,000 dólares a quién encuentre una falla en Qmail. Este tipo de investigaciones es lo que permitirá mantener actualizado al administrador de sistemas.

## 6.4. Respaldos.

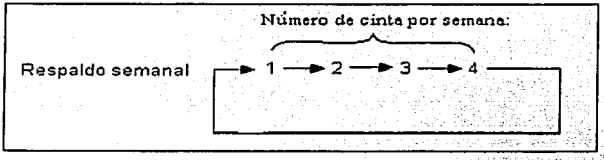
Es también, muy necesario establecer políticas para efectuar los respaldos. Estos no se deben llevar a cabo solo cuando se tenga información importante. Los respaldos deben ser efectuarse de acuerdo a políticas y calendarios lógicamente establecidos, de manera que se pueda contar con nuestra información en cualquier momento.

En los servidores UNIX, los sistemas de compresión y respaldo permite extraer y anexar información a un archivo de respaldo anteriormente creado. De esta forma, se puede regenerar completamente dicha información. Al igual que se puede agregar nuevos archivos al respaldo, de igual forma se puede seleccionar solo aquellos archivos de interés para recuperar. No es necesario restaurar al backup completo para obtener los archivos que se desean.

Una buena estrategia de respaldo es hacer un respaldo total del sistema una vez por semana. Este respaldo puede ser almacenado en un determinado número de cintas.

Si por hay 4 cintas, se puede utilizar una cinta por semana y reutilizarlos en orden cronológico.

Figura 6.7. Estrategia mensual de respaldo con cuatro cintas:




El uso de varias cintas permite tener respaldos de semanas anteriores en caso de necesitar su información. Con esta estrategia nos aseguramos de contar con la información de todo un mes en cualquier momento que sea necesario.

Cabe hacer notar que los sistemas de respaldo (backup) requieren muchos recursos del sistema, el respaldo total de archivos es difícil de ser llevado a cabo cuando se acumulan varios días, por eso es mejor hacer respaldos día con día (los días de la semana 1 corresponde a la cinta 1).

Para respaldar la información diaria muchos sistemas permiten agregar a los archivos de backup solamente la información nueva, es decir los últimos cambios. A esta estrategia se le conoce como respaldos incrementales. Para crear estos respaldos se puede ejecutar comandos que seleccionen los archivos que han sido modificados durante el día por ejemplo con el comando find permite buscar archivos en base a su fecha de modificación y una vez seleccionada la información del día se abren los archivos de backup para añadir la nueva información. Este proceso será mucho más rápido que el respaldo total que se lleva a cabo 1 vez por semana, y así se obtendría la información respaldada del día anterior sin problema alguno para recuperarla.

Otra opción es hacer los respaldos, por ejemplo, es en modo de procesamiento batch[72] para que sean ejecutados a las 0:00 horas del día, siguiente, de esta manera es posible asegurar que se realice todo el respaldo de los archivos modificados sin que afecte el rendimiento del sistema.

[72] Programación de una actividad en el sistema, con una frecuencia, en días y hora determinada (comando cron) o un cierto día en una hora especifica (comando at).

Otro aspecto a considerar es que la información que deseamos respaldar no es solamente la contenida en nuestra base de datos. El respaldo de los archivos del sistema operativo también es muy importante ya que de sufrir algún ataque donde perjudiquen la configuración inicial del sistema, los respaldos dan la posibilidad de restablecer la configuración que se tenía anteriormente. Los archivos del sistema operativo incluye también la estructura de directorios del sistema[73], archivos de grupos, hosts, de usuarios y logs. Si no se contara con el respaldo del sistema operativo tendríamos que acudir a los CD's de instalación de manera que perderíamos los parámetros de configuración que teníamos anteriormente.

En el caso de información histórica[74] de una compañía, es muy recomendable que se utilizen dispositivos de solo lectura como el CD ROM para almacenar información que estará disponible por mucho tiempo. Con el uso de estos dispositivos se garantiza que no será alterada la información.

Cuando se sufre la pérdida de archivos importantes, debe recuperarse la información almacenada en los respaldos. Si el tiempo es un factor importante, entonces una buena manera de recuperar la información es reinstalar solo los archivos que fueron modificados y no todo el respaldo completo que tomaría mucho más tiempo. Esto será posible lograrlo si tenemos la herramienta de tripwire en nuestro servidor, tripwire ayudará a realizar una comparación del sistema actual contra la información original del sistema después de la instalación del sistema operativo (si es la primera vez que se ejecuta tripwire), o contra la información más confiable del sistema (última vez que se analizó el sistema con tripwire) de esta forma podemos ver los beneficios que nos ofrece tripwire.

[73] Estas estructuras fueron tratadas en el capítulo 2.
[74] La información histórica es la que se ha generado con un lapso de tiempo considerable por ejemplo, la información de un determinado año.

## 6.5. Plan de contingencias.

### 6.5.1. Lo primero que se recomienda hacer es mantener la calma.

En ocasiones un administrador del sistema piensa que ha entrado alguien al sistema, pero en muchas ocasiones esto no es cierto. En la mayoría de los casos, es fácil saber si alguien a entrado al sistema, ya que estos siguen un patrón y es posible rastrearlo en la mayoría de los casos.

Si algún intruso entra al sistema y sólo entra como usuario normal, intentará explotar algún fallo del sistema para obtener UID=0 ó lo que es lo mismo, privilegios de superusuario. Ahora si él intruso logra accesar como superusuario intentará controlar el sistema, dejando mecanismos para volver cuando así lo desee, por ejemplo: copiará el archivo /etc/passwd y el /etc/shadow (En caso de que utilice Shadow), también puede instalar un sniffer, troyanos, leer correos ajenos, etc. En el peor de los casos modificaría páginas Web, borraría archivos o los robaría, produciría negaciones de servicios (Denial of Service), cambiaría passwords, etc.

### 6.5.2. Revisión de bitácoras.

Debemos revisar las bitácoras del sistema, por si no hay alguna herramienta instalada. (Lo cual es un grave error) Las bitácoras son una gran ventaja, pero con frecuencia son ignoradas o no conocen su uso. En éste trabajo ya se ha explicado como sacar beneficio de la información que generan estas bitácoras.

Lo primero que debemos hacer es un plan. (Mediante las herramientas y consejos propuestos en los capítulos 2 y 3 buscar rastros de ataques y posteriormente tomar decisiones).

El segundo paso es identificar que bitácoras contienen esa información.

A) Revisión de los accesos a la cuenta comprometida (utilizando last)
B) Revisión de Archivos de Bitácoras:

- syslog.
- sulog.
- messages.
- maillog.
- xferlog.
- secure.
- tcpdlog (sí se cuenta con ella).

