# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## FACULTAD DE INGENIERÍA

# HERRAMIENTAS DE SEGURIDAD PARA EL SISTEMA OPERATIVO UNIX E IMPLEMENTACIÓN DE UN SISTEMA AUDITOR

**T E S I S**

QUE PARA OBTENER EL TÍTULO DE:

**INGENIERA EN COMPUTACIÓN**

P R E S E N T A:

**ANA MARÍA CÓRDOBA MÉNDEZ**

DIRECTOR

**ING. ARMANDO VEGA ALVARADO**

COODIRECTORA

**ING. LAURA SANDOVAL MONTAÑO**


México, D.F. Marzo de 2006

# *Agradecimientos*

El camino que he recorrido está muy lejos de haber sido sencillo, sin embargo muchas personas han intervenido para que logrará llegar a hasta aquí. A ellas deseo dar un sincero agradecimiento y por ello he escrito estas líneas.

*¡Gracias!*

***A ti Dios mío y a tu Santa Madre.***

Por regalarme este momento, por rodearme de tanta gente, por las pruebas tan duras, pero ante todo; por estar cerca de mí.

***A mi Mamá y a mi tía Mary***

A mi mamá, porque no me alcanzan las palabras para agradecerle todo su esfuerzo, jamás tendré mejor ejemplo de tenacidad, sobreponiéndose a todo y a todos; en donde el poco tiempo que compartimos por sus largas jornadas de trabajo, hoy me dejan ver su compromiso ante la vida y el ejemplo para esforzarme en defender con todo mi ser lo que amo y deseo.

A mi tía Mary, mi segunda madre, varias líneas no alcanzarían para redactar su entrega y paciencia a lo largo de toda mi vida, en detalles tan grandes o pequeños que sólo alguien que me ama lo puede hacer, ¡Gracias por hacerlo conmigo!

***A mis Abuelos*** ✞

Por los valores que me inculcaron; gracias por los consejos de mi abuelo José Rosario✞, y la disciplina de mi abuela Catalina✞.

***Al Pbro. Mario Contreras***

Gracias por mostrarme que el mundo puede ser maravilloso, gracias por sus enseñanzas, por su confianza y por su cariño, por la oportunidad de trabajar junto a usted en el decanato, y por confiar en mí.

***A María del Carmen Arancón García***

Mi hermana en Cristo de la Escuela de Pastoral, por abrirme las puertas de su casa y de su corazón, por escucharme, apoyarme en mis actividades dentro de la región que juntas atendimos, por impulsarme a no abandonar ni lo más pequeño, por creer en mi, pero ante todo por ayudarme a sanar mis "heridas de batalla" con sus sabios consejos y palabras.

***A mis Hermanos en Cristo***

Por sus palabras de aliento, porque son católicos convencidos de su fe, pero ante todo hombres y mujeres exitosos en su vida profesional, a mis hermanos de la II Vicaría: Grace López, Carmen Alvarado, Elfego Morales, Miguel Robles y a mis amadas "Comisión de Cristo Rey y Comisión de Basilica" Enrique Durán, Venturina Gutiérrez, Elvira López, Miguel Ángel Velásquez, Víctor López e Isela García.

***A mis compañeros de DGSCA.***

A la Química Laura Mata, por la oportunidad de pertenecer a su equipo, por la formación que me brinda cada día, por el voto de confianza, por el apoyo, pero ante todo por su sinceridad para señalarme mis aciertos y mis fallas.

A la Matemática Carmen Bravo, por darme la oportunidad de ser académico y crecer como universitaria.

Al Químico Arturo Hernández, que en mi servicio social, me oriento con paciencia y me brindo la confianza de ser becaria.

A Maru Solis y a Esther, por creer en mí, por brindarme su amistad y cariño, por compartir mis secretos, por no juzgar mis fallas y por respetar mí silencio.

Así como a todos mis compañeros de los Departamentos de Infraestructura y Control Escolar.

***A mi Director y Codirectora de Tesis***

Al Ing. Armando Vega Alvarado director de esta tesis, por su paciencia, su comprensión y su apoyo, sin el cual este proyecto jamás hubiera sido concluido.

A la Ing. Laura Sandoval, por sus observaciones y tiempo, muestra inequívoca de excepcional formación académica.

***A mis Sinodales.***

Gracias, por ser parte de este momento, espacialmente como representantes de la Facultad de Ingeniería y principalmente de la UNAM, Máxima Casa de Estudios de México.

Ana María Córdoba Méndez
**¡POR MI RAZA HABLARÁ EL ESPÍRITU!**
***Marzo de 2006***

## ÍNDICE

# PRÓLOGO

La presente tesis, ofrece un panorama general de los aspectos más importantes en seguridad para los sistemas montados en ambiente UNIX, un enfoque que muestra las diversas herramientas de seguridad en software libre, como una alternativa confiable para el resguardo de información, y además; la propuesta de una nueva herramienta, creada con el fin de enriquecer la ya basta gama de posibilidades existentes en la red, como una opción real y accesible ante el creciente número de ataques informáticos registrados en nuestro tiempo.

Sin embargo, se ha considerado sumamente útil iniciar con la documentación de algunos sucesos relevantes registrados de manera cronológica, que han permitido tomar conciencia, de las devastadoras consecuencias de los ataques por software nocivo; esta historia, conocida y dolorosamente vivida por muchos usuarios y creadores o administradores de grandes sistemas, nos permitirá tener los antecedentes para respaldar nuestras “paranoicas” decisiones ante la posibilidad de un intruso.

Pues bien, a lo largo del Primer Capítulo se presentarán como antecedentes, una serie de estas citas cronológicas, y además, una breve reseña de los más dañinos virus informáticos que ocasionaron el fin de grandes sistemas. Por otro lado, se mencionarán definiciones de importantes conceptos dentro del ambiente de seguridad en cómputo, así como las descripciones de las amenazas al sistema y los principios en que se fundamentan los ataques.

Posteriormente a lo largo del Segundo Capítulo, conoceremos algunas de las herramientas del grupo de software libre para sistemas UNIX, organizadas de acuerdo a sus características; ofreceremos una descripción general de sus funciones, y algunas ligas para conseguirlas, en algunos casos nos fue posible obtener el dato de su creador y también de su significado, datos por demás interesantes, si consideramos que al ser parte del software libre muchas de ellas han sido enriquecidas por una comunidad cibernética, que en el afán de construir ambientes de trabajo seguros y confiables, se ha dado a la tarea de compartir su conocimiento en el mutuo beneficio de enriquecer un bien común.

La intención al presentar primero un marco en el que logremos evaluar el posible daño a nuestro sistema, y las herramientas disponibles para contrarrestar el ataque, pondrán sobre la mesa los antecedentes en los que se basa nuestra propuesta: “crear una nueva herramienta que contrarreste el efecto del intruso, y que cuente con un recurso inteligente, la memoria.

Por ello, la propuesta del Auditor de Nocividad de Archivos (ANA), abarca el Tercer Capítulo de este trabajo y da a conocer a nuestro lector las características básicas de funcionamiento, de la nueva herramienta propuesta; documentando su desarrollo.

Más adelante, ofreceré una conclusión a la premisa, del sentido que tiene crear una nueva herramienta, ante la existencia de una amalgama de posibilidades en el mercado.

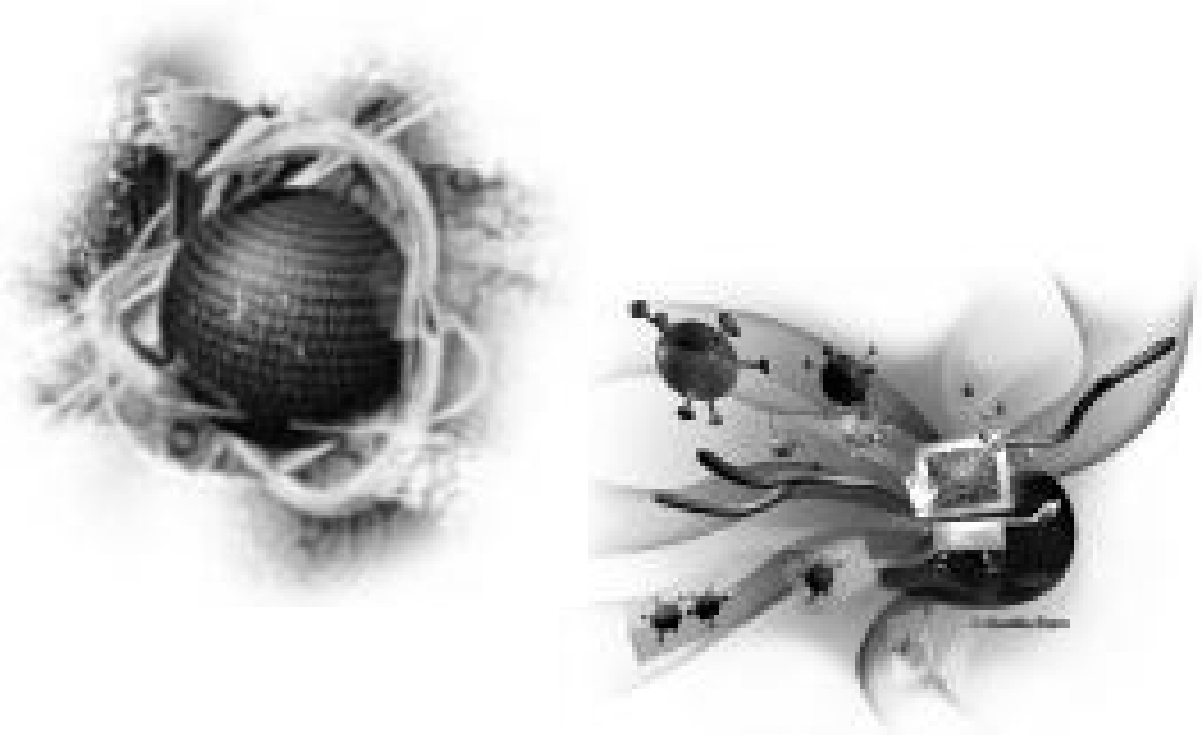
Posteriormente se ha agregado una serie de anexos en los cuales se integra información detallada de otros tópicos de seguridad: recomendaciones de seguridad en general y políticas de seguridad que se establecen para lograr el tener el control de los factores que amenacen a nuestro sistema.

Empeñados en la construcción de un sistema invulnerable, es importante que reconsideremos el punto del que partimos, el hoy, y por ello se ha incluido un detallado proceso de auditoría informática, con todos los instrumentos de evaluación para despertar a la realidad en el sistema que actualmente trabajamos.

Por lo tanto, presentaremos aquí a la seguridad como un tópico de múltiples aristas, que albergan un sin fin de posibilidades de efectividad, pero antes que nada debemos entender que la seguridad no es cuestión de suerte o de la aplicación de soluciones, sólo la conciencia de trabajar con sistemas vulnerables nos ayudará a planear y prever fallas; y a aprender de los ataques registrados para tomar las decisiones que nos lleven a convertirnos en buenos realmente valiosos administradores.

# CAPÍTULO I

# INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO



# CAPÍTULO I. INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

El concepto de Seguridad ha dejado de ser un elemento opcional en el desempeño diario de las operaciones requeridas por el área de sistemas, hoy nos vemos rebasados por la gran cantidad de ataques al resguardo de información y a la información misma, por lo que nadie nos puede garantizar el 100% en la efectividad de los *firewall* utilizados para evitar este problema, además, los intrusos superan constantemente a los administradores, antes de que se encuentren soluciones muchas veces de limitada rapidez, pues en gran cantidad de casos, esa rapidez está determinada por el tiempo en el rastreo de los daños y la identificación del intruso que los realizó, dejando como consecuencia la vulnerabilidad del sistema a nuevos ataques mientras no sea detectado el intruso que los ocasiona, y su efecto se detendrá hasta iniciar con la reparación de la falla, la identificación del intruso, su erradicación de nuestro sistema, y la toma de medidas necesarias para que no se repita este suceso, aunado a la inversión de tiempo en el intento por recuperar la información pérdida tal vez en su totalidad, o bien la reconstrucción de esa información hasta el instante previo al ataque, lo que podría ser logrado si contamos con un respaldo seguro, que nos garantice que la información recuperada de forma parcial sea confiable y pueda ser tomada como el punto de partida para reiniciar las operaciones del sistema en forma regular, todo ésto podría demandar que esa inversión de tiempo se comporte de manera exponencial, lo que en gran parte será determinado por la habilidad del administrador al aplicar la alternativa más conveniente a la falla que se detecte.

Analizando un poco más este problema, podemos mencionar que la tendencia en la creación de software nocivo ha llegado a rebasar, y por mucho, a las empresas encargadas de ofrecer soluciones; los hackers que continuamente están en la búsqueda de fallas o "huecos de seguridad" en los sistemas, tienen como reto personal el violar la integridad de la información con fines que van desde el ego, hasta la remuneración económica; por ejemplo, distintas empresas encargadas de la creación de "antivirus" y/o herramientas de seguridad, han declarado que existe una nueva forma de alianza hackers-creadores de virus[1], mencionando que estos hábiles programadores que al parecer empiezan a trabajar en conjunto, también han iniciado un peligroso intercambio de información librando diferencias con un sólo objetivo, crear una nueva generación de software nocivo; hablaríamos entonces de un caos en los sistemas de información, como ya se ha registrado desde el inicio de este milenio, y sin embargo el daño no terminaría ahí, pues se estima que la nueva generación de virus posee "inteligencia", y es capaz de reenviarse, encriptarse o viajar por correo electrónico a través de la red, para destruir o alterar toda la información a su paso, historia que se ya se ha vivido en los últimos ataques registrados; y se ha estimado que estos intrusos podrían ser creados a una velocidad promedio que va de los 200 a 300 virus al día, debido a que los hackers y sus creadores ahora trabajan unidos.

Por otro lado, el área de desarrollo de sistemas está enfocada en gran parte, a la eficiencia en el uso de recursos de cómputo, a su facilidad de acceso, al atractivo gráfico o a una interface amigable con el usuario, aspectos buscados incansablemente en la compra de

[1] Notimex México.-Hackers y Creadores de Virus Comienzan a Unirse, 28 de mayo de 2004

la última versión de un nuevo producto, para ser instalado de forma sencilla y veloz por cualquier usuario en un equipo propio o de la empresa en donde labore, sin la más mínima precaución, o bien otro lado de la moneda, las empresas invierten en capacitación para que sus trabajadores aprendan a manejar herramientas o paquetes muy específicos que desean que se apliquen de forma inmediata en su ambiente de trabajo, y con ello han dado origen a lo que hoy se conoce como "problemas de cultura" [2], bautizado así por las empresas desarrolladoras de soluciones, cuando califican la actitud de los usuarios que son incapaces de evaluar la vulnerabilidad del inadecuado uso de su equipo, debido a que sólo han recibido la formación esencial para el manejo del sistema en cuestión, aspecto que sin duda puede ser parcialmente resuelto con la creación de claves con privilegios limitados. Con todo ésto, tenemos una perspectiva general del incomodo espacio de acción en que se mueve de forma regular un administrador, ya que por mucho tiempo que dedique al monitoreo del sistema de la empresa no podrá eliminar el factor humano en la incidencia de fallas, y detener este tipo de ataques o pérdida de información por descuidos en los equipos que la albergan, jamás llegará a su fin.

Desgraciadamente, la generación de software nocivo inteligente que está siendo creada, puede interactuar con los usuarios que hemos mencionado, pues es interesante reflexionar sobre la eficiencia de los buscadores en la Red Internacional de Información, abiertos a todo público y que permiten de una forma muy sencilla encontrar el software que podría ser el inicio de la reacción en cadena para terminar con el sistema de una empresa de cualquier tamaño; pues basta con elegir el buscador e introducir la palabra "virus" o cualquier otra búsqueda, y aparecerá todo tipo de ligas con los archivos que contienen información al respecto, o mejor dicho la posibilidad de una infección automática; de la misma forma ocurre con las imágenes, los videos y otros muchos productos que se ofrecen en la red, pero es de lo más común encontrar a un empleado navegando incansablemente en el ciberespacio. En la mayoría de los casos no existen restricciones para los usuarios de estos equipos, muchos de ellos tienen esa cuenta de correo electrónico que hace las veces de medio de transporte de un intruso, lo que parecería absurdo, si pensamos que nadie entraría en estos sitios para encontrar dicha información, sin embargo la constante después de detectar una infección, es que nadie les ha mencionado que sus consultas les traen sorpresas, y los últimos años los reportes por pérdidas económicas han alcanzado cifras alarmantes que son consecuencia de una pérdida total de las bases de datos que son alteradas, todo ello por la acción de un intruso en el sistema, o el uso de información infectada.

La parte más preocupante es que la implementación de herramientas de seguridad, se ha ocupado como un método de solución a los conflictos presentados y no como un método de prevención, pues son regularmente las grandes empresas quienes invierten en el resguardo de su información y tienen presente su importancia, pues de ello depende su permanencia, y su capacidad de competir en el mercado, no así las empresas pequeñas que se conforman con actuar en algunos casos al margen de las políticas, o más aún hacen caso omiso de ellas. La información ofrece poder a quien la posee en casos de consultoría y algunos servicios como los bancarios, la información se considera una parte importante del capital de

[2] Notimex México.-Hackers y Creadores de Virus Comienzan a Unirse, 28 de mayo de 2004

la empresa; mediante este enfoque podríamos justificar la presencia de los costosos sistemas de seguridad que desgraciadamente muy pocas instituciones implementan. Incluso podemos extrapolar este ejemplo a las empresas que desarrollan software como los antivirus; ellas tienen un gran nicho de mercado en el que sólo el más rápido y eficiente sobrevive, pues es éste quien coloca la primera oportunidad en el mercado para reparar la falla, aún cuando poco después aparezca un nuevo producto mejorado, el usuario que ha adquirido dicho producto identificará a su salvador que llegó en primer lugar, por esta razón las empresas que se dedican al desarrollo de soluciones están constantemente a la búsqueda de los mejores programadores; paradójicamente, en esta competencia un gigante como Microsoft, ha sido uno de los más atacados, y sus productos son ya asiduos clientes de los hackers, por lo que adquirió su propia empresa de antivirus, sin embargo para tranquilidad de las otras empresas, Microsoft ha comentado que no incluirá vacunas en su sistema, sólo herramientas como respaldo; un sistema más estable como es el caso de UNIX cuenta con una gran ventaja, maneja herramientas de seguridad con características muy especiales de las que hablaremos más adelante, y así como los hackers hacen alianzas con los creadores de virus, aquellos administradores preocupados por la integridad de su información enriquecen esas herramientas para también unirse en este gran proceso de intercambio, un nuevo tipo de guerra informática, en el que de nueva cuenta permanece el que da la respuesta más rápida.

Pero como opción para apoyarnos en una base firme antes de iniciar cualquier cambio en los recursos de seguridad de un sistema a nuestro cargo, podemos aplicar una auditoría informática, y realizar una revisión de los procedimientos utilizados en los equipos de cómputo, desde la entrada de la información y su procesamiento hasta el archivo y los niveles de seguridad para su resguardo, en suma, una radiografía de la empresa; que nos permitirá conocer el nivel real de seguridad con que se trabaja; y lógicamente con una objetiva interpretación, se pueden detectar fácilmente las vulnerabilidades y las áreas susceptibles de falla, lo que representaría un primer paso para corregir el rumbo en los procedimientos aplicados al manejo de información en caso de que el resultado fuera negativo. Sin duda, no desaparecerán las infecciones por virus en los computadores personales, o los usuarios que se encargan de instalar en sus máquinas las más variadas “basuras informáticas” de promoción, pero el sistema “central” puede resguardar la parte más importante, el servidor será monitoreado constantemente y enviará avisos que nos podrían brindar una oportunidad para tomar decisiones oportunas ante un ataque, de forma previa a perder el control, o bien a tiempo de resguardar el sistema de un conflicto mayor.

Ante esta perspectiva un consultor o bien el administrador debe percibir la necesidad de ofrecer algo más, lo que podría traducirse en una herramienta que se ocupe de realizar el monitoreo, la comunicación y el protocolo de red, o bien un diseño a la medida de las necesidades específicas del sistema que será implementado, con lo que podríamos trabajar en un ambiente más estable y menos propenso a fallas; implicará largas horas de diseño y una adecuada administración, pero valdrá oro molido en el caso de un ataque, sin perder de vista que jamás podremos garantizar el 100% de efectividad. Sin embargo, brindar una oportunidad de recuperar información, puede hacer la diferencia entre la desaparición de una empresa, como hemos comentado anteriormente, aún cuando hablemos de una pérdida

parcial que se traduzca en cifras monetarias que pueden alcanzar cualquier cantidad de ceros, como ha ocurrido recientemente en varios casos a nivel mundial, en donde varias empresas han sido afectadas por la entrada al sistema de un intruso que no se ha detectado a tiempo, sumado al inconveniente de no contar con la oportunidad de detener la destrucción de los archivos a su paso. En estas circunstancias, sería recomendable contar con la herramienta que hemos descrito como monitor, la cual podría disminuir el tiempo de recuperación de nuestra información y brindar un respiro a los propietarios de la empresa, si es que se está a tiempo de evitar un daño mayor.

Aquí podemos mencionar el hecho de que varias empresas tienen montado su sistema en ambiente UNIX y como comentábamos antes puede hacer uso de una serie de ventajosos recursos, que se pueden obtener de forma gratuita; y son conocidos como software libre, creados originalmente en el intento por intercambiar información valiosa de programadores independientes que se unen para enriquecer sus creaciones; en cuestiones de seguridad, las herramientas proporcionan una ventaja adicional además de encontrarse a disposición de cualquier usuario en servidores que se conectan a la red, ésta es, su estructura, puede ser conocida a detalle, y ser modificada de acuerdo a las necesidades de nuestro propio sistema; se provee además una gran cantidad de información, en sitios especializados para ello, bibliografía y un sinnúmero de notas sobre Congresos en los que se ha incluido su estudio, como es el caso de la UNAM, en donde el conocimiento de estas herramientas es materia de uso común para los administradores y usuarios que trabajan en el manejo de los sistemas desarrollados bajo este ambiente, la misma Universidad posee varios sistemas de información que usan como plataforma este Sistema Operativo, por lo que muchos de los involucrados conocen bien el tema, como es el caso del área de Investigación de la DGSCA, en donde de forma continua y como respuesta inmediata a la presencia de un ataque se emiten boletines de información para reportar las características de las fallas registradas y su manejo, dando seguimiento hasta su conclusión, ventaja que se comparte en el sitio de red que lo difunde[3].

Pero a pesar de que existe un gran número de herramientas que pueden ser instaladas como recursos de seguridad en el sistema o bien los medios para conseguir su solución, ninguna de ellas hará el trabajo del administrador, las alarmas colocadas, los monitoreos y las mismas herramientas tendrán que ser revisadas, actualizadas e implementadas una y otra vez para que el sistema conserve su estabilidad, entonces podremos hablar de un resguardo adecuado de nuestra información; podríamos aplicar las auditorías a las que hacíamos referencia, y mejor aún garantizar la integridad del mismo. Una vez logrado el primer objetivo: un sistema estable y la aplicación adecuada de estas herramientas, podríamos explotar todo el potencial de los recursos a nuestro alcance y diseñar un sistema con características de seguridad más sofisticadas que no sólo provean una barrera que podría ser superada por aquellos que al igual que nosotros conocemos el código fuente y por tanto la forma de alterarlo, entonces haremos uso de otro recurso conocido como IDS (*Intrusion Detection Systems*), cuyas siglas en idioma inglés se traducen al castellano como Sistema Detector de Intrusos, el cual mediante un monitoreo permanente del sistema en que

[3] http://www.unam-cert.unam.mx, http://www.seguridad.unam.mx, ftp://ftp.seguridad.unam.mx.

ha sido colocado, envía mensajes a su administrador para que sea revisado y la falla sea rastreada para su eliminación. Parece un camino sin fin pero, se ha demostrado que funciona y en esta base descansa la propuesta de la presente tesis.

El IDS como herramienta, provee los elementos para monitorear continuamente el comportamiento del sistema y reconocer a un intruso antes de que se altere la información, debemos mencionar que el IDS puede estar basado en el funcionamiento de una herramienta de seguridad como es el caso de ANA (*Auditor de Nocividad de Archivos*), pero a diferencia de una herramienta, puede detectar al intruso antes de causar el daño, las herramientas son monitoreadas con sistemas creados por los propios administradores, el IDS trabaja de forma automática después de ser instalado pero al igual que las herramientas puede ser mejorado con características muy específicas a las tareas que deseamos monitorear; el IDS responde a las necesidades especificas para las que ha sido creado y por tanto su estructura es “privada” aún cuando también puede ser creado bajo software libre. Su implementación es redituable en la medida en que nosotros explotemos su uso, y mejor aún, su uso implica prevenir la repuesta a posibles ataques, su desarrollo demanda de manera lógica la etapa de pruebas que cualquier sistema enfrenta antes de volverse estable, pero el IDS que se propone en este trabajo puede ser programado para reconocer el ataque y reconstruir el sistema con la información que ha sido almacenada, para reparar el daño y recuperar el control.

## *1.1 Antecedentes de intrusos*

Las actividades relacionadas con la seguridad en cómputo iniciaron al mismo tiempo que comenzó a difundirse la utilización de las computadoras; y en la misma forma ocurrió el surgimiento de los virus informáticos que son una de las principales preocupaciones en cuestión de seguridad, y como referencia se presenta a continuación una relación cronológica de estos hechos.

### 1.1.1Seguridad

- **1950s**: Establecimiento de la primera organización gubernamental de seguridad en los Estados Unidos: el U. S. Communications Security Board (COMSEC, Consejo de Seguridad en Comunicaciones).
- **1967**: La Conferencia Conjunta de Cómputo de Primavera (Spring Joint Computer Conference) de este año se considera como el sitio donde se llevó a cabo la primera presentación detallada de seguridad en cómputo. En esta sesión de conferencias, coordinada por Willis H. Ware de la Corporación RAND, se presentaron ponencias que tocaban diversos problemas de seguridad en cómputo, enfocadas a un auditorio técnico. En Octubre de ese año, el Departamento de Defensa de los Estados Unidos (DOD) formó un grupo de trabajo auspiciado por el Consejo de Ciencias de la Defensa dentro de la Agencia de Proyectos de Investigación Avanzada (ARPA, actualmente conocido como DARPA), cuya función era examinar sistemas y redes de cómputo, identificar

vulnerabilidades y amenazas, e introducir métodos para proteger y controlar el acceso a las computadoras, sistemas y redes del DOD.

- **1968**: La Oficina Nacional de Normas (National Bureau of Standards o NBS) realiza un estudio inicial para evaluar las necesidades de seguridad en cómputo del gobierno.
- **1970**: Se publica el documento Security Controls for Computer Systems, documento, publicado inicialmente como clasificado, fue una publicación histórica para la seguridad en cómputo, pues fijó la pauta que seguirían muchos grupos de trabajo e investigaciones en años subsecuentes.
- **1972**: Este año un grupo de investigadores del MIT presentaron el primer "Protocolo para la transmisión de archivos en Internet que sentó las bases para el futuro protocolo de transmisión de archivos (FTP)
- **1972**: El DOD emite una directiva y un manual adjunto que establecían una política consistente para todos los controles y técnicas de seguridad en cómputo dentro del DOD. NBS patrocina una conferencia sobre seguridad en cómputo en conjunción con la ACM (Association for Computing Machinery). En este año, y sentando un precedente a nivel mundial, Japón adopta una Política Nacional de Cómputo, en la que ya se consideraban aspectos de seguridad.
- **1973**: NBS inicia un programa para estudiar estándares de desarrollo para mecanismos de seguridad en cómputo. NBS lanza una invitación para presentar técnicas de cifrado de datos que pudieran ser utilizadas como base para algoritmos de cifrado. Uno de los algoritmos presentados a concurso es el DES (Data Encryption Standard), que posteriormente sería adoptado como el estándar de cifrado de datos por el Gobierno de los Estados Unidos.
- **1970s**: Durante el transcurso de esta década comienzan a surgir, normalmente bajo el patrocinio del DOD y de empresas privadas, los llamados tiger teams (“equipos de tigres”), que eran equipos de hackers que intentaban violar los mecanismos de seguridad de los equipos de cómputo, reportando sus resultados con el fin de corregir los problemas encontrados. Estos equipos ayudaron a localizar muchos problemas existentes en sistemas de cómputo de la época, pero tenían el problema de que atacaban fallas específicas en sistemas específicos, y su mecanismo de trabajo era prácticamente “prueba y error”. Por consiguiente, era muy difícil que encontraran todos los problemas existentes, y finalmente se vio que eran necesarios mecanismos más generales y formales de análisis de seguridad.
- **1970s**: durante los 70’s, se iniciaron muchos trabajos de investigación formales sobre seguridad, sentando las bases del estudio teórico de la seguridad en cómputo. Durante este tiempo se desarrollaron los conceptos de política de seguridad y modelo de seguridad. También se realizó el primer modelo matemático de un sistema de seguridad multinivel, que ha sido la base de muchos estudios teóricos e implementaciones hasta nuestros días.
- **1977**: El Departamento de la Defensa de los Estados Unidos anuncia la Iniciativa de Seguridad en Cómputo del DOD, bajo el auspicio de la Secretaría de la Defensa para la Investigación y la Ingeniería (Secretary of Defense for Research and Engineering). Esta iniciativa tenía como objetivo atraer la atención y los recursos nacionales hacia la seguridad en cómputo, y lo logró organizando una serie de seminarios y talleres de trabajo con la industria y el gobierno para tratar problemas de seguridad. Estos talleres

resultaron en la publicación de numerosos reportes. Un aspecto digno de mencionarse es que, desde entonces, se reconoció el hecho de que ningún sistema de cómputo puede considerarse completamente seguro. El reporte del taller realizado en 1977 dice: De acuerdo a cualquier definición razonable de "seguro", ningún sistema operativo actual puede considerarse "seguro". Se acepta además al algoritmo DES (Data Encryption Standard) como estándar federal de procesamiento de información (Federal Information Processing Standard, FIPS), con lo cual se convirtió en el método oficial de protección de información en las computadoras de las agencias del Gobierno de los Estados Unidos.

- **1981**: El 2 de Enero de este año, se forma el Centro de Seguridad en Cómputo del DoD (DoD Computer Security Center, CSC) dentro de la Agencia Nacional de Seguridad (National Security Agency, NSA), con el fin de continuar el trabajo comenzado por la Iniciativa de Seguridad en Cómputo del DOD. Por otro lado se define el protocolo TCP/IP (Transfer Control Protocol / Internet Protocol) y ARPANET lo adopta como estándar en 1982, sustituyendo a NCP. Son las primeras referencias a Internet, Internet es la abreviatura de Interconnected Networks, es decir, Redes interconectadas, o red de redes.
- **1983**: Se publica la primera versión del Trusted Computer System Evaluation Criteria (TCSEC), comúnmente conocido como Libro Naranja debido al llamativo color de su cubierta, y que se convertiría (hasta nuestros días) en el estándar por el cual se mide la seguridad de los sistemas de cómputo a nivel mundial. Este documento fue publicado por el NCSA, basándose en los trabajos previos de la Corporación Mitre y en trabajos de investigación como el de Bell- LaPadula.
- **1984**: El presidente de los Estados Unidos Ronald Reagan firma, el 17 de Septiembre de este año, la Directiva de Decisión de Seguridad Nacional 145 (National Security Decision Directive 145, NSDD 145), también conocida como la Política Nacional de Seguridad de Sistemas Automáticos de Telecomunicaciones e Información. Este documento tuvo consecuencias importantes en el mundo de la seguridad en cómputo, al establecer reglas para el manejo de información clasificada, y autorizar a la NSA a aconsejar a la iniciativa privada.
- **1985**: El CSC se convierte en el Centro Nacional de Seguridad en Cómputo (National Computer Security Center, NCSC), al expandirse sus responsabilidades a todas las agencias federales de los Estados Unidos. NSA combina sus funciones de seguridad en cómputo y comunicaciones bajo la Dirección de Sistemas de Seguridad de la Información, conocido comúnmente como INFOSEC.
- **1986**: Se expide la Ley sobre Fraude y Abuso en Cómputo (Ley pública 99-474 en los Estados Unidos), que establece las penalizaciones por acceso no autorizado o fraudulento a computadoras del gobierno.
- **1990**: Se publica en Inglaterra la Ley sobre Mal Uso del Cómputo, que define los delitos de seguridad en cómputo y las penas aplicables por ellos.
- **1992**: Se publica el Trusted Information Technology Security Evaluation Criteria (ITSEC) en Alemania. Este documento es la contraparte europea del Libro Naranja, y pretende tener una visión más amplia que pueda convertirse, en algún momento, en un estándar internacional de seguridad en cómputo.

- **2003**: El costo del impacto mundial de los ataques cibernéticos creció constantemente de 3 mil 300 millones de dólares en 1997 a un monto estimado de 12 mil millones de dólares en 2003, según el Computer Economics de Carlsbad, California.
- **2003**: El Centro de Coordinación (CERT) de la Universidad Carnegie Mellon informó que el número de incidentes de seguridad de TI aumentó, pasando de 52.658 en el año 2001 a 82.094 en el 2002. Solamente en el primer trimestre de 2003, se registraron 42.586 incidentes. Según el último Informe sobre las amenazas a la seguridad en Internet de Symantec, las amenazas combinadas representaron el 60% de los códigos maliciosos presentados durante el primer semestre de 2003 y presentaron un aumento del 20% hacia el final del año.

### 1.1.2 Virus informáticos

- **1949**: la primera referencia de un programa que, al igual que los virus biológicos, es capaz de reproducirse solo data de 1949, cuando el matemático John Von Neuman mencionó el concepto en su artículo «Theory and Organization of Complicated Automata».
- **1970**: virus Creeper difundido por la red ARPANET. mostraba el mensaje "SOY CREEPER...ATRÁPAME SI PUEDES!". Ese mismo año es creado su antídoto: el antivirus Reaper cuya misión era buscar y destruir al *Creeper*.
- **1983**: Fred Cohen, un estudiante de la Universidad de California del Sur, presentó un experimento el 10 de noviembre de 1983. Cohen utilizó el término virus informático, dentro de su tesis doctoral en la que desarrolló una teoría matemática sobre la expansión de estos destructores de programas. Hoy, Fred Cohen dirige su propia empresa de seguridad informática, «Fred Cohen & Associates».
- **1986**: Brain. Considerado como el primer virus informático encontrado fuera de un laboratorio. Infectaba el sector de arranque de los disquetes, utilizando técnicas de enmascaramiento para hacer que el ordenador no se percatara de su presencia. Fue el primer virus reseñado en los medios de comunicación.
- **1987**: el primer caso de contagio masivo de computadoras a través del MacMag Virus también llamado Peace Virus sobre computadoras Macintosh.
- **1987**: Jerusalem También llamado «Viernes 13», porque los efectos del virus tenían lugar en tal fecha. Uno de los primeros en infectar archivos, los borraba cuando se ejecutaban. Es uno de los virus más famosos de la historia, por su técnica de programación y porque a partir de él se crearon muchas variantes. Fue descubierto a finales de 1987 en la Universidad Hebrea de Jerusalem.
- **1992**. Michelangelo Infectaba el sector de arranque de los disquetes y el registro maestro de arranque (MBR) de los primeros discos duros el día 6 de marzo, coincidiendo con el aniversario del nacimiento de Miguel Ángel Bounarroti.
- **1995**. Concept primer virus de macro. Aprovechaba el «lenguaje» de macro que Microsoft creó para automatizar tareas en Microsoft Office. Infectaba archivos que anteriormente se pensaba que sólo contenían datos y no un código ejecutable.
- **1998**: el virus difundido por Internet más destructivo ha sido el CIH. el "I love you" que como Melissa, llegó vía correo electrónico y como archivo adjunto.

- **1999**. Chernobyl o CIH formateaba el disco duro y, además, conseguía afectar al hardware del ordenador infectado, dado que era capaz de reescribir la memoria Flash BIOS del equipo, con lo que no podía arrancar y quedaba inservible.
- **1999**. Melissa Primer virus de macro con capacidad de propagación a través del correo electrónico. Se difundía a las direcciones de la libreta de direcciones de Outlook/Outlook Express.
- **1999**. Funlove Primer gran virus de red, que hoy en día todavía se encuentra activo y provoca infecciones en redes empresariales.
- **2000**. VBS_Loveletter primer gusano de correo escrito en VBScript. Hasta entonces los «scripts» ejecutables de Windows habían pasado inadvertidos a los expertos en seguridad.
- **2000**: los problemas que se plantearon de las computadoras conocido como "año 2000" (Y2K). dentro de los muchos sistemas del gobierno estadounidense, originaron que el presidente Clinton formara un consejo de más de 30 agencias. cuya meta primordial consistió en mantener los servicios gubernamentales básicos operando más allá del 2000 con sistemas corregidos por medio de un "planes de contingencia"; que permitieron que los sistemas siguieran operando luego de ocurridas las condiciones de falla. En este concepto y planificación trabajó un gran Grupo de Investigación en Seguridad y Virus Informáticos de distintos países, para evitar la perdida o alteración de información en las bases existentes alrededor del mundo.
- **2001**. Code Red primer virus con capacidad de propagación mediante http. Es la primera vez que se usa una vulnerabilidad de seguridad como medio de propagación. En este caso utilizaba una debilidad de Internet Information Server
- **2001**. Nimda Uno de los primeros virus con su propio motor SMTP, de forma que no necesita que el usuario tenga configurado un servidor de correo. A partir de este momento, con una conexión a Internet es suficiente para empezar a propagarse. Utiliza la vulnerabilidad IFRAME, que le permite infectar sin necesidad de que el usuario abra el correo.
- **2001**. Klez Uno de los virus más persistentes de todos los tiempos. Se propaga por mucho medios e infecta sin abrir el mensaje. Reúne todas las características de los virus más dañinos.
- **2001** puede ser recordado como el año de los gusanos de Internet. Utilizando su propia tecnología dentro del servicio de correo explotando vulnerabilidades conocidas. Cada uno de los 12 virus más prevalecientes de 2001 se capitalizaron para su diseminación en el correo electrónico Microsoft Outlook. El virus que estuvo en la cima Sircam, fue el primero que llegó. Central Command's Emergency Virus Team vio gusanos disfrazados de: estrellas de tenis, actrices, personajes de Disney, desnudos para personas curiosas.

| Rango | Nombre del gusano | Porcentaje |
|---|---|---|
| 1 | I-Worm.Sircam.A | 22.7% |
| 2 | I-Worm.Badtrans.B | 17.9% |
| 3 | Win32.Nimda.A@mm | 17.6% |
| 4 | I-Worm.Hybris.B | 7.0% |
| 5 | Win32.Magistr.A@mm | 6.4% |
| 6 | Win32.Goner.A@mm | 4.1% |
| 7 | VBS.Homepages.A@mm | 2.6% |
| 8 | I-Worm.MTX | 2.0% |
| 9 | VBS.SST.A | 1.5% |
| 10 | I-Worm.KAK | 1.3% |
| 11 | Win32.Magistr.B@mm | 1.1% |
| 12 | I-Worm.Badtrans.A | 1.0% |
| 13 | Otros | 14.8% |

- **2002**, Los 'hackers' hicieron un total de 30.839 ataques informáticos, el mayor índice de sitios web '*hackeados*' desde que surgiera este fenómeno en 1995, según un Informe presentado por la empresa de seguridad mi2g.
- **2002**. Bugbear Es un virus complejo al igual que Klez, que utiliza la vulnerabilidad IFRAME. Se replica mediante unidades compartidas de red, detiene los procesos antivirus de la máquina. Y para 2003 surgió una variante, Bugbear.B.
- **2003**. Slammer, ataca a sistemas tan críticos para las empresas como son las bases de datos. Se propaga utilizando una vulnerabilidad de SQL server. Se replicaba de forma tan continuada que llegó a colapsar todas las redes infectadas. Sólo funciona en memoria, así que es muy difícil de erradicar con la tecnología de archivo de firmas. Blaster, Sobig F y Mimail. Blaster llegó tan solo 26 días después de que Microsoft descubriera una deficiencia en Windows RPC DCOM y sacara un parche para los sistemas vulnerables. El gusano aprovechó lo que los expertos en seguridad denominan la deficiencia en Windows de mayor expansión. Durante un tiempo, Blaster llegó a infectar hasta 2.500 equipos por hora.
- **2003**: Los administradores del sistema que asistieron al congreso sobre *hackeo* DefCon informaron que los atacantes en línea estaban utilizando un programa para comprometer a los servidores de Windows y controlarlos remotamente a través de las redes de charla interactiva en Internet (IRC). Varios programas, incluyendo uno que aprovechaba la vulnerabilidad de Windows RPC DCOM, crearon rápidamente una herramienta de ataque remoto. La herramienta podía aceptar órdenes de un atacante a través de las redes de IRC y explorar y comprometer equipos vulnerables a la deficiencia en Windows. Un estudio que correlacionó aproximadamente 1,5 millones de exploraciones durante año y medio, descubrió que el 50 % de todos los sistemas vulnerables seguían sin parches 30 días después de que los parches de seguridad del software salieran al mercado. El estudio, realizado por Qualys y publicado en julio, también reveló que el 80 % de los ataques se lanzaron en los primeros 60 días después del anuncio de la vulnerabilidad.

## *1.2 Definiciones*

### 1.2.1 Seguridad Informática

Podemos entender como seguridad a la característica de cualquier sistema (informático o no) que nos indica si éste está libre de peligro, daño o riesgo.

Hablar de seguridad informática implica hablar de información, un concepto del que sólo podemos analizar sus características; pero la información está formada por datos o series de éstos, al manejarla debemos considerar si debe ser del conocimiento general y accesible para todos los usuarios que lo deseen y por tanto ser *pública*; por otro lado tendremos la información que sólo puede ser visualizada por un selecto grupo que trabaja con ella, y a la que se le denomina *privada*; siendo esta última en la que debemos centrar todos nuestros esfuerzos consideremos como las características esenciales de nuestra información las que a continuación comentaremos.

Iniciemos por señalar tres principales: Confidencialidad, Integridad, y disponibilidad, todas ellas con relación a las tecnologías de la información, y que son mejor conocidas como Modelo CIA[4].

**Confidencialidad**

La primera característica del modelo CIA es implementar las medidas de seguridad que garanticen que la información sólo se encuentra disponible para aquellos que tengan necesidad de conocerla con previa autorización. En estos casos de falta de confidencialidad en la información puede provocar severos daños a su dueño -como es el caso de los antecedentes médicos de algún paciente- o volverse obsoleta como los planes de desarrollo de un producto que se “filtra” a una empresa competidora.

Los datos en que la confidencialidad es el objetivo principal suelen encontrarse segmentados y contar con medidas estrictas de control de acceso para impedir que se introduzca alguien no autorizado.

**Integridad/Autenticidad**

Implementar medidas de seguridad para garantizar que los datos son fiables y que no se han modificado es la segunda característica del modelo CIA.

La integridad/autenticidad resulta crítica cuando los datos se utilizan para llevar a cabo transacciones, análisis estadísticos o cálculos matemáticos, el objetivo es conseguir que el contenido permanezca inalterado a menos que sea modificado por el personal autorizado, y esta modificación sea registrada para posteriores controles o auditorías.

[4] Mike Horton y Clinton Mugge Claves Hackers McGraw Hill / Interamericana Madrid 2004; ISBN: 84-481-4052-4

Limitar el acceso a la lectura de los datos puede que no sea un tema crítico, pero impedir y detectar cualquier modificación no autorizada puede convertir se en nuestro principal objetivo; pero también una falla de integridad/autenticidad puede estar dada por anomalías en el hardware, software o virus informáticos.

**Disponibilidad**

Implementar las medidas de seguridad necesarias para garantizar que los datos son accesibles en el momento que se necesiten es la tercera característica.

La disponibilidad puede resultar de crítica importancia cuando se tenga que acceder a los datos o a las aplicaciones en tiempo real; esta última característica consiste en impedir el acceso no autorizado o la manipulación de los datos a la vez que se garantiza el acceso autorizado.

Esto requiere que la información se mantenga correctamente almacenada con el hardware y el software funcionando perfectamente y que se respeten los formatos para su recuperación en forma satisfactoria, los temas relacionados con la disponibilidad juegan con frecuencia, un papel decisivo en la forma en que se implementa una arquitectura o un sistema de seguridad.

Un diagrama que expresa claramente los aspectos relacionados con la Seguridad Informática es el siguiente.