Observe, que en diversas ocasiones los intrusos modifican las bitácoras, por lo que no podemos confiarnos totalmente en la lectura de las mismas. Frente a este tipo de situaciones es cuando resulta de gran utilidad contar con herramientas como Cops, Tiger, Portsentry[75], Tripwire que tienen la capacidad de enviar información de los sucesos en el sistema en bitácoras, dispositivos externos, incluso otra cuenta de correo. El uso de las herramientas pueden evitarnos invertir mucho tiempo e implementar inmediatamente la recuperación del sistema.

### 6.5.3. Iniciar una búsqueda de archivos ocultos.

[75] Programa que esta formado de otros programas cuya función es entrar todos los puertos como sea posible en un servidor UNIX.

Buscar los archivos ocultos del sistema (aquellos que empiezan con punto) ya que se pueden utilizar para esconder herramientas que rompen la seguridad del sistema, por ejemplo un programa crack o también contener el /etc/passwd del sistema o de otros sistemas a los que ha entrado el intruso. Muchos intrusos suelen crear directorios ocultos utilizando nombres como "..."(punto punto punto), ".."(punto punto), "..^g"(punto punto control+g). También se dan casos en los cuáles utiliza nombres como ".x" o hasta ".mail".

- buscar archivos ocultos o inusuales en el sistema (.*).

```
find / -name ".*" -print
```

- Buscar programas que adivinan contraseñas (crack, John The Ripper, Cracker Jack, Hades, etc.) y que no hayan sido colocados por el administrador, para eliminarlos.

```
find / -name \( -name "*crack*"-o –name "*Crack* \) -print
```

6.5.4. Buscar archivos setuid.

Se debe buscar cuidadosamente archivos SETUID o SETGID (especialmente los que pertenecen a root). Para eso podemos utilizar, el comando find.

```
find / -group wheel -perm -2000 -print
find / -user root -perm -4000 -print -xdev
ncheck -s /dev/rsd0g
```

Este ultimo comando "ncheck" nos permite buscar archivos SETUID por particiones.

6.5.5. Revisar archivos binarios.

Buscar archivos que contengan caballos de troya en los archivos binarios (programas). Esta es una de las tareas principales de un intruso cuando ha comprometido la seguridad de un servidor; por ejemplo: login, ls, su, find, telnet, du, df, netstat, ifconfig, libc, sync, w, who. En estos casos es muy útil contar con programas como Tripwire.

6.5.6. Revisar los archivos que son ejecutados por "cron" y "at".

Algunos intrusos depositan puertas traseras mejor conocidas como backdoors, que les permitan volver al sistema aunque se les niegue el acceso al mismo. Hay que asegurarse que todos los archivos que están programados por el archivo del cron sean legítimos. Para saber con que archivos trabajan los comandos cron, se puede usar el comando *"cron –l >/tmp/arch.txt"* con esto se le dice al programa cron que muestre su archivo de configuración y lo diriga a un archivo, después se tendría que analizar dicho archivo para ver si los programas que ejecuta el cron son validos.

6.5.7. Busqueda de Sniffers (capturadores de red).

Tenemos que buscar sniffers, ya que esta es una de las opciones favoritas y más utilizadas por los intrusos. Los sniffers son utilizados para capturar todo el tráfico de nuestra red, incluyendo las sesiones de ftp y telnet hacia otros sistemas.

- Buscar sniffers (linsniff, esniff, solsniff, sunsniff, sniffit, etc.)

```
find / -name "*sniff*" -print
```

Escuchando el tráfico de la red, un atacante puede obtener cuentas de usuarios (logins) y passwords. Es posible ver el sistema en modo promiscuo[76] en /var/log/messages, por ejemplo en Linux se vería de la siguiente forma:

```
Apr 27 17:03:38 mozart kernel:eth0:Setting promiscuous mode
Apr 27 17:03:43 mozart kernel:eth0:Setting promiscuous mode
```

Algunos de los sniffers más conocidos son los siguientes:

- linsniff666.c
- sunsniff.c
- esniff.c
- sniffit
- solsniff.c

6.5.8. Revisión del archivo /etc/inetd.conf

Hay que buscar en especial entradas que ejecuten un shell (por ejemplo: /bin/sh o /bin/csh) y comprobar que todos los programas son legítimos y no troyanos. De igual forma hay que revisar que los demonios de los programas como telnet, ftp y todos los servicios que preste el servidor sean legítimos.

6.5.9. Búsqueda de alteraciones en el sistema y en los archivos.

En especial hay que buscar entradas con el signo "+" o servidores de máquinas no apropiados en archivos como /etc/host.equiv, /etc/hosts.lpd y en todos los archivos .rhost del sistema, con especial interés los de root, uucp y ftp. Estos archivos no deberían tener atributo de escritura.

6.5.10. Examinar los equipos de nuestra red local.

Hay que buscar indicios de que la red ha sido comprometida. En particular aquellos equipos que compartan NIS o NFS[77] o aquellos sistemas listados en el /etc/hosts.equiv. Lógicamente también revisar los sistemas que los usuarios comparten mediante el acceso del .rhosts.

6.5.11 Revisar el archivo /etc/passwd.

Hay que buscar alteraciones en las cuentas de los usuarios o la creación de cuentas nuevas, especialmente aquellas con UID 0, las que no tienen password, etc. Otra cosa que hará el intruso es obtener la tabla de passwords, sobre la cual ejecutará un programa para buscar passwords débiles y así obtener más cuentas para entrar al sistema.

6.5.12. Utilizar el comando finger.

Ejecutando el comando "finger" se puede sacar información del intruso así como el origen de la intrusión, por ejemplo:

```
finger @intruso.net
finger intruso@intruso.net
```

Si tenemos suerte podremos sacar información de la máquina de la cual provino la intrusión.

6.5.13. ¿Qué hacer si el intruso se encuentra en el sistema?

---

[76] Modo promiscuo en ámbito de seguridad en computo, se refiere a que la información que viaja en la red esta en estado de texto claro.

[77] Sistema de archivos compartidos en red.

A continuación se señalan algunas recomendaciones a seguir cuando detectemos a un intruso en sesión:

- Asegurarse de tener respaldos recientes y en buen estado.
- Ignorarlo, tratar de hablar con él (write, talk) o monitorear sus actividades (tcpdump, snoop).
- Sacarlo del sistema (matar sus procesos, cambiar su contraseña, desconectar la máquina de la red, tirar los servicios de red).

```
ps [-fea | aux] | grep cuenta_intruso | grep -v grep | xargs kill -9
```

- Rastrearlo usando los comandos who, w, last, lastcomm, netstat, whois, nslookup, finger, telnet, strings /var/adm/lastlog, /var/adm*, obtener información del ruteador, examinar los archivos de historia ($HOME/.history, $HOME.sh_history),etc.
- Si existe un teléfono de contacto, llamar al encargado y hacerle saber el problema. NO USAR Correo electrónico, a menos que sea la única opción, en cuyo caso se deberá ser discreto).
- Contactar a otros administradores involucrados y dar aviso al CERT[78].