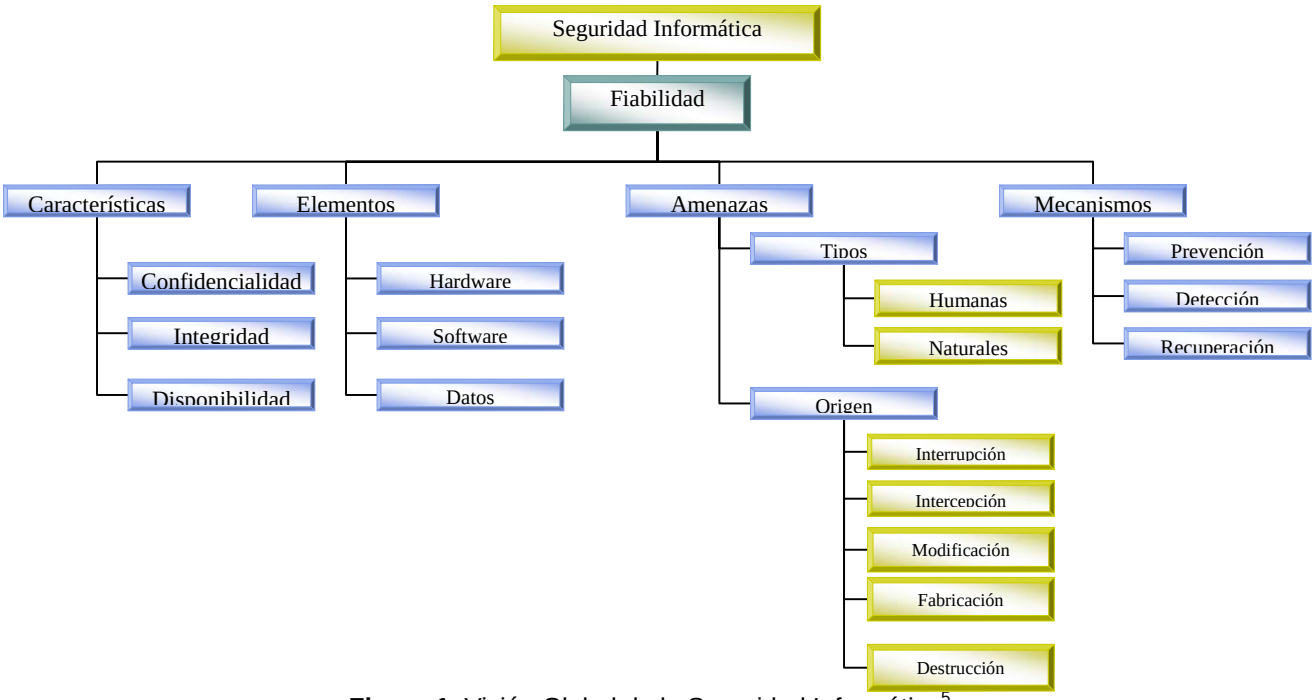



**Figura 1**: Visión Global de la Seguridad Informática[5]

[5] Antonio Villalón Huerta, Tesis: "Seguridad en UNIX y redes", México 2002

### 1.2.2 Amenazas a la Seguridad en Cómputo

Una amenaza es una persona entidad, evento o idea que plantea algún daño a un activo, las amenazas para la seguridad en cómputo se accionan como un mecanismo que puede ser analizado en tres momentos importantes: el previo al ataque, durante el ataque y después del ataque

- **Prevención (antes)**: mecanismo que aumenta la seguridad o fiabilidad de un sistema durante su funcionamiento normal, como es el caso del cifrado de la información.

- **Detección (durante)**: es el mecanismo orientado a revelar intromisiones a la seguridad. y generalmente son programas de Auditoría.

- **Recuperación (después)**: mecanismos que se aplican después de haber detectado una intromisión al sistema y se trabaja para regresarlo a funcionamiento normal, por ejemplo la recuperación desde las copias de seguridad.

Evidentemente al analizar los tres momentos a los que hemos hecho referencia, podríamos pasar ahora a establecer el tipo de amenazas a las que se encuentra expuesto un sistema de información.

Las amenazas se han clasificado de diversas formas pero en esta tesis se citan sólo dos de ellas.

**Clasificación de amenazas [6](primera):**

1. **Amenazas fundamentales**: afectan directamente los cuatro objetivos básicos de la seguridad: fugas de información, violación a la integridad, negación de servicios y uso legítimo.

2. **Amenazas habilitadoras de las primarias**: Son importantes porque la realización de cualquiera de estas amenazas puede conducir directamente a la realización de las amenazas fundamentales. Estas son:

- *Suplantación*: Una persona o entidad pretende ser otra diferente. Es la forma más común de penetración al perímetro de seguridad.
- *Sobrepasar los controles*: Un atacante explota las fallas de un sistema o debilidad de seguridad para adquirir acceso no autorizado a los recursos para obtener privilegios.
- *Violación con autorización*: Una persona autorizada para utilizar un sistema o recurso, lo utiliza para lograr un propósito no autorizado. Es conocido como amenaza interna.
- *Caballo de Troya*: Un software que contiene una parte invisible de código, la cual cuando se ejecuta compromete la seguridad del sistema.

[6] Morant R. José Luis, 1994. Seguridad y protección de la información. 1ª edición. Madrid.

- ✧ *Puerta Trasera*: es una característica incorporada en un software que ante un evento o entrada ejecuta acciones que pueden comprometer la seguridad del sistema.
- ✧ *Bombas Lógicas*: Son códigos adicionados a los programas que ante ciertas fechas o tiempo de ejecución ejecutan acciones perjudiciales para el sistema.
- ✧ *Virus*: Son programas que se autoreplican y afectan principalmente los archivos ejecutables, a veces llegan a afectar a miles de computadores.

3. **Amenazas subyacentes**: si analizamos cualquiera de las amenazas fundamentales o de habilitación de las primarias en un ambiente dado, podemos identificar amenazas subyacentes particulares, cualquiera de las cuales puede habilitar las amenizas fundamentales. Por ejemplo si consideramos la amenaza fundamental de fugas de información podemos encontrar varias amenazas subyacentes, tales como: escuchar sin autorización, Análisis de tráfico, Indiscreción por personal, Reciclaje de medios, etc.

**Esquema de Clasificación de Amenazas a la Seguridad (segunda)**

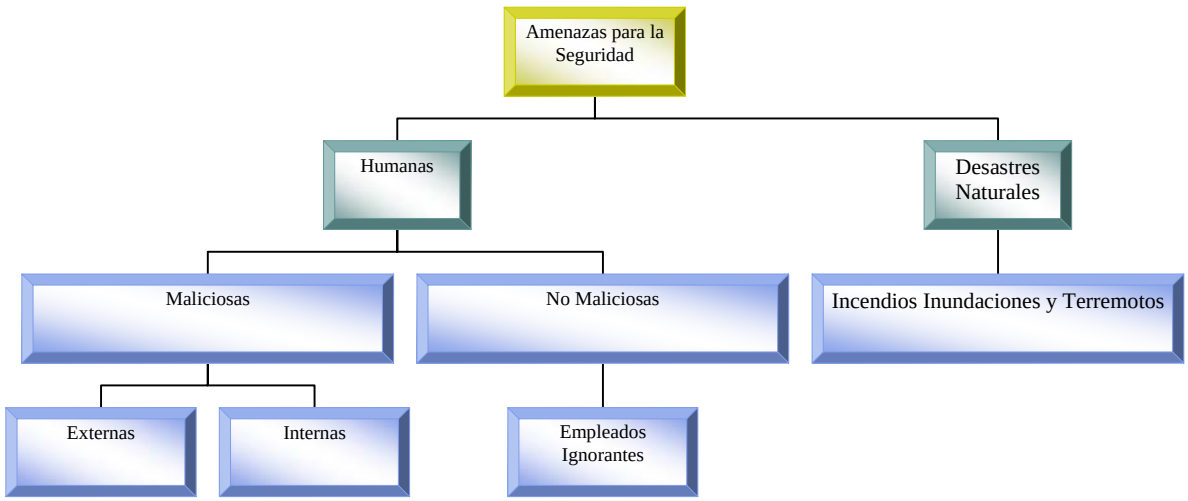



**Figura 2**: "Amenazas para la Seguridad"[7]

El esquema de la figura 2 muestra una división simple de las amenazas a la seguridad de un sistema, en comparación con los planteamientos hechos en la primera clasificación, sin embargo lo que verdaderamente nos ocupa es la forma de implementar herramientas de seguridad.

Nuestro planteamiento no debe ser exclusivamente defensivo, pues con ésto sólo lograríamos resolver el problema de manera parcial, en la realidad, debemos considerar la

[7] Tesis "Seguridad Informática" de Cristian F Borghello, 2001

presencia continua de riesgo o amenaza de violación a la integridad del sistema, y preguntarnos:

1. ¿Cuánto tardará la amenaza en superar la “solución” planteada actualmente?
2. ¿Cómo se hace para detectar una amenaza a tiempo?
3. ¿Cómo se neutraliza una amenaza?

Es aquí en donde el administrador del sistema debe establecer las políticas de prevención, de detección y de recuperación respectivamente.

Para responder más ampliamente, definiremos el término **riesgo** que citamos en el párrafo anterior, como la posibilidad de un daño sobre un bien, sea un acto natural, errores u omisiones humanas o actos intencionales, por lo que una buena forma de resolverlo sería la siguiente.

1. Minimizar la posibilidad de su ocurrencia de una amenaza
2. Reducir al mínimo el daño producido si no ha podido evitarse su ocurrencia
3. Diseñar métodos para la más rápida recuperación de los daños registrados
4. Mejorar las medidas de seguridad utilizadas en función de la experiencia dada.

Si analizamos el **daño**, como un resultado de la efectividad de la amenaza, debido a que no se detecto a tiempo o bien que al ser detectada no se atendió de forma adecuada, en este contexto el administrador del sistema será el encargado y principal responsable, de detectar cada una de las **vulnerabilidades** (debilidades) que pueden ser utilizadas por las diversas amenazas de origen humano o natural, para comprometer la integridad de la información.

Conseguir que un sistema esté libre al 100% de riesgo o daño es una característica muy difícil de conseguir o mejor dicho imposible, como ya en otras ocasiones los expertos han mencionado; es por eso que la **fiabilidad** queda expresada como la probabilidad de que un sistema se comporte tal y como se espera de él, y que el día de hoy se hable de sistema fiable, en vez de un sistema seguro.

### 1.2.3 Ataques a la Seguridad en Cómputo

Se puede definir un ataque de una manera concreta como “la realización de una amenaza”, tema que hemos comentado en el apartado anterior; nuestra atención ahora se centrará en identificar las características de los ataques a un sistema de información y sus elementos.

En cualquier sistema informático existen tres elementos básicos a proteger, el hardware, el software y los datos. Por hardware entendemos el conjunto de todos los sistemas físicos del sistema (CPU, cableado, impresoras, CD-ROM, componentes de comunicación); el software son todos los elementos lógicos de hacen funcional al hardware (Sistema operativo, aplicaciones y utilidades); y los datos son el conjunto de información lógica que maneja el software y el hardware (Bases de datos, documentos y archivos). Para cualquiera de los elementos descritos existen multitud de amenazas, pero los ataques pueden ser clasificados en:

**Ataques Pasivos**: El atacante no altera la comunicación, sino que únicamente la escucha o monitorea, para obtener información que está siendo transmitida. Sus objetivos son la intercepción de datos y el análisis de tráfico. Generalmente se emplean para:

- ✧ Obtención del origen y destinatario de la comunicación, a través de la lectura de la cabecera de los paquetes monitorizados.
- ✧ Control del volumen de tráfico intercambiado entre las entidades monitorizadas, obteniendo así información acerca de actividad o inactividad inusuales.
- ✧ Control de las horas habituales de intercambio de datos entre las entidades de la comunicación, para extraer información acerca de los períodos de actividad.

**Ataque Activos**: estos ataques implican algún tipo de modificación del flujo de datos transmitido, la creación de un falso flujo de datos, generalmente son realizados por hackers, piratas informáticos o intrusos remunerados y se les puede subdividir en cuatro categorías, que gráficamente pueden ser observadas en la Figura 3.

- ✧ *Interrupción*: si hace que un objeto del sistema se pierda, quede inutilizable o no disponible.

- ✧ *Intercepción*: Si un elemento no autorizado consigue el acceso a un determinado objeto del sistema

- ✧ *Modificación*: si además de conseguir el acceso, consigue modificar el objeto.

- ✧ *Fabricación*: se consigue un objeto similar al original atacado de forma que es difícil distinguirlos entre sí.

- ✧ *Destrucción*: es una modificación que inutiliza el objeto.

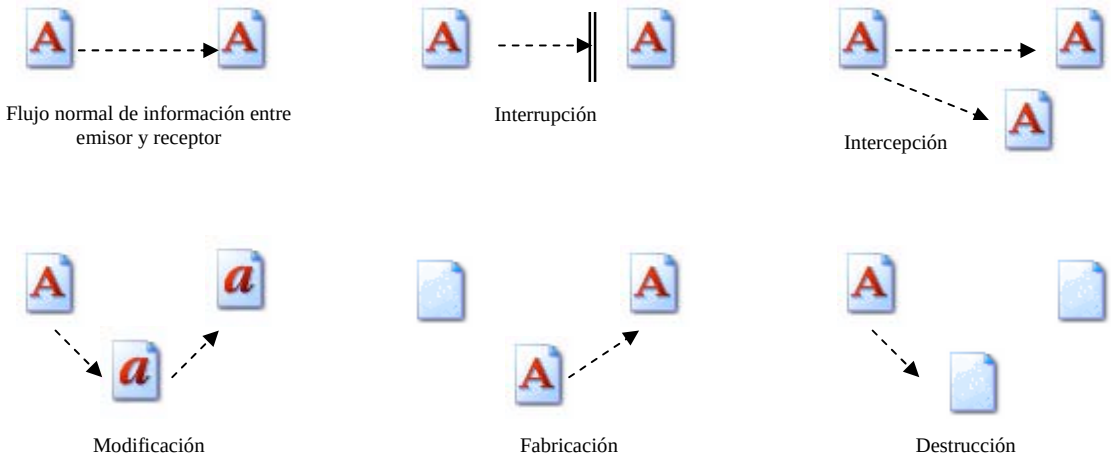



**Figura 3**. Tipos de Ataques Activos[8]

¿Pero cuál es la estructura de un ataque?, algunos autores documentan la forma en que se realizan comúnmente estas intervenciones, por lo que considero interesante dar un vistazo a los árboles de ataque.

## Árboles de Ataque[9]

Una vez identificados los objetivos de alto valor el *hacker* puede comenzar a detectar los puntos y vectores de ataque que afectan al sistema. Los puntos de ataque se definen como las debilidades o los medios de los que dispone un atacante para acceder al sistema o superar los controles. Los vectores de ataque buscan estos puntos en relación con las amenazas humanas, con independencia de que sea un usuario anónimo de la Internet. Combinados y aplicados en etapas sucesivas, los puntos y vectores de ataque se utilizan para modelar los árboles de ataque, el siguiente es un ejemplo de árbol de ataque.

Los modelos de ataque sobre la infraestructura se pueden identificar a nivel macro o microscópico, lo que permite la distribución de las responsabilidades entre diferentes áreas, por lo que los remedios a los ataques suelen ser realizados por los equipos de desarrollo de software de las empresas diseñadoras.

[8] Howard Jhon D. Tesis.An analisys of security on the Internet, 1995 EE.UU.
[9] Mike Horton y Clinton Mugge Claves Hackers McGraw Hill / Interamericana Madrid 2004; ISBN: 84-481-4052-4

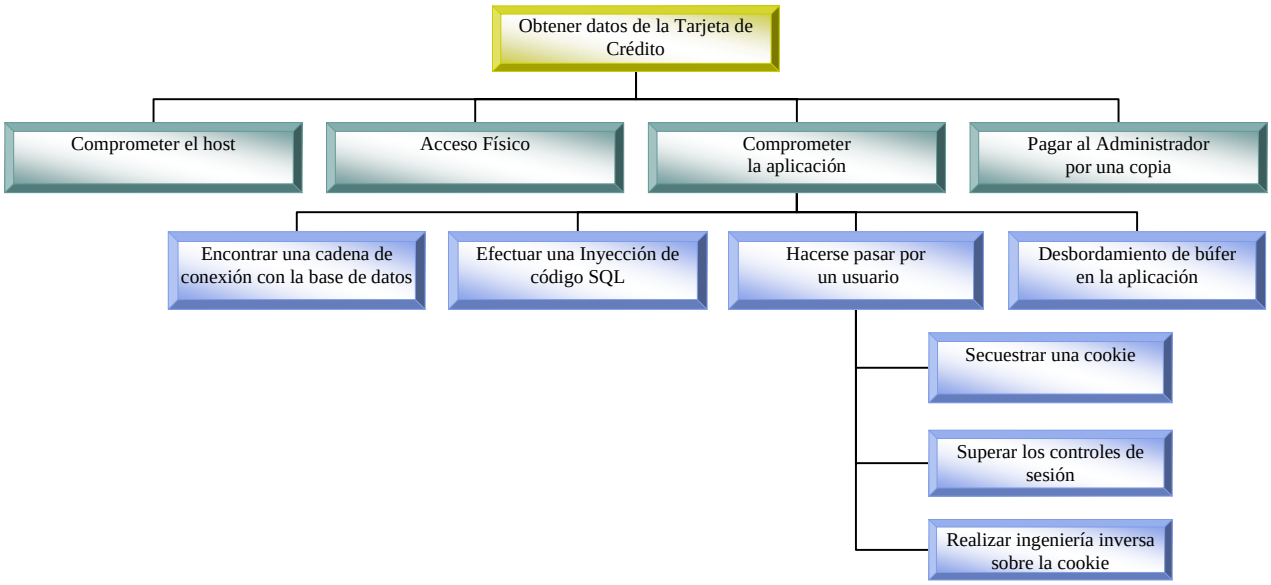



**Figura 4**. Árbol de Ataque

### 1.2.4 Servicios de Seguridad

Después de tener un contexto más amplio sobre el tema de la seguridad informática, las amenazas a los que se enfrenta un sistema de información y la realización de un ataque, plantearemos los servicios básicos desarrollados en un ambiente de comunicaciones, los cuales son categorizados en 5 tópicos.

**Servicio de autenticación**: Provee aseguramiento de la identidad de alguna entidad. Las contraseñas o *passwords* son la forma más común de proveer este servicio. Autenticación es el servicio más importante ya que todos los otros servicios dependen en gran medida de éste. Definimos dos contextos de autenticación.

1. *Autenticación de entidad*, en el cual la entidad remota se identifica ante el servicio, también se conoce como autenticación simple.

2. *Autenticación del origen de los datos:* autentica la fuente real de los datos sin importar que pase por muchos sistemas intermedios.

**Servicio de control de acceso:** Protege contra acceso no autorizado o manipulación de recursos. Contribuye directamente a lograr las metas de seguridad, de confidencialidad, autenticidad y disponibilidad. El modelo general para el control de acceso asume un conjunto de entidades activas llamadas Sujetos (iniciadores) los cuales intentan acceder un miembro de un conjunto de recursos pasivos llamadas Objetos (Destinos)

**Servicio de confidencialidad**: Protege que la información sea divulgada o distribuida a entidades no autorizadas. Se distinguen dos tipos de confidencialidad.

1. *Confidencialidad de datos*: El cual está relacionado con el almacenamiento de información.

2. *Confidencialidad del flujo de tráfico*: El cual está relacionado con el proceso de transmisión de información. Se dan tres casos diferentes de confidencialidad del flujo: confidencialidad de un servicio no orientado a la conexión y servicio de la confidencialidad de campo selectivo.

**Servicio de integridad de datos**: Asegura que los datos no sean cambiados, borrados o sustituidos sin autorización, al igual que el servicio de confidencialidad, diferencia entre integridad de datos e integridad de la comunicación.

Servicio de no repudio: Protege que cualquiera de las dos entidades que participen en una comunicación nieguen que el intercambio ha ocurrido. A diferencia de los anteriores servicios, el objetivo de éste es proteger a los usuarios de la comunicación contra amenazas del otro usuario legítimo más que de atacantes. Cada una de las partes posee pruebas irrefutables ante desacuerdos del origen o destino de los datos. Sin embargo la provisión de este servicio debe considerar el concurso de una tercera parte que ambos extremos de una comunicación confían. Podemos distinguir dos casos.

1. *Repudio de origen*: El destino tiene pruebas demostrables ante terceros de la autenticidad del origen de los datos.
2. *Repudio de destino*: El origen tiene pruebas demostrables ante terceros de la autenticidad del receptor y recepción de los datos.

### 1.2.5 Seguridad de Red

La Internet ha abierto una puerta a millones de usuarios finales, exponiendo en sitios Web, valiosa información corporativa, aplicaciones de gestión de misión crítica e información privada de clientes aumentando el riesgo hasta cotas nunca alcanzadas, en la vulnerabilidad a los ataques de intrusos.

Por lo tanto las conexiones de red merecen en la actualidad una atención especial, por el impacto de los fallos, en el sistema y sus consecuencias a distintos plazos en la integridad de la información.

En el caso de Software libre al tener la opción de ver y estructurar el código, se tiene la ventaja de que los administradores del sistema encargados de la seguridad, rápidamente detectan los fallos y se pueden corregir con una velocidad asombrosa. En la que es importante además actualizar, los mecanismos que se siguen para las conexiones en red, logrando obtener en general un nivel de seguridad y funcionalidad aceptables.

Pero de manera lógica la seguridad será un factor importante desde el momento en que se diseña la red, por lo que sería interesante profundizar en el mecanismo que nos ayudaría a conseguir nuestro objetivo.

**El Diseño de la seguridad de la red**

Al diseñar una red, puede es aconsejable implementar las tecnologías de seguridad adecuadas a la organización, para empezar, es recomendable completar las siguientes tareas a medida que desarrolla el Plan de seguridad de red:

- *Evaluar los riesgos de la seguridad de red***:** La posibilidad de compartir y obtener información conlleva algunos riesgos significativos. Por ejemplo la información de un producto antes de que salga al mercado, alguien podría modificar las páginas Web con malas intenciones o sobrecargar los equipos y dejarlos inservibles. También existe la posibilidad de que algunos usuarios no autorizados pudieran obtener acceso a información que no les concierne. La solución para asegurarse de que sólo las personas adecuadas tienen acceso a recursos y datos, es estudiar cuidadosamente las tecnologías de seguridad de red y planear las estrategias correctamente. Además de dar seguimiento a la forma en que se utilizan los recursos por parte de los usuarios.
- *Determinar el tamaño y emplazamiento de los servidores*: A la hora de establecer una conexión entre una red interna e Internet u otra red pública, se debe considerar cuidadosamente el lugar donde establecerá la conexión. Por lo general, se suele realizar en la parte central de la red de la organización, de modo que haya una distancia efectiva mínima entre los servidores e Internet. Habitualmente se hará también en una ubicación donde los especialistas en redes puedan tener acceso fácilmente para su mantenimiento. Idealmente, se desea tener una sola conexión a Internet para toda una empresa. De esta forma se simplifica la administración de las conexiones y se reduce la vulnerabilidad potencial de la seguridad debido a la aplicación irregular de políticas y procedimientos. Una vez decidido el lugar para colocar la conexión a Internet, deberá determinarse el servidor necesario para respaldar las tecnologías de seguridad de red. Las características de estos servidores dependerán de las tecnologías que piense implementar y de la carga de trabajo prevista .