6.5.14. La recomendación final es que todas las fallas detectadas debemos corregirlas inmediatamente. A lo largo de este trabajo se ha explicado como poder corregir las vulnerabilidades del sistema, y también instalar las herramientas que se han sugerido, si es el caso de que no había ninguna herramienta instalada, se asume también que un respaldo de la información del sistema se realiza siempre.

## 6.6. Conclusión.

Como ya se análizo a lo largo del capítulo el uso de las herramientas de seguridad informática combinadas con una serie de métodos y políticas, ayudan a mantener un nivel de seguridad adicional en cualquier sistema UNIX, lo cuál prepara una cultura de seguridad informática y un ambiente para que los usuarios puedan trabajar con confianza en el sistema.

Al no emplear las herramientas de seguridad, ni las políticas o los métodos descritos en este capítulo, se expone tanto al sistema con toda la información que mantiene, ante varios ataques informáticos.

Un sistema que emplea herramientas de seguridad tiene la capacidad de enfrentar cualquier clase de ataque infomático, una serie de políticas y métodos proporcionan una buena administración del sistema y nos prepara para enfrentar cualquier problema generado la mala administración, mal uso, o cualquier otro problema relacionado con la seguridad del sistema.

Al emplear las herramientas, técnicas y políticas mencionados en este capitulo, no se garantiza una seguridad absoluta por siempre, pero se crean las condiciones necesarias para enfrentar cualquier problema relacionado con la seguridad informática así como un mejor uso para que sus usuarios puedan trabajar con la información ayudando con esto a cumplir con los objetivos de la organización.

[78] El CERT es un organismo de seguridad en computo de nivel internacional, el CERT en México es el organismo que en casos de ataques, el CERT registra estas actividades y las reporta a listas de seguridad en computo donde hay personas que pueden asesorar directamente a la persona con el problema.

# CONCLUSIÓN:

Emplear una cultura de seguridad informática, es brindar tranquilidad y confianza tanto a los usuarios del sistema como a quienes son responsables del mismo, ya que contar con información confiable, oportuna y verídica es uno de los principales requisitos que debe cumplir la información para que sea útil. Los administradores de sistemas deben estar conscientes de que la información puede ser obtenida por una persona dentro o fuera de la organización y existe el riesgo de que dicha información pierda su integridad o pueda ser borrada.

Es muy importante que un administrador del sistema comprenda la importancia de mantener con orden el sistema de archivos, pues esto facilita la elaboración de respaldos estratégicos que reflejen la parte crítica del sistema y permita hacer respaldos que servirán para recuperar al sistema en caso de que el sistema sea comprometido en su seguridad. También es recomendable que se realice un calendario para auditar al sistema con las herramientas de seguridad, de esta forma se puede monitorear las actividades del servidor y permite mantener un control sobre el mismo.

Las herramientas estudiadas en este trabajo tienen por objetivo ofrecer al administrador del sistema un control sobre el mismo, dichas herramientas son de diversas índole:

- Las que controlan accesos como por ejemplo un firewall, tcp – wrappers, etc.
- Las que revisan la integridad del sistema como cops, tiger, tripwire, etc.
- Las que brindan seguridad en las comunicaciones como ssl, ssh, etc.

Mediante el uso de estas herramientas no se garantiza que el sistema siempre será seguro sin embargo se generan los mecanismos necesarios para que el administrador del mismo pueda mantener un control y sobre todo recuperar la información del servidor en caso de sufrir un ataque informático.

Parece que entre más redundante es la seguridad del sistema, puede ocasionar un severo daño a la velocidad de procesamiento de datos del mismo, para ello es recomendable que las actividades que se realizan para monitorear al servidor, mediante el uso de las herramientas de seguridad estudiadas en este trabajo, se reflejen en un calendario donde los tiempos de ejecución no afecten las operaciones de la organización, esto se puede lograr distinguiendo los tiempos críticos del sistema; por mencionar un ejemplo, existen empresas que tienen cierre contable al finalizar cada mes, en esta situación el sistema está saturado de procesos, por lo que no sería conveniente hacer una auditoría al sistema con Tripwire en ese momento crítico, como tampoco sería viable hacer un respaldo completo del sistema en esos instantes.

Cabe hacer notar que se puede lograr optimizar los procesos de la codificación de información en un servidor, considerando el uso de hardware especializado para hacer dichos procesos, donde el administrador del sistema tendrá que evaluar con base a las necesidades del sistema así como del presupuesto, la adquisición de hardware, por ejemplo una tarjeta de red, que tiene la capacidad de codificar la información evitando que el servidor tenga que hacer esos procesos, optimizando con esto los procesos del servidor.

Finalmente se debe elaborar reglas y políticas (tratadas en este trabajo), aceptadas, aprobadas y firmadas por la dirección de la organización, alternando con cursos de capacitación donde se informe a los usuarios los beneficios que obtendrán si emplean el reglamento, (cuotas de disco para cada usuario, hacer passwords seguros, no prestar cuentas, etc.) como también enseñar los beneficios que proporcionan el uso de herramientas de seguridad como pgp, secure shell, etc.

## FUENTES DE CONSULTA.

- **BIBLIOGRAFÍA:**


Rosen, Kenneth H. UNIX sistema V version 4, México, Editorial Mc Graw – Hill, 1991, 1046 páginas.
Tare, Ramkrishna. Procesamiento de datos en UNIX, México, Editorial Mc Graw – Hill, 1990, 357 páginas.
Schumer, Larry. Using UNIX Special Edition, Edit. Que, Indianapolis, Indiana, 1995, 948 páginas.
Arnold, N. Derek. UNIX security, a practical tutorial, Edit. Mc Graw – Hill, New York, 1993, 386 páginas.
Tanenbaum, Andrew, Redes de ordenadores, México, Editorial Prentice Hall Hispanoamericana, 1991, 759 páginas.
David Marín Carreño. Guía del Servidor Linux Conectiva S.A., Brasil, Segunda impresión, 26 de abril de 2000, 436 páginas.
Brian W. Kernighan. Traducido por Alberto Villareal Cueva. El Entorno de Programación UNIX, México, Ed. Prentice Hall, 1987.
J.J. Del Grande M.A. Introducción al Calculo Elemental, México, 1986, 448 páginas.
David Jones, Bruce Jamieson. An Introduction to Linux Systems Administration, Tercera Edición, 412 páginas. 1999.
Wirth Niklaus. Algoritmos y Estructuras de Datos, Ed. Prentice Hall, 1987, 306 páginas.
IBM de México, S.A. de C.V., Administracion del Sistema AIX, Impreso en México, 1998, 618 páginas.
David G Korn, Introduction to ksh-99, Bell laboratories, File case 49059-6, Charge case 311531-0101, Enero 14 de 1999, 37 páginas.


- **TESIS:**


L.I. Edith Tapia Rangel, Seguridad en Redes, Universidad Nacional Autónoma de México, Facultad de Contaduría y Administración, México, 1995, 250 páginas.
ING. Salvador Mandujano Vergara, Herramientas y métodos para la seguridad en computo. Universidad Nacional Autónoma de México, México D.F., 1998, 135 páginas.
ING. Carlos Enrique Lechuga Tello, Linux como opción de UNIX para sistema de archivos y servidores de Internet, México, 1999, Universidad Nacional Autónoma de México, 157 páginas.


- **URL´s:**

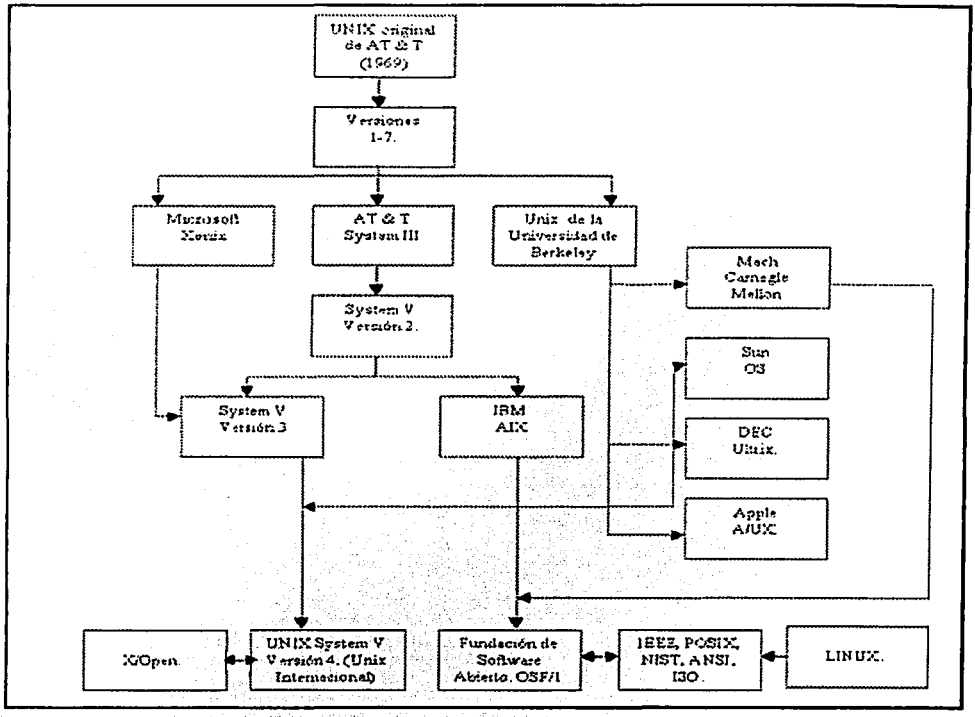

http://www.asc.unam.mx
http://www.ugu.com
http://www.cert.org/security-improvement/implementations/i042.07.html
Manual de Instalación para SSH. (UNIX, Windows, Mac):
http://www.seguridad.unam.mx/Tutoriales/Tutoriales/ssh/111
Secure Shell cliente/servidor para sistemas UNIX.
Sitio FTP de Secure Shell:
http://ftp.ssh.com/pub/ssh
Departamento de Seguridad en Cómputo:
ftp://ftp.seguridad.unam.mx/Herramientas/UNIX/Comunicacion
Secure Shell cliente para sistemas Windows.
Sitio FTP de Secure Shell (Solo para clientes con protocolo SSH2):
http://ftp.ssh.com/pub/ssh
Departamento de Seguridad en Cómputo:
ftp://ftp.seguridad.unam.mx/Herramientas/Windows/Comunicacion
Secure Shell cliente para sistemas Mac.
Departamento de Seguridad en Cómputo:
ftp://ftp.seguridad.unam.mx/Herramientas/Mac/Comunicacion/Ssh1

Manual de Instalación para Sendmail:
http://www.asc.unam.mx/Tutoriales/Tutoriales/sendmail/index.html
Download para obtener Sendmail:
ftp://ftp.sendmail.org/pub/sendmail/sendmail.8.11.0.tar.gz
ftp://ftp.asc.unam.mx/Herramientas/UNIX/Correo/Sendmail/sendmail.8.11.0.tar.gz
Manual de Instalación para Tripwire:
http://www.asc.unam.mx/Tutoriales/Tutoriales/tripwire/index.html
Download para obtener Tripwire Para UNIX. (Versión 1.2):
ftp://ftp.asc.unam.mx/Herramientas/UNIX/Monitor_Sistema/Tripwire/Tripwire-1.3.1-1.tar.gz
Download para obtener Tripwire Para Linux. (Versión 1.3):
ftp://ftp.asc.unam.mx/Herramientas/UNIX/Monitor_Sistema/Tripwire/tripwire.i386.tar.gz
Nota: En cualquiera de estos caso se obtiene acceso por anonymous.
Manual de Instalación para Portsentry:
http://www.asc.unam.mx/Tutoriales/Tutoriales/tripwire/index.html
Download para obtener Portsentry de Psionic Software Inc.:
http://www.psionic.com/abacus/portsentry
http://www.psionic.com/download
Download para obtener Portsentry en Departamento de Seguridad en Cómputo:
ftp://ftp.seguridad.unam.mx/Herramientas/UNIX/Monitor_Red
RPM de Portsentry de Linux RedHat:
http://rpmfind.net/linux/RPM/portsentry.html
Manual de Instalación para Cops:
http://www.asc.unam.mx/Tutoriales/Tutoriales/cops/index.html
Download para obtener Cops:
ftp://ftp.asc.unam.mx/pub/tools/cops.tar.gz
ftp://ftp.cert.org/pub/tools/cops/cops.1.04.tar.gz
ftp://coast.cs.purdue.edu/pub/tools/UNIX/cops/cops.1.04.tar.gz
Manual de Instalación para TCP Wrappers:
http://www.asc.unam.mx/Tutoriales/Tutoriales/tcpwrappers/index.html
Download para obtener Cops:
ftp://ftp.asc.unam.mx/pub/tools/tcp-wrappers.tar.gz
ftp://ftp.win.tue.nl/pub/security/tcp_wrappers_7.6.tar.gz
ftp://ftp.cert.org.pub/tools/tcp_wrappers/tcp_wrappers_7.6.tar.gz
ftp://coast.cs.purdue.edu/pub/tools/UNIX/tcp_wrappers/tcp_wrappers_7.6.tar.gz
Manual de Instalación para TCP Firewall:
http://linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html#toc4
Download para obtener Firewall:
http://rpmfind.net/linux/RPM/
Solaris/SunOS:
http://www.sunsolve.sun.com
IRIX:
http://www.sgi.com
SCO:
http://www.sco.com
AIX:
ftp://aix.software.ibm.com
HP-UX:
ftp://us-ffs.external.hp.com
NetBSD:
http://www.NetBSD.ORG
Documentación para Open SSL:
http://www.openssl.org/docs/
Download para obtener Open SSL:
http://www.openssl.org/source/
Documentación para Mod SSL:
http://www.modssl.org/docs/2.8/