- *Desarrollar políticas y procedimientos de seguridad***:** Las políticas y procedimientos, en el caso de la seguridad, resultan decisivos. Es necesario crear y publicar las políticas para conseguir consenso acerca de cómo tratar los problemas de seguridad específicos y asegurarse de que todo el mundo las comprenda claramente. Los procedimientos formales aseguran que el mantenimiento del sistema y los cambios deben realizarse siempre de forma bien planeada. Un factor que se debe considerar es cómo controlar las infracciones o intentos de quebrar la seguridad. Y esto se resuelve si hay procedimientos de control implantados en la organización. La definición de políticas ayuda a establecer procedimientos que permiten enfrentarse a estos problemas. La confiabilidad es otro factor de seguridad importante, se debe disponer de planes para cuando falle un componente de la infraestructura de seguridad, y decidir de antemano la

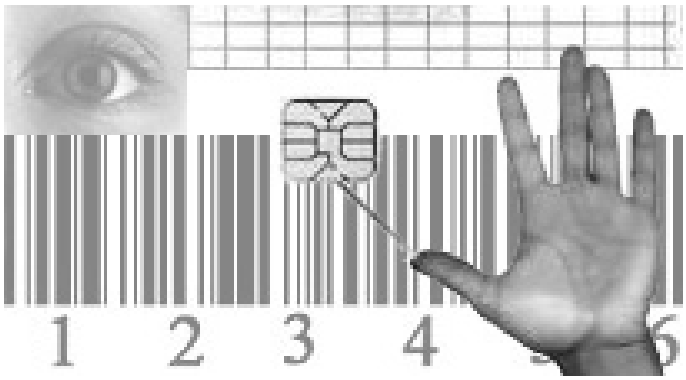
acción apropiada para cualquier posible infracción de la seguridad disponiendo de recursos para reparar el problema tan pronto como sea posible.

- *Crear un plan para distribuir las tecnologías de seguridad***:** Como parte del plan de distribución general de las herramientas en un proyecto; se debe crear un plan detallado para distribuir las tecnologías de seguridad de red, ésto asegurará que las tecnologías se distribuyan de forma sistemática, minuciosa y con una mínima oportunidad de que se produzcan errores. Completando además este proceso mediante la comunicación de las políticas de seguridad de red a todas las partes interesadas y las políticas y procedimientos a los usuarios de la red. Al realizar una prueba piloto. las tecnologías de seguridad de red propuestas deben ser sometidas a pruebas realistas y significativas en un entorno seguro. Esto ayudará a asegurar que la arquitectura funciona como se pretende.

- *Desarrollar estrategias para proteger las conexiones de red***:** La seguridad de red se vuelve más importante cuando los equipos se conectan a una red en la que no se confía enteramente. Los problemas de seguridad de red son comunes, excepto en aquellos casos en que tenga mucho campo de acción para investigar la responsabilidad y aplicar una disciplina, y cuente con una gama de tecnologías de seguridad de red básicas y distribuidas para solucionarlos.
- *Supervisar la seguridad de la red***:** Las tecnologías de seguridad de red que se implementen pueden alcanzar sus objetivos sólo si las planea y configura cuidadosamente. Con una concienzuda preparación, este trabajo se puede efectuar con gran éxito. Sin embargo, puede resultar muy difícil anticipar todos los riesgos posibles: nuevos riesgos que surgen, sistemas que se averían y el entorno cambiante en el que se colocan los sistemas. Las revisiones continuas de las estrategias de seguridad de red pueden reducir estos riesgos. Sin embargo, también es necesario observar la actividad real de la seguridad en la red, para descubrir los puntos débiles antes de que sea tarde y para detener los intentos de quebrantar la seguridad antes de que se hagan efectivos.

# CAPÍTULO II

## HERRAMIENTAS DE SEGURIDAD DEL SISTEMA UNIX




# CAPÍTULO II. HERRAMIENTAS DE SEGURIDAD DEL SISTEMA UNIX

**¿Por qué usar herramientas de seguridad?**
El activo principal de un sistema de cómputo son sus datos, que a diferencia del software y del hardware son insustituibles un problema grave es perder dicha información de este o que fuera alterada intencionalmente por un tercero, para ello es como antes hemos comentado, existe la implementación de diversas herramientas pero éstas no resuelven todo.

**Consideraciones.**[1]

1. **Las Herramientas no resuelven todo**

El administrador no debe olvidar los siguientes principios:

- ✧ El trabajo del administrador no es sustituido por las herramientas.
- ✧ Los sistemas y las herramientas instaladas requieren de mantenimiento
- ✧ Las herramientas de seguridad requieren de actualización, ya que se vuelven obsoletas la configuración o las reglas de uso.

Son las herramientas de seguridad así como las tareas de administración las que nos proporcionarán la confiabilidad sobre el sistema y sus datos.

2. **Deben tomarse en cuenta las siguientes observaciones al momento de instalar cualquier una herramienta de seguridad:**

- ✧ Fuente de los paquetes de instalación.
- ✧ Disposición de firmas digitales
- ✧ Conocimiento del autor o distribuidores.
- ✧ Comentarios positivos o negativos.

Sobre un sistema en producción nunca debe hacerse pruebas de herramientas de seguridad sin el pleno conocimiento de como funciona.

Aún después de haber realizado pruebas, buscar los comentarios positivos o negativos vertidos en listas de discusión en Internet o medios relacionados con la seguridad en cómputo.

Preferir una herramienta madura o probada a ser un conejillo de indias en el caso de los servidores en producción.

3. **Para obtener las herramientas**

Asegurarse de obtener la herramienta de una fuente confiable.

[1] Curso Herramientas de Seguridad en Unix. Alejandro Núñez Sandoval, Departamento de Seguridad UNAM, México 1995

4. **Verificación de la, integridad de los paquetes. Verificar la firma del paquete, usando paquetes de cifrado**

- ✧ pgp
- ✧ gpg
- ✧ md5

Por convención siempre existen archivos con las firmas en el mismo ftp, donde se obtiene las herramientas, seria conveniente hacer comparaciones con firmas de distintos *mirror* del ftp, para tener la certeza de la integridad del paquete.
Al momento de desempaquetar la herramienta no olvide leer los archivos de documentación - README, INSTALL, README.BUG, PORTING, etc. –

5. **Cuidados que deben tenerse al usar la cuenta de administrador**

En el caso de ser una herramienta donde se proporcionen las fuentes, la configuración de la compilación, la compilación debe ser realizada por un usuario no privilegiado.
Al momento de la instalación - make install - debe tenerse la certeza de lo que se está realizando.

Riesgos de instalación de herramientas de seguridad obtenida de fuentes no confiables.

- ✧ configuración deficiente.
- ✧ Código *backdoor*
- ✧ Código troyano.
- ✧ Perdida de información

## *2.1 Herramientas de Autenticación*

En la autenticación en UNIX, cuando un usuario explora una página de un servidor Web, el servidor recibe una solicitud GET de HTTP con un nombre de usuario y una contraseña. El servidor Web comprueba los permisos del usuario en el archivo de acceso apropiado. Si no se puede ejecutar la solicitud GET bajo la autenticación de usuario, el servidor Web devuelve un error 401, que indica que el usuario no está autorizado. Entonces el cliente pide un nombre y una contraseña nuevos y vuelve a enviar la solicitud. Cuando se autentica el usuario, el servidor Web intenta recopilar el contenido del sistema de archivos. Si UNIX no permite que el servidor Web tenga acceso al archivo por alguna razón, devolverá un error fatal al servidor Web. En tal caso, no se pide al usuario un nuevo nombre de usuario.

Cuando un cliente o un explorador de Web envían una solicitud POST de HTTP, como ocurre cuando un autor de FrontPage publica en el programa de creación de FrontPage author.exe, el servidor Web comprueba los permisos del usuario en el archivo de acceso apropiado. Si no se puede ejecutar la solicitud POST bajo la autenticación de usuario, el servidor Web devuelve el error 401. Entonces el cliente pide un nombre y una contraseña nuevos y vuelve

a enviar la solicitud. Cuando se ha autenticado el usuario, el servidor Web intenta ejecutar author.exe. Si el bit SUID de author.exe está establecido, el proceso se ejecuta como propietario del archivo. De lo contrario, author.exe se ejecuta bajo la cuenta UNIX del servidor Web.

Los servidores Web de UNIX normalmente se comunican con los programas cliente mediante la Autenticación básica. Cuando se utiliza la Autenticación básica, el explorador de Web (o el Explorador de FrontPage) pide al usuario un nombre de usuario y una contraseña. El nombre de usuario y la contraseña se transmiten a través de HTTP, ligeramente codificados mediante una técnica llamada UUencoding. (Algunos servidores Web utilizan una técnica conocida como MD5, pero este método no es compatible con FrontPage.)

La Autenticación básica plantea un riesgo de seguridad: los nombres de usuario y las contraseñas se transmiten a través de la red en formato fácilmente decodificable. Sin embargo, la Autenticación básica es la única técnica de autenticación que funciona en un servidor de seguridad a través de un servidor proxy.

### 2.1.1 PASSWD+

**Significado del nombre:** Passwords

**Descripción:** Es un programa de cambio de *passwords* que impide que el usuario escoja *passwords* débiles. El rechazo de *passwords* se basa en un archivo de configuración que permite la utilización de expresiones regulares, comparación con diccionarios o la ejecución de programas externos para examinar el *password*.

El *password* es la primera línea de defensa de una cuenta contra los intentos de acceso no autorizado. Sin embargo, en muchas ocasiones la gente tiende a elegir *passwords* débiles, es decir, que son fácilmente adivinables, como el mismo nombre de la cuenta, el nombre o apellido de la persona, su fecha de cumpleaños, el nombre de su esposa o de algún hijo, etc. Esto hace mucho más sencillo para un atacante el acceso a alguna cuenta legítima del sistema, desde donde es mucho más fácil obtener privilegios mayores.

Passwd+ es un programa que reemplaza al passwd estándar de Unix, cuya función es permitirle al usuario cambiar su *password*. Sin embargo, el passwd+, a diferencia del passwd (con la excepción de algunas versiones), no permite la elección de *passwords* débiles, basándose en una serie de reglas definidas por el usuario. Por supuesto, ésto no elimina los *passwords* débiles que ya existen en el sistema, pero impide que los nuevos usuarios o los que cambien su *password* hagan una selección fácilmente adivinable. Como el archivo de reglas es modificable por el administrador del sistema, la verificación puede hacerse tan estricta u holgada como se desee.

**Desarrollador:** Matt Bishop.

**Disponible en:** ftp://ftp.super.unam.mx/pub/security/tools/passwd+.tar.gz

### 2.1.2 NPASSWD

**Significado del Nombre:** newpasswd

**Descripción:** El comando npasswd, pretende sustituir al comando passwd estándar de UNIX, fue diseñado para reemplazar los programas passwd, chfn y chsh. Npasswd. Permite a un administrador de sistema forzar las políticas de selección de *passwords*, haciendo que las contraseñas sean más seguras evitando que los usuarios elijan unos ``*passwords* vulnerables''. Alguna de las opciones que presenta este programa son las siguientes:

- ✧ Longitud mínima de la contraseña
- ✧ Permite que se pueda obligar a los usuarios a mezclar mayúsculas y minúsculas.
- ✧ No aceptar palabras simples, como una letra repetida.
- ✧ Rechazar contraseñas que contengan el nombre
- ✧ de la máquina u otra información relacionada con ella.
- ✧ Comprueba si la palabra elegida contiene el nombre de cuenta, o el nombre de pila o apellidos.
- ✧ Comprueba si la palabra está incluida en algunos de los diccionarios, incluido el del sistema.

**Desarrollador:** Clyde Hoover

**Disponible en:** ftp://ftp.cc.utexas.edu/pub/npasswd/

### 2.1.3 KERBEROS

**Significado del Nombre:** Es el perro de tres cabezas que en la mitología griega vigila la puerta de entrada a Hades, el infierno.

**Descripción:** Es un sistema de autenticación para redes físicamente inseguras, basado en el modelo de distribución de llaves presentado por R.M. Needham y M.D. Schroeder [NS78]. Permite a los elementos que intervienen en una comunicación identificarse entre si y al mismo tiempo evitar “espionaje” en la red o ataques de repetición. También proporciona integridad en el flujo de datos (detección de modificaciones) y privacidad, utilizando sistemas criptográficos como DES. Kerberos tiene restricciones de exportación hacia fuera de los Estados Unidos.

El uso de Kerberos se produce principalmente en el *login*, en el acceso a otros servidores (por ejemplo, mediante rlogin) y en el acceso a sistemas de archivos en red como NFS. Una vez que un cliente está autenticado o bien se asume que todos sus mensajes son fiables, o si se desea mayor seguridad se puede elegir trabajar con mensajes seguros (autenticados) o privados (autenticados y cifrados). Kerberos se puede implementar en un servidor que se ejecute en una máquina segura, mediante un conjunto de bibliotecas que utilizan tanto los clientes como las aplicaciones; se trata de un sistema fácilmente escalable y que admite

replicación, por lo que se puede utilizar incluso en sistemas de alta disponibilidad (aunque como veremos al final del capítulo está fuertemente centralizado).

Hasta que se diseñó Kerberos, la autenticación en redes de computadores se realizaba principalmente de dos formas: o bien se aplicaba la autenticación por declaración (Authentication by assertion), en la que el usuario es libre de indicar el servicio al que desea acceder (por ejemplo, mediante el uso de un cliente determinado), o bien se utilizaban contraseñas para cada servicio de red. Evidentemente el primer modelo proporciona un nivel de seguridad muy bajo, ya que se le otorga demasiado poder al cliente sobre el servidor; el segundo modelo tampoco es muy bueno: por un lado se obliga al usuario a ir tecleando continuamente su clave, de forma que se pierde demasiado tiempo y además la contraseña está viajando continuamente por la red. Kerberos trata de mejorar estos esquemas intentando por un lado que un cliente necesite autorización para comunicar con un servidor (y que esa autorización provenga de una máquina confiable), y por otro eliminando la necesidad de demostrar el conocimiento de información privada (la contraseña del usuario) divulgando dicha información.

Kerberos se ha convertido desde entonces en un referente obligatorio a la hora de hablar de seguridad en redes. Se encuentra disponible para la mayoría de sistemas Unix, y viene integrado con OSF/DCE (Distributed Computing Environment). Está especialmente recomendado para sistemas operativos distribuidos, en los que la autenticación es una pieza fundamental para su funcionamiento: si conseguimos que un servidor logre conocer la identidad de un cliente puede decidir sobre la concesión de un servicio o la asignación de privilegios especiales. Sigue vigente en la actualidad (en su versión V a la hora de escribir este trabajo), a pesar del tiempo transcurrido desde su diseño; además fue el pionero de los sistemas de autenticación para sistemas en red, y muchos otros diseñados posteriormente, como KryptoKnight, SESAME o Charon se basan en mayor o menor medida en Kerberos.

**Desarrollador** : presentado por R.M. Needham y M.D. Schroeder Mit

**Disponible en:** ftp://athena-dist.mit.edu/pub/kerberos/

### 2.1.4 ANLPASWD

**Significado del Nombre:** formalmente perl-passwd2

**Descripción:** El programa del anlpasswd (antes Perl-passwd2) es un inspector proactive de la contraseña que rechaza deja a usuarios elegir "malas" contraseñas. Un programa de cambio de *passwords* que impide que el usuario escoja *passwords* débiles.

**Desarrollador:** Laboratorio Nacional de Argonne.

**Disponible en:** ftp://ftp.super.unam.mx/pub/security/tools/anlpasswd.tar.gz
http://ciac.llnl.gov/ciac/ToolsUnixAuth.html

2.1.5 SUDO

**Descripción:** Permite al administrador del sistema proporcionar ciertos privilegios a un usuario o un grupo de usuarios de forma que puedan ejecutar algunos comandos como si fueran *root*.

Características:

- ✧ La habilidad de restringir por comandos o usuarios.
- ✧ Sudo genera un registro por cada ejecución de comandos, permitiendo una auditoría de los comandos ejecutados.
- ✧ Sudo implementa ticket timing. Cuando un usuario invoca sudo e ingresa
  - o su *password*, otorgando un ticket por 5 minutos. (este tiempo es configurable en la compilación). Cada comando sudo actualiza el ticket por 5 minutos.
- ✧ Archivo de configuración sudoers, permitiendo una administración flexible y centralizada.

Sudo está orientado principalmente a sistema BSD sin embargo es portable a distintos sistemas UNIX.

**Disponible en** http://www.courtesan.com/sudo/dist

## *2.2 Herramientas de Cifrado*

El Sistema Operativo UNIX incluye diversas herramientas para hacer uso de la Criptografía. Además existen múltiples aplicaciones que permiten usar esta técnica para lograr diversos objetivos relacionados con la seguridad.

Entre los usos que pueden darse a la criptografía en UNIX podemos destacar los siguientes: 1.- Cifrar los *passwords* de los usuarios, 2.- Cifrar los documentos para mantenerlos secretos en la máquina, 3.- Cifrar los mensajes para lograr una transmisión con diversas características de seguridad y 4.- Certificar y autentificar los mensajes enviados por correo electrónico.

Cuando UNIX solicita la introducción de un *password* al usuario, necesita algún método para determinar que el *password* introducido es el correcto. En muchos sistemas antiguos los *passwords* se mantenían completamente visibles en un archivo denominado archivo de passwords. Bajo circunstancias normales, el sistema protegía los *passwords* de modo que estos sólo podían ser consultados por usuarios privilegiados utilizando ciertas órdenes del sistema. Sin embargo este método suponía un grave riesgo para la seguridad, dado que intencionada o accidentalmente estos archivos podían ser leídos por usuarios no autorizados.

UNIX evita este problema manteniendo los *passwords* en un archivo pero en modo cifrado.

Concretamente el valor almacenado se obtiene cifrando un bloque de ceros mediante una función unidireccional implementada en la llamada al sistema crypt(3); el resultado de su aplicación se guarda normalmente en el archivo /etc/passwd. Cuando un usuario accede al sistema lo hace a través del programa /bin/login. En lugar de descifrar el *password* guardado, este programa toma el *password* tecleado y lo utiliza para cifrar un bloque de ceros. El resultado se compara con el bloque guardado y si coincide se autentifica al usuario y se le permite el acceso.

### 2.2.1 MD5

**Significado del Nombre**: MD5 (massage digest 5)

**Descripción**: Es un algoritmo *hash* que toma como entrada un mensaje de una longitud arbitraria y produce un mensaje o huella digital como salida. EL MD5 es usado en firma digital cuando un gran archivo debe comprimirse en una forma segura antes de comenzar a encriptarse con una clave privada (secreta) bajo un criptosistema de clave pública como RSA. El MD5 crea una representación numérica del contenido de un mensaje y la visualiza como valor del hexadecimal de 16 caracteres.

**Desarrollador:** Hans Dobbertin

### 2.2.2 PGP

**Significado del Nombre:** PGP son las siglas de Pretty Good Privacy.

**Descripción:** Se trata de un programa para cifrar y descifrar datos de todo tipo, y resulta especialmente práctico para el uso en correo electrónico. He aquí algunas de sus ventajas:

- ✧ Es un programa gratuito para usos no comerciales
- ✧ Está disponible para múltiples tipos de ordenadores y sistemas operativos.
- ✧ Es un programa de manejo sencillo, especialmente si se usa a través de MS-DOS con un interfaz ("shell"),
- ✧ o en otros sistemas operativos (Windows, Mac, Linux...)
- ✧ Resulta virtualmente indescifrable si la longitud de la clave es lo bastante larga.

Básicamente hablando, PGP funciona como un algoritmo del tipo de clave pública o asimétrica. En un sistema de clave pública, cada usuario crea un par de claves que consiste en una clave pública y una clave privada. Se puede cifrar un mensaje con la clave pública y descifrarlo con la privada (NO se puede cifrar y descifrar con la misma clave). El usuario difunde la clave pública, poniéndola a disposición de cualquiera que quiera enviarle un mensaje. Una vez que el mensaje ha sido recibido por el usuario, éste podrá descifrarlo con su clave privada. Es evidente que la clave privada debe ser mantenida en secreto por el propietario.
Puede considerar este esquema como si fuese un buzón con dos llaves, una para abrir y otra para cerrar. Cualquiera puede introducir un mensaje en el buzón y cerrarlo, pero solamente

el propietario podrá abrirlo. Una gran ventaja de este tipo de esquema criptográfico es que, al contrario que los sistemas tradicionales donde la clave de cifrado y descifrado coinciden, no es necesario encontrar un procedimiento seguro para enviar la clave al recipiente del mensaje.

También permite la opción de "firmar" un mensaje con una firma digital que nadie, ni siquiera el receptor, puede falsificar. Esto resulta especialmente útil, aunque no se cifre el mensaje en sí, porque actúa como certificación de autenticidad, ya que permite comprobar si el mensaje ha sido alterado durante la transmisión. También permite al receptor confirmar que el mensaje ha sido enviado realmente por el remitente (resulta demasiado fácil trucar los encabezamientos de los mensajes de correo electrónico).

La estructura operativa de PGP es bastante curiosa. Para cifrar los datos se emplea un algoritmo de clave simétrica, cuya clave es cifrada con un algoritmo de clave asimétrica. ¿Por qué esta mezcla? Porque de este modo se combinan las mejores propiedades de ambos: la seguridad de un algoritmo asimétrico (donde clave pública y privada son distintas) con la rapidez y robustez de un algoritmo simétrico (cuya clave es única y, por tanto, vulnerable). Un tercer algoritmo se emplea para firmar documentos: se extrae un conjunto de bits del mensaje llamado resumen (*hash*) y se cifra con la clave privada del emisor. Así, el sistema de operación de PGP (y programas similares) consta de tres subsistemas: cifrado del documento, cifrado de clave simétrico y firmado del documento.

PGP, en su popular -aunque ya en desuso- versión para DOS (2.6.3i para usuarios no norteamericanos), utiliza una combinación de los más seguros algoritmos existentes en la actualidad: RSA (Rivest - Shamir - Adleman) para el cifrado de claves, IDEA (International Data Encryption Algorithm) para el cifrado del documento y MD5 (Message Digest Algorithm 5) para la creación de firmas digitales. La clave de tipo Diffie-Hellman, de reciente creación, emplea los algoritmos IDEA para el cifrado de documentos, Diffie-Hellman o DH (variante ElGamal) para el cifrado de la clave, y DSS (Digital Signature Standard) para firma digital. Las versiones modernas para Windows 9x y otros sistemas operativos permiten la elección del algoritmo para cifrado de documentos entre tres de los mejores que se conocen: IDEA, TripleDES y CAST.

En la actualidad, los usuarios pueden elegir entre variantes del programa PGP para diversos sistemas operativos. En esta página y las siguientes nos centraremos en la antigua para MS-dos, conocida como versión 2.6.3i. Ésta ha sido libremente distribuida por todo el mundo y utiliza las claves RSA antedichas; aunque muy potente, es un programa que corre bajo DOS, por lo que requiere interfaces (shells). Las versiones posteriores resultan más fáciles de usar, especialmente cuando se combina con programas de correo electrónico como Eudora u Outlook. Actualmente es objeto de debate entre la comunidad criptográfica, principalmente por la adopción de claves Diffie-Hellman (DH), de nuevo diseño.

**Desarrollador** : desarrollado por Philip Zimmermann. Reelaborado para su uso internacional por Ståle Schumacher.

**Disponible en**: http://www.pgpi.org y http://www.pgp.com

2.2.3 SNEFRU

**Significado del Nombre:** (s. XXVI a.J.C.) Faraón del antiguo Egipto (c. 2600- c. 2576 a.J.C.), fundador de la IV dinastía. Realizó victoriosas campañas en Libia, Nubia y Sinaí. Hizo construir sendas pirámides en Dahshur y Médium.

**Descripción**: Una de las funciones *hash* diseñada por Ralph Merkle, Crea 128 o 256 bit de longitud de valores *hash* usa un algoritmo H el cual realiza un *hashes* 512-bits o m-bits, tomando el primer bit m de salida de H como el valor hash H esta basada en un bloque de cifrado reversible E operando sobre bloques de 512-bit H es el ultimo m-bits de la salida de E XOR'd con el primer m-bits de la entrada de E, E esta compuesto de varios pases, cada pase tiene 64 vueltas de un cicloS-box y XOR, E puede usar de 2 a 8 pases sobre escritura de algoritmo rompe mensaje en bloques de 512-m bits Cada trozo tiene un valor previo de *hash* agregado H esta programado en este valor, dando un Nuevo valor *hash* después del último bloque, el valor *hash* es agregado al valor de longitud del mensaje y H programado sobre este, el valor resultante siendo la MAC.

Snefru ha sido roto por un ataque de Biham y Shamir de *hashes* de 128-bit , y posiblemente de 256-bit cuando de 2 a 4 pases son usados en E Merkle recomienda 8 pases, pero esto es lento, procesar el mensaje en segmentos de 16 palabras (512-bit), usando 3 vueltas de operaciones de 16 bit sobre cada segmento y buffer; el valor de la salida *hash* es el valor final del buffer

**Desarrollador:** Ralph Merkle de Xerox PARC

## *2.3 Herramientas de Comunicación*

Los procesos en UNIX no comparten memoria, ni siquiera los padres con sus hijos. Por tanto, hay que establecer algún mecanismo en caso de que se quiera comunicar información entre procesos concurrentes. El sistema operativo UNIX define tres clases de herramientas de comunicación entre procesos (IPC): los semáforos, la memoria compartida y los mensajes.

2.3.1 OPENSSH

**Significado del Nombre:** OpenSSH (desarrollada por openBSD)

**Descripción:** Qué es OpenSSH ? Es una herramienta de seguridad que permite la conexión de computadoras por medio de un canal cifrado, esta herramienta es muy útil para los usuarios que realizan conexiones a través del servicio telnet, ftp.

Permite entre otras cosas, la conexión remota a una máquina, para poder ejecutar comandos sobre ella, al igual que telnet o rlogin, pero con la ventaja respecto a los anteriores

programas de que la información va cifrada. Esto es, en el caso de que alguien de la misma red en la que nos encontramos, esté usando un *sniffer* para "capturar" las contraseñas para conectarnos a un ftp o a una sesión telnet, con ssh, estará perdiendo el tiempo ya que van cifradas.

**Desarrollador** OpenSSH (desarrollada por openBSD) es la versión libre (licencia BSD) del protocolo SSH.

**Disponible en**: http://www.openssh.org.

2.3.2 PROFTPD

**Descripción:** Proftpd ha sido elige como servidor ftp oficial por la distribución Debian/GNU Linux. Su elección se debe a sus numerosas funcionalidades y por la gran cantidad de programadores que trabajan en él corrigiendo errores. El uso de módulos permite fácilmente extender sus funcionalidades. Existen módulos para:

- Gestión de cuotas (de usuario)
- Otros sistemas de autenticación como LDAP, Mysql, Postgresql, ...
- Gestión de ratios (por ejemplo, el usuario no puede descarga mas de 1 MB diario)
- Gestión de ancho de banda

2.3.3 SSH

**Significado del Nombre**

SSH (o *S*ecure SHell) es un protocolo para crear conexiones seguras entre dos sistemas. Usando SSH, la máquina del cliente inicia una conexión con una máquina del servidor.
**Descripción:** Es una aplicación de seguridad que permite la conexión entre computadoras de forma segura, a través de un demonio sshd.

Encripta la sesión de registro imposibilitando que alguien pueda obtener una contraseña de texto.

SSH está diseñado para reemplazar los métodos comunes para registrarse remotamente en otro sistema a través de la *shell* de comando. El programa scp reemplaza otros programas diseñados para copiar archivos entre *hosts* como por ejemplo ftp o rcp. Ya que estas aplicaciones antiguas no encriptan contraseñas entre el cliente y el servidor, las evita siempre que sea posible. El uso de métodos seguros para registrarse remotamente a otros sistemas hará disminuir los riesgos de seguridad para ambos sistemas y el sistema remoto.

SSH proporciona los siguientes tipos de protección: Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor durante sesiones ulteriores.

El cliente puede transmitir su información de autenticación al servidor, como el nombre de usuario y la contraseña, en formato cifrado.

Todos los datos enviados y recibidos durante la conexión se transfieren por medio de encriptación fuerte, lo cual los hacen extremamente difícil de descifrar y leer.

El cliente tiene la posibilidad de usar X11 aplicaciones lanzadas desde el intérprete de comandos de la *shell*. Esta técnica proporciona una interfaz gráfica segura (llamada *reenvío por X11*), proporciona un medio seguro para usar aplicaciones gráficas sobre una red.

Ya que el protocolo SSH encripta todo lo que envía y recibe, se puede usar para asegurar protocolos inseguros. El servidor SSH puede convertirse en un conducto para convertir en seguros los protocolos inseguros mediante el uso de una técnica llamada *reenvío por puerto*, como por ejemplo POP, incrementando la seguridad del sistema en general y de la seguridad de los datos.

Red Hat Linux 7.3 contiene el paquete general de OpenSSH (openssh), el servidor de OpenSSH (openssh-server) y los paquetes de clientes (openssh-clients). Consulte el capítulo titulado OpenSSH en el Manual oficial de personalización de Red Hat Linux para obtener instrucciones sobre la instalación y el desarrollo de OpenSSH. Observe que los paquetes OpenSSH requieren el paquete OpenSSL (openssl). OpenSSL instala varias bibliotecas criptográficas importantes que ayudan a OpenSSH a proporcionar comunicaciones encriptadas.

Una gran cantidad de programas de cliente y servidor puede usar el protocolo SSH, incluyendo muchas aplicaciones *open source* a disposición gratuita. Hay varias versiones de cliente diferentes de SSH a disposición para casi todos los sistemas operativos más importantes en uso actualmente. Aún si los usuarios que se conectan a su sistema no ejecutan Red Hat Linux, de cualquier manera pueden encontrar y usar un cliente de SSH nativo para su sistema operativo.

**Disponible en** : ftp://ftp.ssh.com/pub/ssh/

## *2.4 Herramientas de Monitoreo de Sistema*

### 2.4.1 COPS

**Significado del Nombre**

COPS significa Computer Oracle and Password System, es un herramienta que pretende descubrir posibles riesgos de seguridad dentro de un sistema a nivel de la administración ya sea por error u omisión al configurar el sistema.

**Descripción:** El principal objetivo de cops es la detección de una gran variedad de posibles problemas de seguridad particulares de Unix como pueden ser:

- Permisos de archivos, directorios y dispositivos del sistema.
- Cuentas sin *password*.
- Contenido, formato y seguridad de los archivos de grupos y *password*.
- Los programas y archivos que corren en /etc/rc* y en la tabla de cron.
- La existencia de archivos SUID propiedad de *root*, y sus características.
- Verificación CRC de binarios.
- ftp anónimo habilitado.
- Posibles vulnerabilidades en los binarios.

Esta herramienta está integrada básicamente en los siguientes módulos:

- Un conjunto de programas que intentan de forma automática detectar potenciales riesgos de seguridad.
- Documentación
- El *script* COPS de ejecución en shell y perl

**Desarrollador** Esta herramienta fue desarrollada por Daniel Farmer y Eugene H. Spafford, en los laboratorios COAST de la Universidad de Purdue.

**Disponible en** ftp://ftp.seguridad.unam.mx/,
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

2.4.2 TARA

**Significado del Nombre:** Tiger Analytical Research Assistant (TARA)

**Descripción:** Es una actualización del programa TAMU tiger. Como tiger no había sido actualizado desde 1994, hubo numerosos cambios en los sistemas, que volvieron prácticamente obsoleto leer un reporte de tiger. TARA se encargó de mejorar estas características así como incluir nuevas características que responderán de forma más adecuada a los cambios del sistema.

TARA es un conjunto de *script* que verifican un Sistema Unix en busca de potenciales problemas de seguridad.

TARA verifica los siguientes puntos en el sistema:

- Aliases
- Nisplus
- Root
- ftp anónimos

- passwd
- permisos de cuentas
- cron
- path
- grupos
- rhosts
- inetd
- sendmail

**Disponible en:** http://www-arc.com/tara/, ftp://ftp.seguridad.unam.mx/Herramientas/

2.4.3 TITAN

**Descripción:** UNIX es a menudo criticado como un sistema complejo, difícil de administrar y difícil de mantener en altos niveles de seguridad. La dificultad de la configuración ha hecho que los vendedores no distribuyan sistemas seguros ya que se requiere gran inversión de tiempo, recursos y expertos en seguridad. Estas son algunas razones por lo que tantos sistemas UNIX son inseguros en le Internet y para hacer frente a estos problemas no solo basta tener las mejores tecnologías ya que el campo de la seguridad cambia rápidamente lo que implica un esfuerzo considerable por parte de los administradores por mantenerse en capacitación constante y actualizarse con la última información en sistemas y redes que sea publicada.
Titan hace el intento de proveer por lo menos una solución parcial a todas estas situaciones tratando de resolver y mejorar muchos de los problemas de seguridad más comunes en el sistema. Titan no reemplaza otras herramientas de seguridad, ni soluciona los *bug's* de seguridad en programas; su propósito es mejorar el sistema basándose en muchos trucos y *tip's* de seguridad, y usando Titan en combinación con otras herramientas de seguridad puede ayudar a transformar una "caja negra" en un *firewall* y hacer la tarea de administración de la seguridad más fácil y eficiente.

Titan es una herramienta de seguridad para equipos de cómputo que se encuentra disponible libremente y puede ser usado para mejorar y auditar la seguridad de un sistema UNIX. Fue escrito casi en su totalidad en Bourne Shell y cuenta con un *script* maestro que controlan la ejecución de muchos otros programas pequeños. Cada uno de los programas arregla o detecta potenciales problemas de seguridad con respecto a la configuración de un sistema UNIX.

Este simple diseño modular también permite llevar un registro en su ejecución para ayudar a verificar el sistema. Finalmente, cualquiera que pueda escribir programas en Shell Script podrá crear sus propios módulos de Titan fácilmente.

¡Podrán dejar de funcionar algunas cosas, pero el sistema será más seguro!

Titan resuelve los siguientes problemas

- Deshabilita puertas de entrada al sistema.
- Minimiza y previene varios ataques de negación de servicio (DoS).
- Habilita y mejora el nivel de monitoreo y auditoría en el sistema.
- Mejora las defensas del sistema y de la red.
- Ayuda en la definición de la programación y mejora las reglas de seguridad del sistema.

Qué no resuelve

Existen varios problemas de seguridad en múltiples programas del sistema, los cuales Titan no intenta arreglar o corregir. Programas como *CGI's* utilizados en servidores de Web, son uno de los problemas de seguridad más extensos en el Internet y es casi imposible descubrirlos o estar previniéndolos. Titan tampoco hace a un sistema impenetrable a los ataques, sólo abarca problemas de *bug's* comunes ya que nuevos *bug's* y agujeros constantemente están siendo descubiertos.

**Disponible en:** ftp://ftp.seguridad.unam.mx/Herramientas/Unix/Monitor_Sistema, http://www.fish.com/titan,

2.4.4 TRIPWIRE

**Significado del Nombre**

**Descripción:** Es un monitor de la integridad de los sistemas Unix. Este usa varias rutinas de *checksum*/mensaje, huella/segura y *hash*/firma para detectar cambios en archivos, además de rastrear determinados campos seleccionados de la administración del sistema.

El sistema tripwire persigue los objetivos de rastrear:

- Cambios en los permisos de los archivos.
- Ligas.
- Tamaños en archivos y directorios.
- Groupid, userid.

**Desarrollador:** En 1993, Gene Kim y Eugene Spafford trabajando en el Laboratorio COAST de la Universidad de Purdue desarrollaron el concepto de software Tripwire que podría ayudar a detectar cambios en la estructura de los archivos o directorios gracias al uso de algoritmos de firmas.

Cada resultado inesperado puede ser investigado para determinar si el cambio fue provocado maliciosamente o accidentalmente. La tecnología de firma digital es utilizada hoy en día en muchos aspectos de la seguridad informática, incluyendo pruebas de identidad, autentificación, autorización, integridad y n repudio.

**Disponible en:** ftp://ftp.asc.unam.mx/pub/tools/tripwire.tar.gz,
ftp://ftp.cert.org/pub/tools/tripwire.tar.Z

2.4.5 NMAP

**Significado del Nombre:** NMAP

**Descripción:** Es una utilería para explorar una red o auditar la seguridad de un sistema. Fue diseñada para barrer de forma rápida una red, NMAP utiliza diversas técnicas de barrido de puertos que le permiten determinar el sistema operativo, los puertos disponibles y los filtros que se detectaron, entre otras características.

**Disponible en:** http://www.insecure.org

2.4.6 SARA

**Significado del Nombre:** SARA Security Auditor's Research Assistant
**Descripción:** Es una herramienta para auditar la seguridad de una red, obtiene tanta información acerca de los *hosts* y redes remotos como es posible, examinando servicios de red como finger, NFS, NIS, ftp y tftp, rexec y otros servicios, investigando cualquier problema potencial de seguridad. Prueba vulnerabilidades de UNIX y Windows, soporta diccionario CVE (Cómmon Vulnerabilities and Exposures), Modo de demonio para accesos corporativos controlados, motor de búsqueda para análisis de auditoría.

**Desarrollado por:** Esta herramienta es derivada de SATAN (Security Administrator Tool for Analyzing Networks) desarrollada por Dan Farmer y Wietse Venema. SATAN es la base de la máquina de Seguridad de SARA.

**Disponible en:** http://www-arc.com/sara/

2.4.7 SAINT

**Significado del Nombre:** SAINT Security Administrator's Integrated Network Tool

**Descripción:** recolecta información acerca de los servicios de los *hosts* y redes, finger, NFS, NIS, ftp y tftp rexd, statd y otros servicios, Analiza y determina si el sistema puede presentar algún potencial problema de seguridad.

**Disponible en:** http://www.wwdsi.com/,
ftp://ftp.seguridad.unam.mx/Herramientas/Unix/Monitor_Sistema/Saint/

## *2.5 Herramientas de Monitoreo de Red*

Para prevenir errores en el sistema existe una computadora que está "monitoreando" el funcionamiento de la red.
Estos errores a menudo se deben a problemas de ruido en la línea de transmisión y crean situaciones que no existen, tales como direcciones de computadoras que no pertenecen a ninguno de los nodos, errores en la información, por mencionar algunos.
Cada una de las computadoras realiza un chequeo sobre la información contenida en el paquete que viaja a lo largo de toda la red, si esta información no es válida por alguna razón, se declara inválido el paquete escribiendo una bandera de error.
Cada uno de los nodos lleva cuenta de los errores que están ocurriendo en la red, de tal forma que si una computadora se da cuenta de que el número de errores excedió a la cuenta permitida, le informa a la computadora que está "monitoreando" a la red, a fin de que pueda

### 2.5.1 TCP WRAPPERS

**Descripción:** TCP Wrappers es una herramienta simple que nos sirve para monitorear y controlar el tráfico que llega por la red. Esta herramienta ha sido utilizada exitosamente en la protección de sistemas y la detección de actividades ilícitas.

Un Wrapper es un programa que es usado para controlar el acceso a un segundo programa. El Wrapper literalmente cubre alrededor de un segundo programa, permitiendo de esta forma un alto nivel de seguridad.

Los Wrappers son de reciente invención dentro de la Seguridad en Sistemas UNIX. Estos programas nacieron de la necesidad de controlar el acceso al sistema sin tener que modificar la estructura del Sistema Operativo.

Sin embargo el uso de los Wrappers se ha acrecentado, y han llegado a formar parte de herramientas de seguridad por diversas razones:

Debido a que la seguridad física es concentrada en un solo programa, además de que son fáciles y simples en la validación de datos.

Debido a que el Wrapper se mantiene como una entidad separada y puede ser actualizada sin necesidad de volver a instalar el paquete.

Debido a que los Wrappers se invocan mediante llamadas al sistema estándar (*exec*) y a que un solo Wrapper puede ser usado para controlar el acceso a una variedad de programas Wrappers.

Un ejemplo sencillo del uso de los Wrappers es limitar la cantidad de información que se puede obtener a través de la red (limitar servicio *finger*)

**Desarrollador:** Esta herramienta fue desarrollada por Wietze Zweitze Venema.

**Disponible en :**

ftp://mezcal.super.unam.mx/pub/security/tools/tcp_wrappers_7.6.tar.gz
ftp://coast.es.purdue.edu/pub/tools/unix/tep_wrappers/tcp_wrappers_7.6.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

2.5.2 SNORT

**Significado del Nombre:** SNORT

**Descripción:** Es un sistema detector de intrusos, con la capacidad de analizar el tráfico y paquetes en una red. Este puede analizar el contenido de los paquetes buscando patrones definidos de ataques. Buffer Overflow, barrido de puertos ataques a CGI, OS fingerprinting entre otras. Snort usa un lenguaje flexible de reglas para determinar el tráfico.

Snort tiene la capacidad de incorporar mecanismos de alerta vía syslog, socket Windows messages.

**Disponible en:** http://www.snort.org

2.5.3 TCPDUMP

**Significado del Nombre:** TCPDUMP

**Descripción:** Este paquete permite desensamblar la cabecera de la información TCP que se encuentra en la red - paquetes TCP/IP - esto lo realiza de forma pasiva ya que no tiene la capacidad de modificar el contenido de estos paquetes.
Dentro del "grupo TCPDUMP" esta es una herramienta de monitoreo de red y adquisición de datos, propiamente es una interfaz de control a una biblioteca - *libpcap* - que tiene la capacidad de interpretar la información TCP.

**Desarrollado por:** Van Jacobson como parte de un proyecto de investigación de *gateway*.

**Disponible en:** http://www.tcpdump.org,
http://ita.ee.lbl.gov/html/software.html

## *2.6 Herramientas IDS*

Llamaremos intrusión a un conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso, analizando esta definición, podemos darnos

cuenta de que una intrusión no tiene por qué consistir en un acceso no autorizado a una máquina: también puede ser una negación de servicio.

A los sistemas utilizados para detectar las intrusiones o los intentos de intrusión se les denomina sistemas de detección de intrusiones (Intrusion Detection Systems, IDS) o, más habitualmente - y aunque no sea la traducción literal - sistemas de detección de intrusos; cualquier mecanismo de seguridad con este propósito puede ser considerado un IDS, pero generalmente sólo se aplica esta denominación a los sistemas automáticos (software o hardware): es decir, aunque un guardia de seguridad que vigila en la puerta de la sala de operaciones pueda considerarse en principio como un sistema de detección de intrusos, lo habitual, es que a la hora de hablar de IDSes no se contemplen estos casos.

Una de las primeras cosas que deberíamos plantearnos a la hora de hablar de IDSes es si realmente necesitamos uno de ellos en nuestro entorno de trabajo; a fin de cuentas, debemos tener ya un sistema de protección perimetral basado en cortafuegos, y por si nuestro *firewall* fallara, cada sistema habría de estar configurado de una manera correcta, de forma que incluso sin cortafuegos cualquier máquina pudiera seguirse considerando relativamente segura.

La respuesta es, sin duda, sí; debemos esperar que en cualquier momento alguien consiga romper la seguridad de nuestro entorno informático, y por tanto hemos de ser capaces de detectar ese problema tan pronto como sea posible.

Ningún sistema informático puede considerarse completamente seguro, pero incluso aunque nadie consiga violar nuestras políticas de seguridad, los sistemas de detección de intrusos se encargarán de mostrarnos todos los intentos de multitud de piratas para penetrar en nuestro entorno, no dejándonos caer en ninguna falsa sensación de seguridad: si somos conscientes de que a diario hay gente que trata de romper nuestros sistemas, no caeremos en la tentación de pensar que nuestras máquinas están seguras porque nadie sabe de su existencia o porque no son interesantes para un pirata.

Los sistemas de detección de intrusos no son precisamente nuevos: el primer trabajo sobre esta materia data de 1980; no obstante, este es uno de los campos más en auge desde hace ya unos años dentro de la seguridad informática. Y no es extraño puesto que la capacidad para detectar y responder ante los intentos de ataque contra nuestros sistemas es realmente muy interesante.

Durante estos veinticinco años, cientos de investigadores de todo el mundo han desarrollado, con mayor o menor éxito, sistemas de detección de todo tipo, desde simples procesadores de *logs* hasta complejos sistemas distribuidos, especialmente vigentes con el auge de las redes de computadoras.

### 2.6.1 Los requisitos de un IDS

Sin importar los sistemas que vigile o su forma de trabajar, cualquier sistema de detección de intrusos ha de cumplir algunas propiedades para poder desarrollar su trabajo correctamente.

1. En primer lugar, y quizás como característica más importante, el IDS ha de ejecutarse continuamente sin nadie que esté obligado a supervisarlo; independientemente de que al detectar un problema se informe a un operador o se lance una respuesta automática, el funcionamiento habitual no debe implicar interacción con un humano. Podemos fijarnos en que esto parece algo evidente: muy pocas empresas estarían dispuestas a contratar a una o varias personas simplemente para analizar *logs* o controlar los patrones del tráfico de una red, hemos de tener presente que los sistemas de detección son mecanismos automatizados que se instalan y configuran, de forma que su trabajo habitual sea transparente a los operadores del entorno informático.

2. Otra propiedad, y también como una característica a tener siempre en cuenta, es la aceptabilidad o grado de aceptación del IDS; al igual que sucedía con cualquier modelo de autenticación, los mecanismos de detección de intrusos han de ser aceptables para las personas que trabajan habitualmente en el entorno. Por ejemplo, no ha de introducir una sobrecarga considerable en el sistema (si un IDS ralentiza demasiado una máquina, simplemente no se utilizará) ni generar una cantidad elevada de falsos positivos (detección de intrusiones que realmente no lo son) o de *logs*, ya que entonces llegará un momento en que nadie se preocupe de comprobar las alertas emitidas por el detector. Por supuesto (y esto puede parecer una tontería, pero es algo que se hace más a menudo de lo que podamos imaginar), si para evitar problemas con las intrusiones simplemente apagamos el equipo o lo desconectamos de la red, tenemos un sistema bastante seguro...pero inaceptable.