Download para obtener Mod SSL:
http://www.modssl.org/source/
Manual de SATAN (SecurityAdministrator Tool for Analyzing Networks):
http://www.cert.org/advisories/CA-1995-06.html
Download para obtener SATAN. (SecurityAdministrator Tool for Analyzing Networks):
ftp://ftp.win.tue.nl/pub/security/satan-1.1.1.tar.Z
ftp://ftp.win.tue.nl/pub/security/satan-1.1.1.README
ftp://ftp.win.tue.nl/pub/security/satan_doc.tar.Z
ftp://ftp.win.tue.nl/pub/security/satan_doc.README
Download para obtener Tiger:
ftp://ftp.cerias.purdue.edu/pub/tools/UNIX/scanners/tiger/

## APÉNDICE I.

## LA EVOLUCIÓN DE UNIX.



# APÉNDICE 2.

## ALGORITMO DES.

DES es un codificador en bloque, lo que quiere decir que trabaja en bloques de texto de determinado tamaño (en este caso 64 bits) regresando bloques del mismo tamaño. Cada bloque de 64 bits es dividido en dos bloques de 32 bits cada uno denominados "L" y "R" (izquierdo y derecho, por las siglas en ingles.) Esta división se utiliza en ciertas operaciones.

Sea M el mensaje principal M = 0123456789ABCDEF donde M se encuentra en formato hexadecimal. Al cambiar M a formato binario obtenemos el bloque de 64 bits:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
(8 + 8 + 8 + 8 + 8 + 8 + 8 + 8 = 64 bits.)

L = 0000 0001 0010 0011 0100 0101 0110 0111
(8 + 8 + 8 + 8 = 32 bits.)

R = 1000 1001 1010 1011 1100 1101 1110 1111
(8 + 8 + 8 + 8 = 32 bits.)

L + R = 64 bits.

DES trabaja sobre los bloques de 64 bits utilizando llaves de 56 bits. Las llaves son guardadas como si utilizaran 64 bits pero cada octavo bit es ignorado. Es decir, los bits 8, 16, 24, 32, 40, 48, 56 y 64 son ignorados. Sin embargo, seguiremos denominando a los bits del 1 al 64. Cuando se creen las subllaves, los bits sobrantes serán eliminados.

Como ejemplo, K es la llave hexadecimal K = 133457799BBCDFF1. Transformándola a notación binaria (1 = 0001, 3 = 0011, etc.) y agrupando cada 8 bits, veremos que el último bit no será utilizado:

K = 0001 0011 0011 0100 0101 0111 0111 1001 1001 1011 1011 1100 1101 1111 1111 0001
(8 + 8 + 8 + 8 + 8 + 8 + 8 + 8 = 64 bits.)

Ahora si seguimos los pasos del algoritmo:

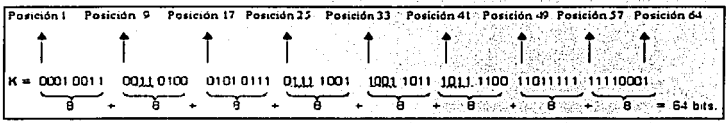
Paso 1: crear 16 subllaves cada una de las cuáles tendrán 48 bits.

La llave de 64 bits es permutada de acuerdo a la siguiente tabla, **PC-1**. Puesto que la primera entrada en la tabla es "57", esto quiere decir que el bit de la posición 57 de la llave original se convierte en el primer bit de la llave permutada **K+**. El bit 49 se convierte en el segundo bit, etc. Note que solo se utilizan 56 bits y faltan precisamente las posiciones múltiplos de 8.

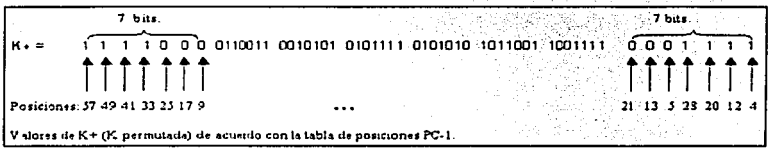
Tabla PC-1

| | | | | | | |
|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

De la llave original de 64 bits:




Obtenemos la permutación:



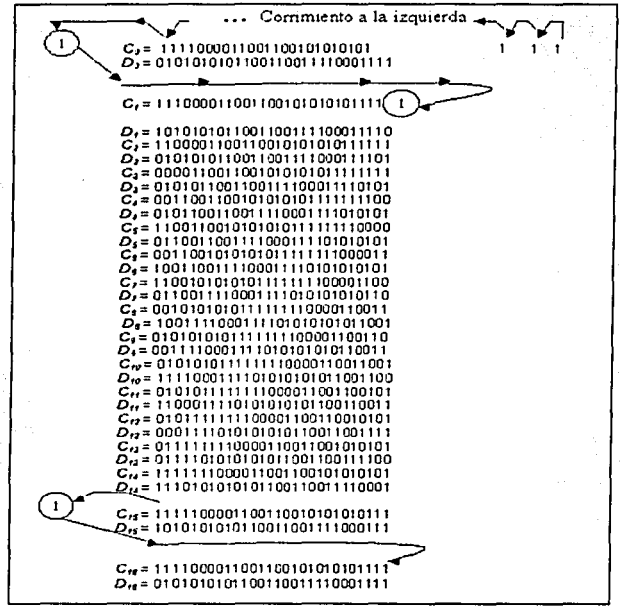

Si Ahora se separa la llave en dos mitades, $C_0$ y $D_0$, donde cada mitad tiene 28 bits.

$C_0$ = 1111000 0110011 0010101 0101111

$D_0$ = 0101010 1011001 1001111 0001111

Con $C_0$ y $D_0$ definidos, se puede crear 16 bloques ($C_n$ y $D_n$, donde n tiene valores desde 1 hasta 16). Después para hacer un corrimiento a la izquierda, mover cada bit un lugar a la izquierda, como en el siguiente cuadro:

Recorrer un bit a la izquierda.