3. Una tercera característica a evaluar a la hora de hablar de sistemas de detección de intrusos es la adaptabilidad del mismo a cambios en el entorno de trabajo. Como todos sabemos, ningún sistema informático puede considerarse estático: desde la aplicación más pequeña hasta el propio kernel de Unix, pasando por supuesto por la forma de trabajar de los usuarios (>quién nos asegura que ese engorroso procedimiento desde una `desfasada' línea de órdenes mañana no se realizará desde una aplicación gráfica, que realmente hace el mismo trabajo pero que genera unos patrones completamente diferentes en nuestro sistema?), todo cambia con una periodicidad más o menos elevada. Si nuestros mecanismos de detección de intrusos no son capaces de adaptarse rápidamente a esos cambios, están condenados al fracaso.

4. Todo IDS debe además presentar cierta tolerancia a fallos o capacidad de respuesta ante situaciones inesperadas; un IDS ha de ser capaz de responder siempre adecuadamente ante los cambios bruscos. Podemos contemplar, por ejemplo, un reinicio inesperado de varias máquinas o un intento de engaño hacia el IDS; esto

último es especialmente crítico: sólo hemos de pararnos a pensar que si un atacante consigue modificar el comportamiento del sistema de detección y el propio sistema no se da cuenta de ello, la intrusión nunca será notificada, con los dos graves problemas que eso implica: aparte de la intrusión en sí, la falsa sensación de seguridad que produce un IDS que no genera ninguna alarma es un grave inconveniente de cara a lograr sistemas seguros.

### 2.6.2 Clasificación de los IDS

Generalmente existen dos grandes enfoques a la hora de clasificar a los sistemas de detección de intrusos.

- ✧ en función del tipo de sistemas que vigilan
- ✧ en función de como vigilan los sistemas, sin importar el tipo

Si elegimos la primera de estas aproximaciones tenemos dos grupos de sistemas de detección de intrusos: los que analizan actividades de una única máquina en busca de posibles ataques, y los que lo hacen de una subred (generalmente, de un mismo dominio de colisión) aunque se emplacen en uno sólo de los *hosts* de la misma.

Esta última puntualización es importante: un IDS que detecta actividades sospechosas en una red no tiene porqué ubicarse en todas las máquinas de esa red.

#### *2.6.2.1 En función del tipo de sistemas que vigilan*

**IDSes basados en máquina**

Un sistema de detección de intrusos basado en máquina (*host*-based IDS) es un mecanismo que permite detectar ataques o intrusiones contra la máquina sobre la que se ejecuta.

Tradicionalmente, los modelos de detección basados en máquina han consistido por una parte en la utilización de herramientas automáticas de análisis de *logs* generados por diferentes aplicaciones o por el propio *kernel* del sistema operativo, prestando siempre especial atención a los registros relativos a demonios de red, como un servidor web o el propio inetd, y por otra en el uso de verificadores de integridad de determinados archivos vitales para el sistema, como el de contraseñas; no obstante, desde hace unos años un tercer esquema de detección se está implantando con cierta fuerza, se trata de los sistemas de detección, *honeypots* o tarros de miel.

El análisis de *logs* generados por el sistema (entendiendo como tales no sólo a los relativos al núcleo, sino también a aquellos de aplicaciones nativas de cada Unix, como syslogd) varía entre diferentes clones de Unix por una sencilla razón: cada uno de ellos guarda la información con un cierto formato, y en determinados archivos, aunque todos - o casi todos - sean capaces de registrar los mismos datos, que son aquellos que pueden ser indicativos de un ataque.

La mayor parte de Unix son capaces de registrar con una granularidad lo suficientemente fina casi todas las actividades que se llevan a cabo en el sistema, en especial aquellas que pueden suponer - aunque sea remotamente - una vulneración de su seguridad; sin embargo, el problema radica en que pocos administradores se preocupan de revisar con un mínimo de atención esos *logs*, por lo que muchos ataques contra la máquina, tanto externos como internos, y tanto fallidos como exitosos, pasan finalmente desapercibidos. Aquí es donde entran en juego las herramientas automáticas de análisis, como *swatch* o *logcheck*; a grandes rasgos, realizan la misma actividad que podría ejecutar un *shellscript* convenientemente planificado que incluyera entre sus líneas algunos *grep* de registros sospechosos en los archivos de *log*.

A qué entradas de estos archivos debemos estar atentos? Evidentemente, esto depende de cada sistema y de lo que sea `normal' en él, aunque suelen existir registros que en cualquier máquina denotan una actividad cuanto menos sospechosa. Esto incluye ejecuciones fallidas o exitosas de la orden, peticiones no habituales al servicio SMTP (como vrfy o expn), conexiones a diferentes puertos rechazadas por TCP Wrappers, intentos de acceso remotos como súper usuario, etc; si en la propia máquina tenemos instalado un cortafuegos independiente del corporativo, o cualquier otro software de seguridad - uno que quizás es especialmente recomendable es PortSentry -, también conviene estar atentos a los *logs* generados por los mismos, que habitualmente se registran en los archivos normales de auditoría del sistema (syslog, messages...) y que suelen contener información que con una probabilidad elevada denotan un ataque real.
Por otra parte, la verificación de integridad de archivos se puede realizar a diferentes niveles, cada uno de los cuales ofrece un mayor o menor grado de seguridad. Por ejemplo, un administrador puede programar y planificar un sencillo *shellscript* para que se ejecute periódicamente y compruebe el propietario y el tamaño de ciertos archivos como /etc/passwd o /etc/shadow; evidentemente, este esquema es extremadamente débil, ya que si un usuario se limita a cambiar en el archivo correspondiente su UID de 100 a 000, este modelo no descubriría el ataque a pesar de su gravedad.

Por tanto, parece obvio que se necesita un esquema de detección mucho más robusto, que compruebe aparte de la integridad de la información registrada en el inodo asociado a cada archivo (fecha de última modificación, propietario, grupo propietario...) la integridad de la información contenida en dicho archivo; y esto se consigue muy fácilmente utilizando funciones resumen sobre cada uno de los archivos a monitorizar, funciones capaces de generar un *hash* único para cada contenido de los archivos. De esta forma, cualquier modificación en su contenido generará un resumen diferente, que al ser comparado con el original dará la voz de alarma; esta es la forma de trabajar de Tripwire, el más conocido y utilizado de todos los verificadores de integridad disponibles para entornos Unix.

Sea cual sea nuestro modelo de verificación, en cualquiera de ellos debemos llevar a cabo inicialmente un paso común: generar una base de datos de referencia contra la que posteriormente compararemos la información de cada archivo.

Por ejemplo, si nos limitamos a comprobar el tamaño de ciertos archivos debemos, nada más configurar el sistema, registrar todos los nombres y tamaños de los archivos que deseemos, para después comparar la información que periódicamente registraremos en nuestra máquina con la que hemos almacenado en dicha base de datos; si existen diferencias, podemos encontrarnos ante un indicio de ataque.

Lo mismo sucederá si registramos funciones resumen, debemos generar un *hash* inicial de cada archivo contra el que comparar después la información obtenida en la máquina. Independientemente de los contenidos que deseemos registrar en esa base de datos inicial, siempre hemos de tener presente una cosa: si un pirata consigue modificarla de forma no autorizada, habrá burlado por completo a nuestro sistema de verificación. Así, es vital mantener su integridad; incluso es recomendable utilizar medios de sólo lectura, como un CD-ROM, o incluso unidades extraíbles - discos o disquetes - que habitualmente no estarán disponibles en el sistema, y sólo se utilizarán cuando tengamos que comprobar la integridad de los archivos de la máquina.

Por último, los *honeypots*, básicamente, estos `tarros de miel' son sistemas completos o parte de los mismos (aplicaciones, servicios, sub entornos...) diseñados para recibir ciertos tipos de ataques; cuando sufren uno, los *honeypots* detectan la actividad hostil y aplican una estrategia de respuesta. Dicha estrategia puede consistir desde un simple correo electrónico al responsable de la seguridad de la máquina hasta un bloqueo automático de la dirección atacante; incluso muchos de los sistemas son capaces de simular vulnerabilidades conocidas de forma que el atacante piense que ha tenido éxito y prosiga con su actividad, mientras es monitorizado por el detector de intrusos.

El concepto teórico que está detrás de los tarros de miel es el denominado `conocimiento negativo': proporcionar deliberadamente a un intruso información falsa - pero que él considerará real - de nuestros sistemas, con diferentes fines: desde poder monitorizar durante más tiempo sus actividades, hasta despistarlo, pasando por supuesto por la simple diversión. Evidentemente, para lograr engañar a un pirata medianamente experimentado la simulación de vulnerabilidades ha de ser muy `real', dedicando a tal efecto incluso sistemas completos (denominados 'máquinas de sacrificio'), pero con atacantes de nivel medio o bajo dicho engaño es muchísimo más sencillo: en muchos casos basta simular la existencia de un troyano como BackOrifice mediante FakeBO (http://cvs.linux.hr/fakebo/) para que el pirata determine que realmente estamos infectados e intente utilizar ese camino en su ataque contra nuestro sistema.

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

Algunos autores dividen el grupo de los sistemas de detección de intrusos basados en máquina, en tres subcategorías:

- ***Verificadores de integridad del sistema*** (SIV). Un verificador de integridad no es más que un mecanismo encargado de monitorizar archivos de una máquina en busca de posibles modificaciones no autorizadas, por norma general *backdoors* dejadas por un intruso (por ejemplo, una entrada adicional en el archivo de contraseñas o un /bin/*login* que permite el acceso ante cierto nombre de usuario no registrado). El SIV más conocido es sin duda Tripwire, comentado en este mismo trabajo; la importancia de estos mecanismos es tal que en la actualidad algunos sistemas Unix integran `de serie' verificadores de integridad, como Solaris y su ASET (Automated Security Enhancement Tools).

- ***Monitores de registros*** (LFM). Estos sistemas monitorizan los archivos de log generados por los programas - generalmente demonios de red - de una máquina en busca de patrones que puedan indicar un ataque o una intrusión. Un ejemplo de monitor puede ser *swatch*, pero más habituales que él son los pequeños *shellscripts* que casi todos los administradores realizan para comprobar periódicamente sus archivos de *log* en busca de entradas sospechosas (por ejemplo, conexiones rechazadas en varios puertos provenientes de un determinado *host*, intentos de entrada remota como *root*...).

- ***Sistemas de decepción***. Los sistemas de decepción o tarros de miel (*honeypots*), como Deception Toolkit (DTK), son mecanismos encargados de simular servicios con problemas de seguridad de forma que un pirata piense que realmente el problema se puede aprovechar para acceder a un sistema, cuando realmente se está aprovechando para registrar todas sus actividades. Se trata de un mecanismo útil en muchas ocasiones - por ejemplo, para conseguir `entretener' al atacante mientras se rastrea su conexión - pero que puede resultar peligroso: >qué sucede si el propio sistema de decepción tiene un bug que desconocemos, y el atacante lo aprovecha para acceder realmente a nuestra máquina?

Pero esta división queda algo pobre, ya que cada día se avanza más en la construcción de sistemas de detección de intrusos basados en *host* que no podrían englobarse en ninguna de las subcategorías anteriores.

***IDSes basados en red*.**

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

Los sistemas de detección de intrusos basados en red (network-based IDS) son aquellos capaces de detectar ataques contra diferentes sistemas de una misma red (en concreto, de un mismo dominio de colisión), aunque generalmente se ejecuten en uno solo de los *hosts* de esa red.

Para lograr su objetivo, al menos uno de los interfaces de red de esta máquina sensor trabaja en modo promiscuo, capturando y analizando todas las tramas que pasan por él en busca de patrones indicativos de un ataque.

Cuáles pueden ser estos `patrones identificativos de un ataque' a los que estamos haciendo referencia? Casi cualquiera de los diferentes campos de una trama de red TCP/IP puede tener un valor que, con mayor o menor probabilidad, represente un ataque real; los casos más habituales incluyen:

- **Campos de fragmentación.** Una cabecera IP contiene dieciséis bits reservados a información sobre el nivel de fragmentación del datagrama; de ellos, uno no se utiliza y trece indican el desplazamiento del fragmento que transportan. Los otros dos bits indican o bien que el paquete no ha de ser fragmentado por un *router* intermedio (DF, Don´t Fragment) o bien que el paquete ha sido fragmentado y no es el último que se va a recibir (MF, More Fragments). Valores incorrectos de parámetros de fragmentación de los datagramas se han venido utilizando típicamente para causar importantes negaciones de servicio a los sistemas y, desde hace también un tiempo incluso para obtener la versión del operativo que se ejecuta en un determinado *host*, por ejemplo, qué le sucedería al subsistema de red implantado en el núcleo de una máquina Unix si nunca recibe una trama con el bit MF reseteado, indicando que es el último de un paquete? se quedaría permanentemente esperándola? y si recibe uno que en teoría no está fragmentado pero se le indica que no es el último que va a recibir? cómo respondería el núcleo del operativo en este caso? Como vemos, si en nuestras máquinas observamos ciertas combinaciones de bits relacionados con la fragmentación realmente tenemos motivos para sospechar que alguien trata de atacarnos.

- **Dirección origen y destino.** Las direcciones de la máquina que envía un paquete y la de la que lo va a recibir también son campos interesantes de cara a detectar intrusiones en nuestros sistemas o en nuestra red. No tenemos más que pensar el tráfico proveniente de nuestra DMZ que tenga como destino nuestra red protegida: es muy posible que esos paquetes constituyan un intento de violación de nuestra política de seguridad. Otros ejemplos clásicos son las peticiones originadas desde Internet y que tienen como destino máquinas de nuestra organización que no están ofreciendo servicios directos al exterior, como un servidor de bases de datos cuyo acceso está restringido a sistemas de nuestra red.

- **Puerto origen y destino.** Los puertos origen y destino son un excelente indicativo de actividades sospechosas en nuestra red. Aparte de los intentos de acceso no autorizado a servicios de nuestros sistemas, pueden detectar actividades que

también supondrán a priori violaciones de nuestras políticas de seguridad, como la existencia de troyanos, ciertos tipos de barridos de puertos, o la presencia de servidores no autorizados dentro de nuestra red.

- **Flags TCP.** Uno de los campos de una cabecera TCP contiene seis bits (URG, ACK, PSH, RST, SYN y FIN), cada uno de ellos con una finalidad diferente (por ejemplo, el bit SYN es utilizado para establecer una nueva conexión, mientras que FIN hace justo lo contrario: liberarla). Evidentemente el valor de cada uno de estos bits será 0 o 1, lo cual de forma aislada no suele decir mucho de su emisor; no obstante, ciertas combinaciones de valores suelen ser bastante sospechosas: por ejemplo, una trama con los dos bits de los que hemos hablado - SYN y FIN - activados simultáneamente sería indicativa de una conexión que trata de abrirse y cerrarse al mismo tiempo. Para hacernos una idea de la importancia de estos bits de control, no conviene olvidar que uno de los problemas de seguridad más conocidos de los últimos años sobre plataformas Windows estaba fundamentado básicamente en el manejo de paquetes OOB (Out Of Band), tramas con el bit URG activado.

- **Campo de datos.** Seguramente, el campo de datos de un paquete que circula por la red es donde más probabilidades tenemos de localizar un ataque contra nuestros sistemas; esto es debido a que con toda probabilidad nuestro cortafuegos corporativo detendrá tramas cuya cabecera sea `sospechosa' (por ejemplo, aquellas cuyo origen no esté autorizado a alcanzar su destino o con campos incorrectos), pero rara vez un *firewall* se parará a analizar el contenido de los datos transportados en la trama.

No todo es tan sencillo como comprobar ciertos parámetros de cada paquete que circula por uno de nuestros segmentos.

También es posible y necesario que un detector de intrusos basado en red sea capaz de notificar otros ataques que no se pueden apreciar en una única trama; uno de estos ataques es la presencia de peticiones que, aunque por sí mismas no sean sospechosas, por su repetición en un intervalo de tiempo más o menos pequeño puedan ser indicativas de un ataque.

Ataques difíciles de detectar analizando tramas de forma independiente son las negaciones de servicio distribuidas (DDoS, Distributed Denial of Service), justamente por el gran número de orígenes que el ataque tiene por definición.

Según lo expuesto hasta ahora en este punto, puede parecer que los sistemas de detección de intrusos basados en red funcionan únicamente mediante la detección de patrones; pero no es así.

En principio, un detector de intrusos basado en red puede estar basado en la detección de anomalías, igual que lo puede estar uno basado en máquinas. No obstante, esta aproximación es minoritaria; aunque una intrusión generará probablemente comportamientos anormales susceptibles de ser detectados y eliminados, con demasiada frecuencia estos sistemas no detectarán la intrusión hasta que la misma se encuentre en un estado avanzado.

Este problema hace que la mayor parte de IDSes basados en red que existen actualmente funcionen siguiendo modelos de *detección de usos indebidos*.

Para finalizar el punto dedicado a los sistemas de detección de intrusos basados en red es necesario hablar de las *honeynets* literalmente, “redes de miel”' .

Se trata de un concepto muy parecido al de los *honeypots*, pero extendido ahora a redes completas: redes diseñadas para ser comprometidas, formadas por sistemas reales de todo tipo que, una vez penetrados, permiten capturar y analizar las acciones que está realizando el atacante para así poder aprender más sobre aspectos como sus técnicas o sus objetivos.

Realmente, aunque la idea general sea común, existen dos grandes diferencias de diseño entre un tarro de miel y una *honeynet*: por un lado, esta última evidentemente es una red completa que alberga diferentes entornos de trabajo, no se trata de una única máquina; por otro, los sistemas dentro de esta red son sistemas reales, en el sentido de que no simulan ninguna vulnerabilidad, sino que ejecutan aplicaciones típicas (bases de datos, sistemas de desarrollo...) similares a las que podemos encontrar en cualquier entorno de trabajo `normal'.

El objetivo de una *honeynet* no es la decepción, sino principalmente conocer los movimientos de un pirata en entornos semireales, de forma que aspectos como sus vulnerabilidades o sus configuraciones incorrectas se puedan extrapolar a muchos de los sistemas que cualquier empresa posee en la actualidad; de esta forma podemos prevenir nuevos ataques exitosos contra entornos reales.
En el funcionamiento de una `red de miel' existen dos aspectos fundamentales y especialmente críticos, que son los que introducen la gran cantidad trabajo de administración extra que una *honeynet* implica para cualquier organización.

Por un lado, tenemos el control del flujo de los datos: es vital para nuestra seguridad garantizar que una vez que un sistema dentro de la *honeynet* ha sido penetrado, este no se utilice como plataforma de salto para atacar otras máquinas, ni de nuestra organización ni de cualquier otra; la `red de miel' ha de permanecer perfectamente controlada, y por supuesto aislada del resto de los segmentos de nuestra organización.

En segundo lugar, otro aspecto básico es la captura de datos, la monitorización de las actividades que un atacante lleva a cabo en la *honeynet*.

Nuestro objetivo principal es conocer los movimientos de la comunidad pirata para poder extrapolarlos a sistemas reales, por lo que también es muy importante para el correcto funcionamiento de una *honeynet* una correcta recogida de datos generados por el atacante, ha de ser capturada toda la información posible de cada acción, de una forma poco agresiva (esto es, sin tener que realizar grandes cambios en cada uno de los sistemas) y por supuesto sin que el atacante se entere. Además estos datos recogidos nunca se han de mantener dentro del perímetro de la *honeynet*, ya que si fuera así cualquier pirata podría destruirlos con una probabilidad demasiado elevada.

El concepto de *honeynet* es relativamente nuevo dentro del mundo de la seguridad y, en concreto, de los IDS; a pesar de ello, se trata de una idea muy interesante que presumiblemente va a extenderse de una forma más o menos rápida (no todo lo rápida que nos gustaría, ya que implantar y explotar una *honeynet* no es algo ni trivial, ni mucho menos rápido); cada día más, las herramientas de seguridad no se conforman con detectar problemas conocidos, sino que tratan de anticiparse a nuevas vulnerabilidades que aún no se han publicado pero que pueden estar presentes en multitud de sistemas.

### *2.6.2.2 En función de como vigilan los sistemas*

La segunda gran clasificación de los IDSes se realiza en función de cómo actúan.

Actualmente existen dos grandes técnicas de detección de intrusos: las basadas en la detección de anomalías (anomaly detection) y las basadas en la detección de usos indebidos del sistema (misuse detection). En donde la idea básica de los mismos es la siguiente:

**Detección de anomalías**

La base del funcionamiento de estos sistemas es suponer que una intrusión se puede ver como una anomalía de nuestro sistema, por lo que si fuéramos capaces de establecer un perfil del comportamiento habitual de los sistemas seríamos capaces de detectar las intrusiones por pura estadística: probablemente una intrusión sería una desviación excesiva de la media de nuestro perfil de comportamiento.
Desde que en 1980 James P. Anderson propusiera la detección de anomalías como un método válido para detectar intrusiones en sistemas informáticos, la línea de investigación más activa es la denominada Anomaly Detection IDS, IDSes basados en la detección de anomalías.
La idea es a priori muy interesante: estos modelos de detección conocen lo que es `normal' en nuestra red o nuestras máquinas a lo largo del tiempo, desarrollando y actualizando conjuntos de patrones contra los que comparar los eventos que se producen en los sistemas. Si uno de esos eventos (por ejemplo, una trama procedente de una máquina desconocida) se sale del conjunto de normalidad, automáticamente se cataloga como sospechoso.

Los IDSes basados en detección de anomalías se basan en la premisa de que cualquier ataque o intento de ataque implica un uso anormal de los sistemas-

Pero, >cómo puede un sistema conocer lo que es y lo que no es `normal' en nuestro entorno de trabajo? Para conseguirlo, existen dos grandes aproximaciones o es el sistema el que es capaz de aprenderlo por sí mismo (basándose por ejemplo en el comportamiento de los usuarios, de sus procesos, del tráfico de nuestra red...) o bien se le especifica al sistema dicho comportamiento mediante un conjunto de reglas.

La primera de estas aproximaciones utiliza básicamente métodos estadísticos (medias, varianzas...), aunque también existen modelos en los que se aplican algoritmos de aprendizaje automático;

La segunda aproximación consiste en especificar mediante un conjunto de reglas los perfiles de comportamiento habitual basándose en determinados parámetros de los sistemas (con la dificultad añadida de decidir cuáles de esos parámetros que con mayor precisión delimitan los comportamientos intrusivos).

En el primero de los casos (el basado en métodos estadísticos), el detector observa las actividades de los elementos del sistema, activos - sujetos -, pasivos - objetos - o ambos, y genera para cada uno de ellos un perfil que define su comportamiento; dicho perfil es almacenado en el sistema, y se actualiza con determinada frecuencia envejeciendo la información más antigua y priorizando la más fresca.

El comportamiento del usuario en un determinado momento se guarda temporalmente en otro perfil, denominado `perfil actual' (current profile), y a intervalos regulares se compara con el almacenado previamente en busca de desviaciones que puedan indicar una anomalía.
Se definen diferentes tipos de datos o medidas que pueden ser útiles en la elaboración de estos perfiles:

- **Intensidad de la actividad**. Reflejan el ratio de progreso de la actividad en el sistema, para lo cual recogen datos a intervalos muy pequeños - típicamente entre un minuto y una hora -. Estas medidas detectan ráfagas de comportamiento (por ejemplo, una excesiva generación de peticiones de entrada/salida en un cierto intervalo) que en espacios de tiempo más amplios no podrían ser detectadas.
- **Numéricas.** Se trata de medidas de la actividad cuyo resultado se puede representar en forma de valor numérico, como el número de archivos leídos por cierto usuario en una sesión o la cantidad de veces que ese usuario se ha equivocado al teclear su contraseña de acceso al sistema.

- **Categóricas.** Las medidas categóricas son aquellas cuyo resultado es una categoría individual, y miden la frecuencia relativa o la distribución de una actividad determinada con respecto a otras actividades o categorías; por ejemplo, cual es la relación entre la frecuencia de acceso a un determinado directorio del sistema en comparación con la de acceso a otro. Seguramente la palabra `categoría' no es la más afortunada (por lo menos, no la más clara), ya que bajo este término se pueden englobar tanto a objetos (por ejemplo, archivos) como a eventos (por ejemplo, llamadas a la función crypt()) del sistema; esta definición genérica puede resultar más sencilla si distinguimos entre categorías globales e individuales: en castellano plano, podemos entender las categorías globales como acciones muy genéricas dentro de un entorno, mientras que las categorías individuales serían la particularización para un elemento determinado del sistema. Así, una categoría global puede ser la formada por el conjunto de accesos remotos a la máquina, mientras que una individual sería la formada por los accesos desde una determinada ubicación física.

- **Distribución de registros de auditoría** Esta medida analiza la distribución de las actividades generadas en un pasado reciente basándose en los logs generados por las mismas; dicho análisis se realiza de forma ponderada, teniendo más peso las

actividades más recientes, y es comparado con un perfil de actividades `habituales' previamente almacenado, de forma que permite detectar si en un pasado reciente se han generado eventos inusuales.

La segunda aproximación a la que antes hemos hecho referencia era la consistente en indicar mediante un conjunto de reglas el comportamiento habitual del sistema; suele ser denominada detección de anomalías basada en especificaciones (specification-based anomaly detection), y fue propuesta y desarrollada inicialmente por Calvin Cheuk Wang Ko y otros investigadores de la Universidad de California en Davis, durante la segunda mitad de los noventas.

La idea en la que se sustentan los sistemas de detección de anomalías basados en especificaciones es que se puede describir el comportamiento `deseable' (entendiendo por `deseable' el comportamiento `normal') de cualquier programa cuya seguridad sea crítica; esta descripción se realiza en base a una especificación de seguridad mediante gramáticas, y se considera una violación de la seguridad a las ejecuciones de dichos programas que violen su respectiva especificación.

Para ver más claramente el concepto de la detección de anomalías basada en especificaciones, podemos pensar en la ejecución de un programa que se puede considerar crítico, por ser privilegiado: /bin/passwd. Si conseguimos diferenciar las de esta orden que se pueden considerar `habituales' (por ejemplo, cuando un usuario cambia su contraseña sin problemas, cuando se equivoca, cuando el sistema no le deja cambiarla por los motivos típicos - es débil, hace poco que la cambió...-), podríamos especificar formalmente cada una de estas secuencias de operación. De esta forma, cada vez que un usuario invoque a /bin/passwd, el sistema de detección monitorizará las operaciones que esa llamada genere, y considerará una intrusión a cualquiera que no sea `habitual'.

La idea de los sistemas de detección de intrusos basados en la detección de anomalías es realmente atractiva; no obstante, existen numerosos problemas a los que estos mecanismos tratan de hacer frente.

En primer lugar podemos pararnos a pensar en las dificultades que existen a la hora de `aprender' o simplemente especificar lo habitual: si alguien piensa que por ejemplo obtener un patrón de tráfico `normal' en una red es fácil, se equivoca; quizás establecer un conjunto de procesos habituales en una única máquina resulte menos complicado, pero tampoco se trata de una tarea trivial. Además, conforme aumentan las dimensiones de los sistemas (redes con un gran número de máquinas interconectadas, equipos con miles de usuarios...) estos se hacen cada vez más aleatorios e impredecibles.

Otro gran problema de los sistemas basados en detección de anomalías radica en la política de aprendizaje que éstos sigan; si se trata de esquemas donde el aprendizaje es rápido, un intruso puede generar eventos para conseguir un modelo distorsionado de lo `normal' antes de que el responsable del sistema se percate de ello, de forma que el IDS no llegue a detectar un ataque porque lo considera algo `habitual'.

Si por el contrario el aprendizaje es lento, el IDS considerará cualquier evento que se aleje mínimamente de sus patrones como algo anómalo, generando un gran número de falsos positivos (falsas alarmas), que a la larga harán que los responsables de los sistemas ignoren cualquier información proveniente del IDS, con los evidentes riesgos que esto implica.

***Detección de usos indebidos***.

Dentro de la clasificación de los sistemas de detección de intrusos en base a su forma de actuar, la segunda gran familia de modelos es la formada por los basados en la detección de usos indebidos.

El funcionamiento de los IDSes basados en la detección de usos indebidos presupone que podemos establecer patrones para los diferentes ataques conocidos y algunas de sus variaciones; mientras que la detección de anomalías conoce lo normal (en ocasiones se dice que tienen un `conocimiento positivo', positive knowledge) y detecta lo que no lo es, este esquema se limita a conocer lo anormal para poderlo detectar (conocimiento negativo, negative knowledge).

El esquema de detección de usos indebidos se basa en especificar de una forma más o menos formal las potenciales intrusiones que amenazan a un sistema y simplemente esperar a que alguna de ellas ocurra; para conseguirlo existen cuatro grandes aproximaciones:

- los sistemas expertos
- os análisis de transición entre estados
- las reglas de comparación y emparejamiento de patrones (*pattern matching*)
- la detección basada en modelos.

Los primeros sistemas de detección de usos indebidos, como NIDES, se basaban en los sistemas expertos para realizar su trabajo; en ellos las intrusiones se codifican como reglas de la base de conocimiento del sistema experto, de la forma genérica if-then (if CONDICIÓN *then* ACCIÓN). Cada una de estas reglas puede detectar eventos únicos o secuencias de eventos que denotan una potencial intrusión, y se basan en el análisis - generalmente en tiempo real - de los registros de auditoría proporcionados por cualquier sistema Unix: esta es una de las principales ventajas de los sistemas expertos, el hecho de que el mecanismo de registro dentro de Unix venga proporcionado `de serie', ya que de esta forma el sistema de detección trabaja siempre por encima del espacio del sistema operativo, algo que facilita enormemente su integración dentro de diferentes clones de Unix.

Podemos pensar en un caso real que nos ayude a comprender el funcionamiento de los sistemas expertos a la hora de detectar intrusiones; el típico ejemplo es la detección de un mismo usuario conectado simultáneamente desde dos direcciones diferentes. Cada vez que un usuario se autentica correctamente en el sistema, cualquier Unix genera una línea de registro que se guarda en el archivo de *log* correspondiente; así, al conectar a un servidor Linux Slackware vía SSH, se registra el evento de esta forma:

May 27 03:08:27 rosita sshd[5562]: User toni, coming from anita, authenticated.

Al leer este registro, un sistema experto comprobaría si el usuario toni ya tiene una sesión abierta desde una máquina diferente de anita; si esta condición se cumple (recordemos la forma genérica de las reglas del sistema experto, if-then) se realizaría la acción definida en la regla correspondiente, por norma generar una alarma dirigida al responsable de seguridad del sistema.

La segunda implementación de los sistemas de detección de usos indebidos era la basada en los análisis de transición entre estados bajo este esquema, una intrusión se puede contemplar como una secuencia de eventos que conducen al atacante desde un conjunto de estados inicial a un estado determinado, representando este último una violación consumada de nuestra seguridad. Cada uno de esos estados no es más que una imagen de diferentes parámetros del sistema en un momento determinado, siendo el estado inicial el inmediatamente posterior al inicio de la intrusión, y el último de ellos el resultante de la completitud del ataque; la idea es que si conseguimos identificar los estados intermedios entre ambos, seremos capaces de detener la intrusión antes de que se haga efectiva.

Sin duda el sistema de detección basado en el análisis de transición entre estados más conocido es USTAT, basado en STAT. Este modelo fue desarrollado inicialmente sobre SunOS 4.1.1 (en la actualidad está portado a Solaris 2), y utiliza los registros de auditoría generados por el C2 Basic Security Module de este operativo. En USTAT, estos registros del C2-BSM son transformados a otros de la forma S, A, O, representando cada uno de ellos un evento de la forma `el sujeto S realiza la acción A sobre el objeto O'; a su vez, cada elemento de la terna anterior está formado por diferentes campos que permiten identificar unívocamente el evento representado. El sistema de detección utiliza además una base de datos - realmente, se trata de simples archivos planos - formada principalmente por dos tablas, una donde se almacenan las descripciones de los diferentes estados (SDT, State Description Table) y otra en la que se almacenan las transiciones entre estados que denotan un potencial ataque (SAT, Signature Action Table). Cuando USTAT registre una sucesión determinada de eventos que representen un ataque entrará en juego el motor de decisiones, que emprenderá la acción que se le haya especificado (desde un simple mensaje en consola informando de la situación hasta acciones de respuesta automática capaces de interferir en tiempo real con la intrusión).

La tercera implementación que habíamos comentado era la basada en el uso de reglas de comparación y emparejamiento de patrones o pattern matching; en ella, el detector se basa en la premisa de que el sistema llega a un estado comprometido cuando recibe como entrada el patrón de la intrusión, sin importar el estado en que se encuentre en ese momento. Dicho de otra forma, simplemente especificando patrones que denoten intentos de intrusión el sistema puede ser capaz de detectar los ataques que sufre, sin importar el estado inicial en que esté cuando se produzca dicha detección, lo cual suele representar una ventaja con respecto a otros modelos de los que hemos comentado.

Actualmente muchos de los sistemas de detección de intrusos más conocidos (por poner un ejemplo, podemos citar a SNORT o RealSecure) están basados en el *pattern matching*. Utilizando una base de datos de patrones que denotan ataques, estos programas se dedican

a examinar todo el tráfico que ven en su segmento de red y a comparar ciertas propiedades de cada trama observada con las registradas en su base de datos como potenciales ataques; si alguna de las tramas empareja con un patrón sospechoso, automáticamente se genera una alarma en el registro del sistema. En el punto 18.8.2 hablaremos con más detalle del funcionamiento de SNORT, uno de los sistemas de detección de intrusos basados en red más utilizado en entornos con requisitos de seguridad media.

Por último, tenemos que hablar de los sistemas de detección de intrusos basados en modelos ([GL91]); se trata de una aproximación conceptualmente muy similar a la basada en la transición entre estados, en el sentido que contempla los ataques como un conjunto de estados y objetivos, pero ahora se representa a los mismos como escenarios en lugar de hacerlo como transiciones entre estados. En este caso se combina la detección de usos indebidos con una deducción o un razonamiento que concluye la existencia o inexistencia de una intrusión; para ello, el sistema utiliza una base de datos de escenarios de ataques, cada uno de los cuales está formado por una secuencia de eventos que conforman el ataque. En cada momento existe un subconjunto de esos escenarios, denominado *de escenarios activos*, que representa los ataques que se pueden estar presentando en el entorno; un proceso denominado *anticipador* analiza los registros de auditoría generados por el sistema y obtiene los eventos a verificar en dichos registros para determinar si la intrusión se está o no produciendo (realmente, al ser esos registros dependientes de cada sistema Unix, el anticipador pasa su información a otro proceso denominado *planner*, que los traduce al formato de auditoría utilizado en cada sistema). El anticipador también actualiza constantemente el conjunto de escenarios activos, de manera que este estará siempre formado por los escenarios que representan ataques posibles en un determinado momento y no por la base de datos completa.

Hasta hace poco tiempo no existían sistemas de detección de intrusos basados en modelos funcionando en sistemas reales por lo que era difícil determinar aspectos como su eficiencia a la hora de detectar ataques. En 1998 se presentó NSTAT una extensión de USTAT (del cual hemos hablado antes) a entornos distribuidos y redes de computadores, y en el que se combina el análisis de transición entre estados con la detección de intrusos basada en modelos. A pesar de todo, este modelo es el menos utilizado a la hora de detectar ataques y efectuar respuestas automáticas ante los mismos, especialmente si lo comparamos con los basados en la comparación y emparejamiento de patrones.

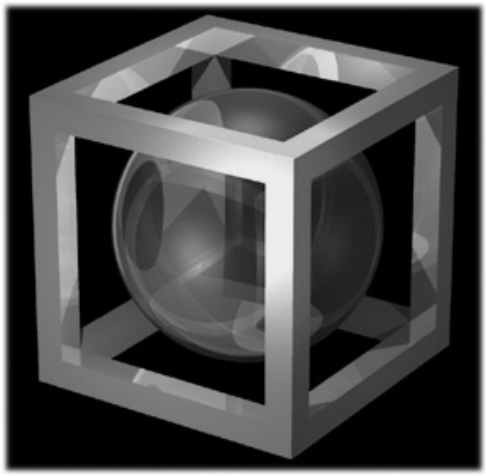
Los IDSes basados en la detección de usos indebidos son en principio más robustos que los basados en la detección de anomalías: al conocer la forma de los ataques, es teóricamente extraño que generen falsos positivos (a no ser que se trate de un evento autorizado pero muy similar al patrón de un ataque); es necesario recalcar el matiz `teóricamente', porque como veremos más adelante, la generación de falsos positivos es un problema a la hora de implantar cualquier sistema de detección. No obstante, en este mismo hecho radica su debilidad: sólo son capaces de detectar lo que conocen, de forma que si alguien nos lanza un ataque desconocido para el IDS éste no nos notificará ningún problema; como ya dijimos, es algo similar a los programas antivirus, y de igual manera que cada cierto tiempo es conveniente (en MS-DOS y derivados) actualizar la versión del antivirus usado, también es

conveniente mantener al día la base de datos de los IDSes basados en detección de usos indebidos. Aún así, seremos vulnerables a nuevos ataques.

Otro grave problema de los IDSes basados en la detección de usos indebidos es la incapacidad para detectar patrones de ataque convenientemente camuflados. Volviendo al ejemplo de los antivirus, pensemos en un antivirus que base su funcionamiento en la búsqueda de cadenas virales: lo que básicamente hará ese programa será buscar cadenas de código hexadecimal pertenecientes a determinados virus en cada uno de los archivos a analizar, de forma que si encuentra alguna de esas cadenas el *software* asumirá que el archivo está contaminado. Y de la misma forma que un virus puede ocultar su presencia simplemente cifrando esas cadenas (por ejemplo de forma semialeatoria utilizando eventos del sistema, como el reloj), un atacante puede evitar al sistema de detección de intrusos sin más que insertar espacios en blanco o rotaciones de *bits* en ciertos patrones del ataque; aunque algunos IDSes son capaces de identificar estas transformaciones en un patrón, otros muchos no lo hacen.

# CAPÍTULO III

# **A**UDITOR DE **N**OCIVIDAD DE **A**RCHIVOS



# CAPÍTULO III. AUDITOR DE NOCIVIDAD DE ARCHIVOS (ANA)


## *3.1 Antecedentes del ANA*

El Auditor de nocividad de archivos "***ANA",*** nació como resultado de una auditoría en informática, aplicada a un sistema en funcionamiento, sacando a la luz las posibles fallas o vulnerabilidades de nuestras barreras de protección.

Para este capítulo se tiene planeado presentar el proceso de auditoría aplicado mediante el uso de ANA, pero Iniciaremos con una descripción breve del concepto "auditoría informática", para establecer el punto de referencia en que trabaja la heramienta.

### *Auditoría en Informática*

La auditoría en informática es la revisión y la evaluación de los controles para sistemas y procedimientos informáticos, utilizados en el procesamiento de la información de manera eficiente y segura, a fin de que por medio de la sistematización de procesos, bien definidos se logre una adecuada toma de decisiones.

La auditoría en informática comprende no sólo la evaluación de los equipos de cómputo, de un sistema o procedimiento específico que se ejecuta, evalúa además de forma completa sistemas de información, a partir de sus entradas, procedimientos, controles, archivos, y seguridad, hasta la obtención de un resultado.

Bajo este concepto las auditorías que se realizan a los distintos sistemas de información, nos preveen como resultado, una serie de valiosas observaciones acerca de nuestras vulnerabilidades; sin embargo el paso realmente importante, es encontrar las medidas que nos permitan no solamente solucionar las fallas detectadas como resultado de la auditoría, sino aquellas posibles medidas de prevención con las cuales evitaremos daños irreversibles.

Por lo tanto la interpretación objetiva de la respuesta de una auditoría, nos proporcionará los *tips* necesarios para planear la manera de que los sistemas a nuestro cargo sean seguros y altamente confiables.

Ahora; un auditor que contenga no solamente la información de nuestro último diagnóstico obtenido y que posea la capacidad de detectar las alteraciones en los principales procesos de nuestro sistema, nos daría la oportunidad de construir una bitácora informática, algo así como la "historia clínica" de nuestros procesos, con la cual, de la misma forma que un médico obtiene un marco de referencia para dar un diagnóstico mucho más preciso, en nuestro caso nos permitiría aplicar el mejor "tratamiento" para aliviar la falla.

En realidad con la cantidad de archivos que tienen que ser manejados para el caso de sistemas tan robustos, existe desde ya mucho tiempo atrás, el manejo de herramientas que son de dominio público, y que han dado lugar a una gran cantidad de variantes para solucionar los problemas que se presentan en el caso de algún ataque. Sin embargo su uso es "aislado", lo que propició para nuestro caso; crear una herramienta que responda a las necesidades específicas de nuestro sistema, lógicamente, basados en las herramientas ya conocidas (de algunas de ellas hemos hablado en el capítulo II), pero una segunda etapa, considera que la nueva herramienta sea capaz de atender otros equipos. Hemos creado un sistema inteligente, provisto de "memoria" a través de la implementación de una base de datos, en la que se almacena el comportamiento del sistema, con la generación de sus firmas en los principales procesos que se auditan.

En efecto, si la comparamos con un IDS, para el momento de correr el proceso de auditoría no sólo detectará la presencia de software nocivo, acción que realizaría un IDS normal, sino que al trabajar como herramienta de seguridad, aplicaría la auditoría automática al sistema, arrojando un resultado claro de la alteración en el código, ***ANA***, al poseer su propia base de datos a la cual nombraremos "repositorio" almacenará a manera de bitácora, el registro de los ataques anteriores, podrá decirnos cual fue el elemento nocivo detectado y la respuesta obtenida

Sin embargo en el caso de ser un ataque registrado por vez primera, manejaríamos los sistemas de espejo para evitar perder la información y recuperar todo hasta el momento previo del ataque. Al funcionar el ***A***uditor de ***N***ocividad de ***A***rchivos, éste sería la pauta para ofrecer, en la red un auditor con memoria en múltiples áreas; dando lugar a un intercambio, rico en análisis de vulnerabilidad y síntomas de ataques por software nocivo.

## *3.2 Desarrollo del ANA*

### Conexión a la base de datos

```
 1-->       printf("\n\n\nDame el nombre del servidor de la base central
            REPOSITORIO]\t-->\t");
 2-->       gets(v_nom_srv);
 3-->       if (v_nom_srv[0]=='\0')
 4-->       strcpy(v_nom_srv,"REPOSITORIO");
 5-->       printf("\n\n\nDame el login de la base de datos a trabajar [COLECTOR]
            \t-->\t");
 6-->       gets(v_login);
 7-->       if (v_login[0]=='\0')    strcpy(v_login,"COLECTOR");
 8-->       printf("\n\n\nDame el password de COLECTOR\t\t\t\t");
 9-->       a_passwd = passwd1;
10-->       a_passwd = getpassphrase(prompt);
11-->       b_ssre = 1;

12-->       DBSETLUSER(login1, v_login);
13-->       dbsetifile("srv_info.ini");
14-->       DBSETLPWD(login1, a_passwd);
15-->       DBSETLENCRYPT (login1,(DBBOOL)1);
16-->       DBSETLAPP(login1, "pro_md5_sis_ANA");
17-->       strcpy(hostname2,hostname1);
18-->       strcat(hostname2,":pro_md5_sis_ANA");
19-->       DBSETLHOST(login1, hostname2);
20-->       dbproc1 = dbopen(login1, v_nom_srv);
21-->       dbproc2 = dbopen(login1, v_nom_srv);
22-->       dbproc3 = dbopen(login1, v_nom_srv);
23-->       if (dbproc1 == NULL || dbproc2 == NULL || dbproc3 == NULL)
24-->       {
25-->       printf("\n\n No se puede establecer la conexion con %s\n", v_nom_srv);
26-->       dbexit();
27-->       exit(1);
28-->       }
29-->       dbuse(dbproc1, "central_d");
30-->       dbuse(dbproc2, "central_d");
31-->       dbuse(dbproc3, "central_d");
```

En la línea 2, el sistema pedirá el nombre del servidor de base de datos en el cual se almacena toda la información referente al conjunto de archivos firmados de un servidor unix auditado.

### Bibliotecas de apoyo

```
1-->    #include <errno.h>
2-->    #include <time.h>
3-->    #include <stdio.h>
4-->    #include <stdlib.h>
5-->    #include <string.h>
6-->    #include <sybfront.h>
7-->    #include <sybdb.h>
8-->    #include <syberror.h>
```

En esta sección inicializaremos las bibliotecas que usará el programa.

**1-->** errno.h: Del inglés ERROr Number

Este archivo de cabecera contiene una serie de definiciones necesarias para el manejo de los errores en las llamadas al sistema estándar. No olvidemos que en los sistemas UNIX se sigue el convenio general de que las llamadas al sistema devuelven el valor -1 si se ha producido algún error en su ejecución. Cuando esto sucede, el sistema almacena en una variable global del proceso, un valor que indica que tipo de error en concreto se ha producido. Esta variable global se llama errno. Pero además, se ofrece una función especial llamada perror que permite mostrar un texto explicativo sobre el error que se ha producido; ambas variables (errno y la función perror) están definidas en el archivo errno.h

### Archivos

**2-->** time.h:
Este archivo incluye las funciones necesarias para acceder a las funciones de tiempo de bajo nivel. De manera concreta, contiene definiciones de tipos y funciones para el manejo de temporizadores.

**3-->** stdio.h : Su nombre proviene de STandarD Input/Output, o Entrada/Salida Estándar.
Este archivo de cabecera incluye funciones para mostrar mensajes en pantalla y acceder a archivos.

**4-->** stdlib.h: Su nombre proviene de STandarD LIBrary, o biblioteca/librería estándar.

Este archivo contiene la definición de varios tipos de funciones como por ejemplo la conversión de cadenas de caracteres a números (atoi,...).

**5-->** string.h: Del inglés STRING, cadena.
Este archivo de cabecera contiene la definición de varias funciones con las que manipular cadenas de caracteres en nuestros programas

**6-->** sybfront.h

Esta biblioteca contiene información referente a el modo de comunicarse con el servidor de base de datos es decir tiene información general del *set* de caracteres y de cómo inicializar el buffer de comunicación.