Ahora se forman las llaves $K_n$, (donde n tiene valores desde 1hasta16), aplicando la siguiente tabla de permutaciones a cada uno de los pares (con su respectivo corrimiento a la izquierda) concatenados $C_nD_n$. Cada par tiene 56 bits, pero PC-2 solo utiliza 48 de estos.

TABLA PC-2.

| | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

Por lo tanto, el primer bit de $K_n$ es el 14avo bit de $C_nD_n$, el segundo bit el 17avo, y así en delante, terminando con el 48avo bit de $K_n$ siendo el 32avo bit de $C_nD_n$.

Para la primer llave (cuando n = 1 en $C_nD_n$ ) tenemos:

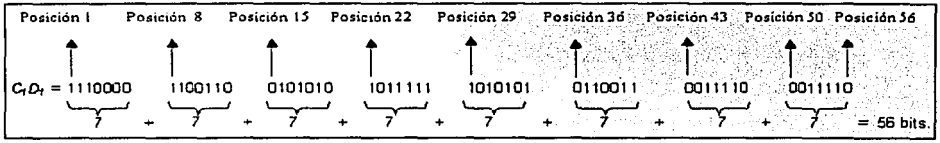
$C_1D_1$ = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

7 + 7 + 7 + 7 + 7 + 7 + 7 + 7 = 56 bits.

La cual, tras aplicar la permutación PC-2, se convierte en:

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010

Para obtener estos valores se muestra la siguiente figura:




Para las demás llaves tenemos:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| K2 | = | 011110 | 011010 | 111011 | 011001 | 110110 | 111100 | 100111 | 100101 |
| K3 | = | 010101 | 011111 | 110010 | 001010 | 010000 | 101100 | 111110 | 011001 |
| K4 | = | 011100 | 101010 | 110111 | 010110 | 110110 | 110011 | 010100 | 011101 |
| K5 | = | 011111 | 001110 | 110000 | 000000 | 111111 | 110101 | 001110 | 101000 |
| K6 | = | 011000 | 111010 | 010100 | 100111 | 110010 | 000111 | 101100 | 101111 |
| K7 | = | 111011 | 001000 | 010010 | 010110 | 111111 | 100001 | 100010 | 111100 |
| K8 | = | 111101 | 111000 | 101000 | 000111 | 010110 | 010011 | 101111 | 111011 |
| K9 | = | 111000 | 001101 | 101111 | 111101 | 011111 | 011110 | 011110 | 000001 |
| K10 | = | 101100 | 011111 | 001101 | 101000 | 111101 | 100100 | 011001 | 001111 |
| K11 | = | 001000 | 010101 | 111111 | 010011 | 110111 | 101101 | 001110 | 000110 |
| K12 | = | 011101 | 010111 | 000111 | 110101 | 100101 | 000110 | 011111 | 101001 |
| K13 | = | 100101 | 111100 | 010111 | 010001 | 111110 | 101011 | 101001 | 000001 |
| K14 | = | 010111 | 110100 | 001110 | 110111 | 111100 | 101110 | 011100 | 111010 |
| K15 | = | 101111 | 111001 | 000110 | 001101 | 001111 | 010011 | 111100 | 001010 |
| K16 | = | 110010 | 110011 | 110110 | 001011 | 000011 | 100001 | 011111 | 110101 |

Paso 2: Codificación de cada bloque de datos de 64 bits (el mensaje).

Existe una permutación inicial **PI** de los 64 bits del mensaje **M**. Esto reacomoda los bits de acuerdo a la siguiente tabla, donde las entradas en la tabla muestran el nuevo acomodo de los bits de acuerdo a su orden inicial. El 58avo bit de **M** pasa a ser el primer bit de **PI**. El 50avo bit de **M** pasa a ser el segundo bit de **PI**. El séptimo bit de **M** es el último bit de **PI**.

PI.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

Aplicando la permutación (con el mismo procedimeinto que PC-1, y PC-2) inicial al bloque de texto M, dado anteriormente, se obtiene:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
PI= 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Aqui el 58avo bit de M es "1", el cual se convierte en el primer bit de PI. El 50avo bit de M es "1" y se convierte en el segundo de PI. El séptimo bit de M es "0", el cual se convierte en el último bit de PI

Ahora se divide el bloque permutado **PI** en una mitad izquierda $L_0$ de 32 bits y una mitad derecha $R_0$ de 32 bits.

$L_0$ = 1100 1100 0000 0000 1100 1100 1111 1111
$R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010

Ahora se procede a través de las 16 iteraciones, para $1<=n<=16$, usando una función ***f*** que opera en dos bloques (un bloque de 32 bits y una llave $K_n$ de 48 bits) para producir un bloque de 32 bits. Para n desde 1 hasta 16 calculamos:

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

Donde $\oplus$ es la operación XOR

Esto resulta en un bloque final, para $n = 16$, de $L_{16}R_{16}$. Esto es, en cada iteración, se toman los 32 bits derechos del resultado previo y se convierten en 32 bits izquierdos del paso actual. Para los 32 bits derechos en el paso actual, aplicamos la operación XOR a los 32 bits izquierdos del paso anterior con la función ***f***.

Para $n = 1$, tenemos

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010
$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$R_1 = L_0 + f(R_0, K_1)$

Falta explicar el funcionamiento de la función ***f***. Para calcular ***f***, primero se expande cada bloque $R_{n-1}$ de 32 bits a 48 bits. Esto se hace utilizando una tabla de selección que repite algunos de los bits en $R_{n-1}$. Le llamaremos al uso de esta tabla de selección como función E. Así $E(R_{n-1})$ tiene un bloque de entrada de 32 bits y un bloque de salida de 48 bits.

Sea E tal que los 48 bits de su salida, escrita como 8 bloques de 6 bits cada uno, sean obtenidos al seleccionar los bits en sus entradas en orden de acuerdo a las siguientes tablas:

**TABLA DE SELECCCIÓN DE BITS E:**

| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

Así los primeros tres bits de $E(R_{n-1})$ son los bits de las posiciones 32, 1 y 2 de $R_{n-1}$ mientras que los dos últimos 2 bits de $E(R_{n-1})$ son los bits de las posiciones 32 y 1.

Calculo $E(R_0)$ de $R_0$ como sigue:

$R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$E(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

(Note que cada bloque de 4 bits originales ha sido expandido a un bloque de salida de 6 bits.)
Ahora se calcula la función *f*, haciendo un XOR con la salida $E(R_{n-1})$ junto con la llave $K_n$:

$K_n + E(R_{n-1})$.
Para $K_1$, $E(R_0)$, tenemos:

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010
$E(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101
$K_1+E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

Aun no se terminan los cálculos de *f*. Hasta este punto se ha expandido $R_{n-1}$ desde 32 bits a 48 bits, utilizando la tabla de selección y se ha aplicado un XOR al resultado con la llave $K_n$. Ahora hay 48 bits, o 8 grupos de 6 bits. Ahora se procesará a cada grupo de 6 bits una S-caja: se utilizarán como direcciones en tablas llamadas **"cajas S"**. Cada grupo de 6 bits nos dará una dirección en una caja S distinta. Localizado en ese lugar se encontrará un número de 4 bits. Este número de 4 bits reemplazará a los 6 originales. El resultado neto es que los 8 grupos de 6 bits serán transformados en 8 grupos de 4 bits (las salidas de 4 bits de las cajas S) para 32 bits en total.

Con el resultado previo, que es de 48 bits, en la forma:

$K_n + E(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$,

Donde cada $B_i$ es un grupo de 6 bits. Ahora se calcula

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$

Donde $S_i(B_i)$ se refiere a la salida de la caja S número *i*. Para repetir, cada una de las funciones *S1, S2,..., S8*, toma un bloque de 6 bits y lo convierte a un bloque de 4 bits. La tabla que determina $S_1$ es mostrada y explicada a continuación:

| | Número de columna | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reng No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |
| | S1 | | | | | | | | | | | | | | | |

Si $S_1$ es la función definida en esta tabla y *B* es un bloque de 6 bits, entonces $S_1(B)$ es determinada de la siguiente manera: El primer y último bit de *B* representan en base 2 el número en el rango decimal 0 a 3 (o binario 00 a 11). Sea el número *i*. Los 4 bits medios de *B* representan en base 2 un número en el rango decimal 0 a 15 (binario 0000 a 1111). Sea ese número *j*. Consultar en la tabla el número en el renglón *i* y en la columna *j*. El resultado de la consulta es un número en el rango de 0 a 15 y esta representado por un bloque de 4 bits. Ese bloque es la salida $S_1(B)$ de $S_1$

para la entrada *B*. Por ejemplo, para la entrada $B$ = 011011 el primer bit es "0" y el último bit "1" dando 01 como el renglón. Este es el renglón 1. Los 4 bits medios son "1101". Este es el equivalente binario del decimal 13, así que la columna es la 13. En el renglón 1, columna 13 aparece 5. Esto determina la salida; 5 es el número binario 0101, así que la salida es 0101. Así $S_1(011011) = 0101$.

Las tablas definiendo las funciones $S_1,\ldots,S_8$ son las siguientes:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

S1

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |

S2

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |

S3

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |

S4

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |

S5

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |

S6

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |

S7

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

S8

Para la primera ronda, obtenemos como resultado de las ocho cajas S:

$K_1 + E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

La última fase del cálculo de *f* es realizar una permutación P de la salida de la S–caja para obtener el valor final de *f*.

$f = P(S_1(B_1)S_2(B_2)\ldots S_8(B_8))$

La permutación P está definida en la siguiente tabla. P produce una salida de 32-bit de una entrada de 32-bit permutando los bits de la entrada.

P

| | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

Para la entrada de las ocho cajas S:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

Se Obtiene:

$f$ = 0010 0011 0100 1010 1010 1001 1011 1011

$R_1 = L_0 + f(R_0, K_1)$

= 1100 1100 0000 0000 1100 1100 1111 1111

+ 0010 0011 0100 1010 1010 1001 1011 1011

= 1110 1111 0100 1010 0110 0101 0100 0100

En la siguiente ronda, teniendo $L_2 = R_1$, que es el bloque que se acaba de calcular y luego se tendrá que calcular $R_2 = L_1 \, A \, f(R_1, K_2)$, y así por 16 rondas. Al final de la última ronda, se obtendrá los bloques $L_{16}$ y $R_{16}$. Luego se invertirá el orden de los dos bloques en el bloque de 64 bits.

$R_{16}L_{16}$

Y aplicar una permutación final PI como es definida por la siguiente tabla:

**PI**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

Esto es, la salida del algoritmo tiene el bit 40 de la entrada como su primer, etc.

Si se procesa los 16 bloques utilizando este método, al llegar a la última ronda (la número 16)

$L_{16}$ = 0100 0011 0100 0010 0011 0010 0011 0100
$R_{16}$ = 0000 1010 0100 1100 1101 1001 1001 0101

Invertimos el orden de estos dos bloques y aplicamos la permutación final a

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100
$PI^{-1}$ = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

que en forma hexadecimal es

85E813540F0AB405.

Esta es la forma codificada de M = 0123456789ABCDEF:

La descodificación es el orden inverso de la codificación. La descodificación es el inverso de la codificación, siguiendo los mismos pasos pero invirtiendo el orden en que las subllaves son aplicadas.

# APÉNDICE 3.

## EL ALGORITMO RSA.

El algoritmo reside en la función C=M^e (mod N) y se va a explicar, a partir de dos personajes ficticios, su funcionamiento para codificar y descodificar mensajes. Aunque, en realidad, el algoritmo emplea números de gran tamaño, utilizaremos valores pequeños que simplifican los cálculos y hace más fácil de comprender.

Primero, Alicia elige dos números primos cualesquiera p y q, por ejemplo, p=17 y q=11, que deben permanecer en secreto y no divulgarse.

Después, calcula N=p*q, es decir, multiplica p por q para obtener un número aún mayor. En este número radica el secreto del algoritmo, en concreto en la dificultad que presenta factorizar un número de gran tamaño, que, computacionalmente, es un problema muy complejo.

Para este ejemplo N=17 * 11 = 187.

Alicia debe escoger otro número e, de tal forma que e y (p-1)*(q-1) sean primos relativos, es decir su máximo común divisor es 1 (no tienen factores comunes salvo el 1). Por ejemplo e = 7, que es primo relativo con 16*10 = 160.

Será el par de números e y N los que constituirán la clave pública de Alicia, si bien e puede pertenecer a la clave pública de cualquier otra persona, N deberá ser diferente para cada persona ya que dependerá de la elección inicial de los factores p y q.

Los valores de e y N son fundamentales para la codificación del mensaje, por lo que deberán estar a disposición de cualquiera que desee enviar un mensaje codificado a Alicia.

Cuando Alicia distribuye su clave pública formada por el par (e, N)=(7, 187), Benito ya puede codificar el mensaje. Para que pueda hacerlo, y que forme parte del algoritmo, Benito debe codificar, en forma de número, el valor M.

Un buen método para hacerlo puede ser a través de los valores binarios de cada carácter (en ASCII) convirtiendo, posteriormente, el valor binario final a decimal. Así, por ejemplo, las letras XYZ son los valores ASCII 88 (1011000), 89 (1011001) y 90 (1011010). De ahí obtenemos el número binario 101100010110011011010, que en decimal equivale a 1453274.

Para simplificar tomaremos el valor de Z, por lo que hacemos M=90. De esta forma ya tenemos los elementos básicos para codificar el mensaje: N=187, e=7 y M=90. Ahora entraría a trabajar la fórmula que se expuso anteriormente C = M^e (mod N) = 90 ^ 7 (mod 187) = 95.

De esta forma, se ha obtenido como texto codificado C el valor 95 (C=95). Las funciones modulares empleadas son funciones de una sola vía, por lo que aunque conozcamos el resultado será muy difícil recuperar el valor original de M. Esto implica que aunque alguien intercepte el mensaje codificado, no podrá descodificarlo aunque conozca la clave pública de Alicia. Sin embargo, ella si podrá, ya que conoce los valores de p (p =17) y q (q =11) que originaron N.

Su clave privada se obtiene de la siguiente forma:

e*d = 1 (mod (p-1)*(q-1)) 7*d = 1 (mod 160)

Mediante el uso del Algoritmo de Euclides se calcula que d=23.

Una vez que Alicia tiene el mensaje en su poder, y dispone de su clave privada para descodificar, podrá interpretar el contenido del mensaje empleando la fórmula original pero cambiando los valores de posición para quedar como M = C^d (mod N). Para el ejemplo (sabiendo que 23 = 1+2+4+16) sería:

M = 95 ^ 23 (mod 187) M = [95 ^ 1 (mod 187) * 95 ^ 2 (mod 187) * 95 ^ 4 (mod 187) * 95 ^ 16 (mod 187)] (mod 187) = [95 * 49 * 157 * 103 ] (mod 187) = 75276005 (mod 187) = 90 = Z en ASCII, el mensaje enviado por Alicia.

## APENDICE 4.

## CUESTIONARIOS.

Las personas entrevistadas son:

Angel Alberto Vázquez Chavez, Gerente de Sistemas.
Jorge Vázquez Molina, Jefe de Soporte técnico.
Rafael García Pineda, Encargado de Internet y Publicaciones.
Verónica Molina Reyes, Jefa de Mercadotecnia.

| Gerente de Sistemas. | Respuestas. |
|---|---|
| ¿En que sistema operativo tienen los servidores?. | En Windows NT y en UNIX. |
| ¿Tienen herramientas de seguridad actualmente?. | No. |
| ¿Han sido víctimas de un ataque informático?, ¿De que tipo?. | Si. La página de Internet, de la compañía, ha sido modificada varias veces, la velocidad de la red se vuelve lenta, problemas con el correo electrónico. |
| ¿Qué medidas, políticas emplean para el uso de los equipos?. | Que los empleados de la compañía no usen Internet, y tampoco envíen más correos de lo necesario. |
| ¿Conocen el sistema operativo UNIX?. | Si. |
| ¿En cuanto tiempo se resuelven los problemas relacionados con la seguridad del sistema?. | Es variable, los que quitan más tiempo es cuando borran la información, porque hay que traer dicha información de los respaldos. |
| ¿Respaldan la información?, ¿Con qué frecuencia?. | 1. Sí. 2. La base de datos todos los días y la página de Nintendo solo 1 vez cada que hay cambios. |
| ¿Saben de dónde provienen los ataques?. | No. |
| ¿Han emitido alguna propuesta para solucionar está problemática?. | No. |

| Jefe de Soporte Técnico. | Respuestas. |
|---|---|
| ¿En que sistema operativo tienen los servidores?. | En Windows NT y en UNIX. |
| ¿Tienen herramientas de seguridad actualmente?. | No. |
| ¿Han sido víctimas de un ataque informático?, ¿De que tipo?. | Negación de servicios principalmente, por ejemplo han cambiado direcciones IP de nuestro router ocasionando que estemos fuera de la red. |
| ¿Qué medidas, políticas emplean para el uso de los equipos?. | Usar Internet sólo cuando sea necesario. |
| ¿Conocen el sistema operativo UNIX?. | Si. |
| ¿En cuanto tiempo se resuelven los problemas relacionados con la seguridad del sistema?. | Por lo general dos horas, dependiendo de la naturaleza del problema. |
| ¿Respaldan la información?, ¿Con qué frecuencia?. | Si. Diario. |
| ¿Saben de dónde provienen los ataques?. | No. |
| ¿Han emitido alguna propuesta para solucionar está problemática?. | No. |

| Encargado de Internet y Publicaciones. | Respuestas. |
|---|---|
| ¿En que sistema operativo tienen los servidores?. | En Windows NT y en UNIX. |
| ¿Tienen herramientas de seguridad actualmente?. | No. |
| ¿Han sido víctimas de un ataque informático?, ¿De que tipo?. | Problemas con el correo, tarda mucho para llegar y aveces recibimos quejas de clientes, proveedores y amistades de que nos envían correos que no recibimos, a veces se cae el enlace con nuestro proveedor. |
| ¿Qué medidas, políticas emplean para el uso de los equipos?. | No usar ICQ, chats, ni tampoco enviar correos con mucha información. |
| ¿Conocen el sistema operativo UNIX?. | Sí. |
| ¿En cuanto tiempo se resuelven los problemas relacionados con la seguridad del sistema?. | Hay ocasiones en que no se resuelve el problema del acceso a Internet hasta de 1 día.. |
| ¿Respaldan la información?, ¿Con qué frecuencia?. | Sí. Diario. |
| ¿Saben de dónde provienen los ataques?. | De Internet. |
| ¿Han emitido alguna propuesta para solucionar está problemática?. | No. |

| Jefe de Mercadotecnia. | Respuestas. |
|---|---|
| ¿En que sistema operativo tienen los servidores?. | No sé. |
| ¿Tienen herramientas de seguridad actualmente?. | No sé. |
| ¿Han sido víctimas de un ataque informático?, ¿De que tipo?. | A veces no tenemos los servicios de internet como el correo, etc. |
| ¿Qué medidas, políticas emplean para el uso de los equipos?. | No usar ICQ. |
| ¿Conocen el sistema operativo UNIX?. | No. |
| ¿En cuanto tiempo se resuelven los problemas relacionados con la seguridad del sistema?. | Se resuelve rápido, pero es muy frecuente. |
| ¿Respaldan la información?, ¿Con qué frecuencia?. | No sé. |
| ¿Saben de dónde provienen los ataques?. | No sé. |
| ¿Han emitido alguna propuesta para solucionar está problemática?. | No sé. |