**7-->** sybdb.h

Este archivo contiene funciones de validación a nivel de sentencias SQL y funciones propias para comunicarse con el servidor de base de datos como son las dbopen, dbsqlexec, etc.

**8-->** syberror.h

En este se contiene los estándares y funciones de manejo de errores generados por el servidor de base de datos, como es el dberrhandle, dbmsghandle

```
 1-->       if (dbinit == FAIL)
 2-->       {
 3-->       printf("\nNo se han inicializado las DB-LIBRARY\n\n");
 4-->       exit(1);
 5-->       }
 6-->       else
 7-->       {
 8-->       dbsetversion(DBVERSION_100);
 9-->       login1 = dblogin();
10-->       printf("\n\n=====================================================\n");
11-->       printf("\n               CONEXION A EL SERVIDOR DE TRABAJO      ");
12-->       printf("\n\n=====================================================\n");
13-->       printf("\n\n\nDame el nombre del servidor al [REPOSITORIO]\t-->\t");
14-->       gets(v_nom_srv);
15-->       if (v_nom_srv[0]=='\0')
16-->       strcpy(v_nom_srv,"REPOSITORIO");
17-->       printf("\n\n\nDame el login de la base de datos  [COLECTOR] \t-->\t");
18-->       gets(v_login);
19-->       if (v_login[0]=='\0')
20-->       strcpy(v_login,"colector");
21-->       printf("\n\n\nDame el password de COLECTOR\t\t\t\t");
22-->       a_passwd = passwd1;
23-->       a_passwd = getpassphrase(prompt);
24-->
25-->       DBSETLUSER(login1, v_login);
26-->       dbsetifile("/export/home/system/ana/srv_info.ini");
27-->       DBSETLPWD(login1, a_passwd);
28-->       DBSETLENCRYPT (login1,(DBBOOL)1);
29-->       DBSETLAPP(login1, "pro_md5_sis_ANA");
30-->       strcpy(hostname2,hostname1);
31-->       strcat(hostname2,":pro_md5_sis_ANA");
32-->       DBSETLHOST(login1, hostname2);
33-->       dbproc1 = dbopen(login1, v_nom_srv);
34-->       dbproc2 = dbopen(login1, v_nom_srv);
35-->       dbproc3 = dbopen(login1, v_nom_srv);
36-->       if (dbproc1 == NULL || dbproc2 == NULL || dbproc3 == NULL)
37-->       {
38-->        printf("\n\n No se puede establecer la conexion con %s\n", v_nom_srv);
39-->         dbexit();
```

```
40-->         exit(1);
41-->       }
42-->       dbuse(dbproc1, "pandora");
43-->       dbuse(dbproc2, "pandora");
44-->       dbuse(dbproc3, "pandora");
45-->       printf("\n\n*****************************************************\n");
46-->       printf("*                                                   *\n");
47-->       printf("*           CONECTADO A %11.11s !!!       *\n", v_nom_srv);
48-->       printf("*                                                    *\n");
49-->       printf("*****************************************************\n");
50-->       printf("\n\n\n\nPresione enter para continuar\n\n\n");
51-->       getchar();
```

**Actuación de las bibliotecas**

En la línea 1 se inicializan las bibliotecas de sybase, que van a ser utilizadas en el programa, que en conjunto con la línea 8, refieren qué versión de bibliotecas se van a utilizar, para este caso es la 10.0.

En la línea 9 dblogin asigna una estructura de login para el uso en el dbopen de la línea 33,34 y 35

En la línea 13 y 14 pide el nombre del servidor que nos va  dar el servicio de base de datos; si el usuario no proporciona esta información y presiona Enter, por default se asignará el servidor de base de datos llamado REPOSITORIO, el cual por medio del archivo mencionado en la línea 26; contendrá la dirección ip y el puerto por el cual se está escuchando el servidor de base de datos.

El contenido del archivo srv_info.ini es el siguiente:

```
REPOSITORIO
      query tcp /dev/tcp \x00021fa484f84e5e0000000000000000


ALTERNO
      query tcp /dev/tcp \x0002371484f84e5e0000000000000000
```

Como podemos ver, se comunicará por medio del protocolo TCP  y la información del servidor y puerto está codificado en hexadecimal, por cuestiones de seguridad, las mismas bibliotecas de sybase se encargan de decodificar estos últimos datos.  Lo importante en este punto es que existe también una línea llamada “ALTERNO” el cual hace referencia a otro servidor de base de datos, para garantizar que cuando el usuario quiera alimentarlo, tenga disponibilidad de la base de datos, de tal manera que en un momento que exista una contingencia con el servidor principal el programa en automático busque al servidor alterno.

En las líneas 17 y 18. Se pide el clave del usuario con el cual se va a conectar el programa, si no se proporciona este datos se asigna un valor por default que para este caso es el de

COLECTOR, por último en las líneas 21 y 22 se pide el password con el cual el programa podrá acceder al servidor de base de datos.

Si no se proporcionan los tres datos; nombre del servidor, clave y passwod del usuario, se da por hecho que el usuario que está operando el programa sólo quiere hacer una auditoría y no un almacenamiento de firmas como se explicará más adelante.
Las líneas 25 a la 29

Se explican brevemente a continuación, a través de las funciones utilizadas y su forma de interactuar en el programa.

| | |
|---|---|
| DBSETLUSER | Pone el username en la estructura de LOGINREC |
| dbsetifile | Especifica el nombre y localización del archivo de interfaces de Sybase. |
| DBSETLPWD | Pone la contraseña de usuario del servidor en la estructura de LOGINREC. |
| DBSETLENCRYPT | Especifica si o no la encriptación de contraseña será usado para acceder al Servidor de SQL. |
| DBSETLAPP | Pone el nombre de la aplicación en la estructura de LOGINREC. |
| DBSETLHOST | Pone el nombre del servidor en la estructura de LOGINREC. |

Las Líneas 30 y 31

Capturan por medio de una función de biblioteca, la dirección ip de la máquina desde la cual se está ejecutando el programa y se asigna a la función DBSETLHOST la cual manda a su vez al servidor de base de datos esta información.

Las líneas 33, 34 y 35.

Gracias a la función dbopen, que crea e inicializa una estructura de DBPROCESS. que es usada como canal de comunicación hacia el servidor, se abre el buffer de comunicación con el servidor de base de datos; cabe mencionar que para este momento la estructura de datos que identifican al usuario ya está llena, por medio de las funciones antes mencionadas, con esto quedan 3 canales (buffers) abiertos hacia el servidor de base de datos llamados: dbproc1, dbproc2, dbproc3.

En la línea 36.

Se verifica que los buffers antes mencionados estén bien inicializados mediante una verificación de que ninguno esté apuntando a una estructura de NULL, si fuera axial con uno sólo que esté en “NULL”, manda un mensaje de error y sale del programa, con un código de retorno de 1.

Por último verifica la comunicación inicializando la aplicación, y principia apuntando los buffers de comunicación a la base de datos, llamada “Pandora”, que guarda la información que más adelante mencionaremos.

Si al llegar a este punto, todo ha salido bien, manda un mensaje en el que especifica que el programa está conectado al servidor “x” y espera que el usuario presione ENTER para continuar.

```
 1-->       main()
 2-->       {
 3-->       dberrhandle(error);
 4-->       dbmsghandle(msg);
 5-->
 6-->       enca1();
 7-->       while(1)
 8-->       {
 9-->       mnu_opc_md5();
10-->       }
11-->       fin_prog();
12-->       }
```

La línea 3

Inicializa la función del usuario para ocuparse de errores de las DB-library. Y en la línea 4 instala una función, para manejar los mensajes del servidor, por ultimo entra en un ciclo, e invoca la función de menú.

Ahora explicaremos el manejo de errores y mensaje de errores del servidor

### Manejo de errores y mensajes de error del servidor

```
 1-->       int error( dbproc, severity, dberr, oserr, dberrstr, oserrstr)
 2-->       DBPROCESS *dbproc;
 3-->       int severity;
 4-->       int dberr;
 5-->       int oserr;
 6-->       char  *dberrstr;
 7-->       char  *oserrstr;
 8-->       {
 9-->       if (dberr==SYBESMSG)
10-->       {
11-->       return(INT_EXIT);
12-->       }
13-->       else
14-->       {
15-->       printf("\n DB-Library  %d  %s ",dberr,dberrstr);
16-->       toma_fecha();
17-->       if (dberr==SYBEPWD)
18-->       printf("%s %s -%s, Login o Passwd incorrecto!! ",fecha,hora,v_nom_srv);
19-->
```

```
20-->       if (dberr==SYBEINTF)
21-->       printf("\n %s %s --  El SRV-%s no es conocido ",fecha,hora,v_nom_srv);
22-->
23-->       if (dberr==SYBESEOF)
24-->       {
25-->       printf("\n\n\n ================================================");
26-->       printf("\n  ___ _ ____   __   __ _  __ _  __ _  __| | ___ | | |");
27-->       printf("\n / __| '__\\ \ \\ / _` |/ _` |/ _` |/ _` |/ _ \\| | |");
28-->       printf("\n \\__ \\ |   \\_) | (_| | (_| | (_| | (_| | (_) |_|_|");
29-->       printf("\n |___/_|    \\_\__,_|\\__, |\\__,_|\\__,_|\\___/(_|_)");
30-->       printf("\n            |_|          |___/                      ");
31-->       printf("\n ================================================");
32-->       return(INT_EXIT);
33-->       }
34-->
35-->       if (dberr == SYBECONN || dberr==SYBETIME )
36-->       printf("\n\n %s %s --  %s, NO!! está activo ",fecha,hora,v_nom_srv);
37-->
38-->       if (oserr != DBNOERR)
39-->       printf("Operating-system error:\n\t%s\n",oserrstr);
40-->       return(INT_CANCEL);
41-->       }
```

---

```
 1-->       int msg(dbproc, msgno, msgstate, severity,  msgtext, srvname, procname,
            line)
 2-->       DBPROCESS   *dbproc;
 3-->       DBINT msgno;
 4-->       int   msgstate;
 5-->       int   severity;
 6-->       char  *msgtext;
 7-->       char  *srvname;
 8-->       char  *procname;
 9-->       DBUSMALLINT line;
10-->
11-->       {
12-->
13-->       if (severity != 0)
14-->       {
15-->       no_errorv=msgno;
16-->       printf("\n\n\tMensaje SQL Server: NO-%d, Estado %d, Severidad %d\n msg
            %s\n Servidor %s Procedimiento %s",msgno, 17--> msgstate, severity,
            msgtext,srvname,procname);
18-->       }
19-->       return(0);
20-->       }
```

---

```
 1-->       printf("\n\n\tMENU DE OPCIONES PARA LA FIRMA DE ARCHIVOS\n\n");
 2-->       printf("\t1.- UN ARCHIVO A FIRMAR\n");
 3-->       printf("\t2.- UN DIRECTORIO A FIRMAR\n");
 4-->       printf("\t3.- TODOS LOS ARCHIVOS A FIRMAR\n");
 5-->       printf("\t4.- REALIZAR LA AUDITORÍA DE LAS FIRMAS DEL SISTEMA\n");
```

```
 6-->       printf("\t5.- REPROCESAR LAS FIRMAS MEDIANTE UN ARCHIVO\n");
 7-->       printf("\t6.- SALIR\n\n");
 8-->       printf("\t    ELIGE OPCION --> ");
 9-->       gets(resp);
```

En la línea 1 imprime el encabezado de menú de opciones para la firma de los archivos, inmediatamente después de la línea 2 a la línea 7 imprime el tipo de opción a elegir.

Para finalizar solicitando al usuario que seleccione alguna de ellas.

```
 1-->       switch(atoi(resp))
 2-->       {
 3-->     case 1:
 4-->       arch_logs();
 5-->       abre_archs();
 6-->       imp_enc_arc();
 7-->       crea_script();
 8-->       system("chmod +x /tmp/veri.sh");
 9-->       printf("\n\nINTRODUCE EL NOMBRE DEL ARCHIVO --> ");
10-->       gets(n_ad);
11-->       bandera=1;
12-->       z=ejecuta_md5(n_ad,bandera);
13-->       procesa_md5(z,0);
14-->       toma_fecha();
15-->       printf("\n\n\n**************************************************\n");
16-->       printf("*    FIN PRO_MD5_SIS_ANA  UN ARCHIVO A FIRMAR)       *\n");
17-->       printf("*    REPORTE: %s ERRORES: %s     *\n",nom_gral_rep,nom_);
18-->       printf("*    %11s               %10s           *\n", fecha,hora);
19-->       printf("**************************************************\n\n\n");
20-->       printf("\n\n\n\t=== FIN DEL PRG PRO_MD5_SIS_ANA ===========\n\n\n");
21-->       getchar();
22-->       fclose(bitacora);
23-->       fclose(reporte);
24-->       fclose(errores);
25-->       fclose(script);
26-->       mnu_opc_md5();
27-->       break;
28-->
29-->     case 2:
30-->       arch_logs();
31-->       abre_archs();
32-->       imp_enc_arc();
33-->       crea_script();
34-->       system("chmod +x /tmp/veri.sh");
35-->       printf("\n\nINTRODUCE EL DIRECTORIO --> ");
36-->       gets(n_ad);
37-->       bandera=2;
38-->       z=ejecuta_md5(n_ad,bandera);
39-->       procesa_md5(z,0);
40-->       toma_fecha();
41-->       printf("\n\n\n**************************************************\n");
42-->       printf("*    FIN DEL PRG PRO_MD5_SIS_ANA (UN DIR FIRMAR)*\n");
43-->       printf("*    REPORTE: %s ERRORES: %s     *\n");
```

```
44-->       printf("*    %11s                  %10s           *\n", fecha,hora);
45-->       printf("*************************************************\n\n\n");
46-->       printf("\n\n\n\t==== FIN DEL PRG PRO_MD5_SIS_ANA==========\n\n\n");
47-->       getchar();
48-->       fclose(bitacora);
49-->       fclose(reporte);
50-->       fclose(errores);
51-->       fclose(script);
52-->       mnu_opc_md5();
53-->       break;
```

De la línea 4 a la 7

Ejecuta las funciones para abrir los archivos de entrada y salida, dichas funciones se declaran más adelante. La línea 4 abre un archivo de tipo .log y la línea 5 abre un archivo .dat, En la línea 9 pide el nombre del archivo y una vez que lo obtiene, ejecuta md5 para generar su firma, y termina imprimiendo un reporte con fecha, hora y errores generados. cierra los archivos y regresa al menú principal.

La línea 29

Corresponde a la opción 2, abre los archivos pide que se introduzca el nombre del directorio y ejecuta md5 para generar su firma, al igual que la primera opción hará reporte del resultado.

```
 1-->       void arch_logs(void)
 2-->       {
 3-->       char nomb_log[100];
 4-->       char fecha_m[7], hora_m[7];
 5-->
 6-->       toma_fecha();
 7-->
 8-->       fecha_m=cambia_fecha();
 9-->       hora_m=cambia_hora();
10-->
11-->       /* Arma el nombre de los archivos de entrada y salida */
12-->       sprintf(nomb_log,"PRO_MD5_SIS_%s%s.LOG",fecha_m,hora_m);
13-->
14-->       if((bitacora=fopen(nomb_log,"a+"))==NULL){
15-->       printf("Error al tratar de crear el archivo de salida: %s\n",nomb_log);
16-->       exit(1);
17-->       }
18-->       }
```

Esta función arma el nombre de los archivos de entrada y salida, con la fecha y hora. En el caso de que el archivo no pueda ser abierto nos enviará un mensaje de error.

```
 1-->       void abre_archs(void)
 2-->       {
 3-->       char nomb_rep[100], nomb_err[100], nomb_script[100];
```

```
 4-->       char fecha_m[7], hora_m[7];
 5-->
 6-->       toma_fecha();
 7-->       fecha_m=cambia_fecha();
 8-->       hora_m=cambia_hora();
 9-->
10-->
11-->       /* Arma el nombre de los archivos de entrada y salida */
12-->       sprintf(nomb_rep,"PRO_MD5_SIS_%s%s.DAT",fecha_m,hora_m);
13-->       sprintf(nomb_err,"PRO_MD5_SIS_%s%s.ERR",fecha_m,hora_m);
14-->       strcpy(nom_gral_rep,nomb_rep);
15-->       strcpy(nom_gral_err,nomb_err);
16-->       sprintf(nomb_script,"/tmp/veri.sh",fecha_m,hora_m);
17-->
18-->       if((reporte=fopen(nomb_rep,"a+"))==NULL){
19-->       printf("Error al tratar de crear el archivo de salida: %s\n",nomb_rep);
20-->       exit(1);
21-->       }
22-->
23-->       if((errores=fopen(nomb_err,"w"))==NULL){
24-->       printf("Error al  crear el archivo de errores: %s\n",nomb_err);
25-->       exit(1);
26-->       }
27-->       if((script=fopen(nomb_script,"w"))==NULL){
28-->       printf("Error al crear el script de proceso: %s\n",nomb_script);
29-->       exit(1);
30-->       }
31-->       }
```

En la línea 12 a la 16

Se arman los nombres de los archivos de entrada y salida junto con la fecha y hora, y en caso de que no puedan abrirse los archivos nos lo hará saber mediante un mensaje de error.

```
 1-->       int procesa_aud(int i,int flag,int vers,char num_op[])
 2-->       {
 3-->       char buffer,nombre[255],cmd[1000],ejec[100],ruta[1000],firma[100],
            fech_arc[20],permisos[11],usuario[20],grupo[20];
 4-->       char  mes[4],dia[3],time[6],nom_tab[260],pat_tab[1000],
 5-->       fir_tab[100],per_tab[15],row[1000],valu[100],
 6-->       shell[100],tipo[50],mirar[ 100];
 7-->       int a,b,c,bb,d,e,j,k,y,x,tr,n,m,fn,fsin,audit,ex_dir,ex_firma,ex_dup;
 8-->       DBCHAR FSI_NOMB[260],FSI_PATH[1000],FSI_HASH[100],FSI_NOMB_SRV[25],
            FSI_PERM_ARC[15],FSI_PROP_ARC[25],FSI_GPO[25];
 9-->       DBCHAR FSI_FECH_ARC[35],FSI_FECH_AUD[35],FSI_NOM_SRV[20],
            FSI_SIST_VER[10];
10-->       DBINT EXISTE=0,DUP=0;
11-->
12-->       FSI_PATH[0]=FSI_FECH_ARC[0]=FSI_FECH_AUD[0]='\0';
13-->
14-->       j=k=y=x=a=b=bb=c=d=tr=n=m=fn=fsin=audit=ex_dir=ex_firma=ex_dup=0;
15-->
16-->       for(j=0;j<i;j++)
17-->       {
```

```
18-->       printf("+");
19-->       fflush(stdout);
20-->       audit=k=y=x=tr=n=m=fn=fsin=0;
21-->       nombre[0]=cmd[0]=ruta[0]=firma[0]=fech_arc[0]='\0';
22-->
23-->       if (j%100==0)
24-->       {
25-->         toma_fecha();
26-->       fprintf(bitacora,"\n %s %s:Llevamos --> %6d Numeros de Archivos
            Procesados\n", fecha,hora,j);
27-->       fflush(bitacora);
28-->       printf("\n %s %s:Llevamos --> %6d Numeros de Archivos Procesados\n",
            fecha,hora,j);
29-->       fflush(stdout);
30-->       }
31-->       if(strstr(alm[j].lin,"("))
32-->       {
33-->         buffer = strtok(alm[j].lin," ()=\n");
34-->         while(buffer != NULL)
35-->           {
36-->             if(strcmp(buffer,"MD5")==0) buffer = strtok(NULL," ()=\n");
37-->             if(strstr(buffer,"/"))
38-->             {
39-->               tr=strlen(buffer);
40-->               strcpy(cmd,buffer);
41-->               for(n=0,m=tr;n=tr/2;n++,m--)
42-->                 {
43-->                   if(cmd[m]=='/')
44-->                   {
45-->                     for(x=m+1;x<tr;x++)
46-->                       {
47-->                         nombre[y]=cmd[x];
48-->                         y++;
49-->                       }
50-->                     fn=m+1;
51-->                     break;
52-->                   }
53-->                 }
54-->               strncpy(ruta,cmd,fn);
55-->               nombre[y]='\0';
56-->               ruta[fn]='\0';
57-->               strcpy(sys[a].l_nomb,nombre);
58-->               strcpy(sys[a].l_path,ruta);
59-->               a++;
60-->             }
70-->             else
71-->             {
72-->               strcpy(firma,buffer);
73-->               strcpy(sys[b].l_hash,firma);
74-->               b++;
75-->             }
76-->             buffer = strtok(NULL," ()=\n");
77-->           }
78-->       }
79-->       else
```

```
80-->       {
81-->         buffer = strtok(alm[j].lin," ");
82-->         while(buffer != NULL)
83-->           {
84-->             if(k==2) strcpy(permisos,buffer);
85-->             if(k==4) strcpy(usuario,buffer);
86-->             if(k==5) strcpy(grupo,buffer);
87-->             if(k==7) strcpy(mes,buffer);
88-->             if(k==8) strcpy(dia,buffer);
89-->             if(k==9) strcpy(time,buffer);
90-->             k++;
91-->             buffer = strtok(NULL," ");
92-->           }
93-->         strcpy(sys[bb].l_perm,permisos);
94-->         strcpy(sys[bb].l_usuario,usuario);
95-->         strcpy(sys[bb].l_grupo,grupo);
96-->         bb++;
97-->       }
98-->     }
99-->   for(c=0;c<a;c++)
101-->     {
103-->      dbfcmd(dbproc1, "select count(*) from fsi where fsi_nomb='%s' and
           fsi_path='%s'", sys[c].l_nomb,sys[c].l_path);
104-->       if(vers==1) dbfcmd(dbproc1, " and fsi_sist_ver='%s'",num_op);
105-->       if (dbsqlexec(dbproc1)==FAIL)
106-->      {
107-->        dbcancel(dbproc1);
108-->        return -1;
109-->      }
110-->       else
111-->      {
112-->        while(dbresults(dbproc1)!=NO_MORE_RESULTS)
113-->          {
114-->            dbbind(dbproc1,1,INTBIND,0,(BYTE *) &EXISTE);
115-->            while(dbnextrow(dbproc1)!=NO_MORE_ROWS)
116-->            {
117-->              ;
118-->            }
119-->          }
120-->      }
121-->
122-->       if(EXISTE>0)
123-->      {
124-->        dbfcmd(dbproc2,   "select   *   from   fsi   where   fsi_nomb='%s'   and
           fsi_path='%s'", sys[c].l_nomb,sys[c].l_path);
125-->        if(vers==1) dbfcmd(dbproc2, " and fsi_sist_ver='%s'",num_op);
126-->        if (dbsqlexec(dbproc2)==FAIL)
127-->          {
128-->            dbcancel(dbproc2);
129-->            return -1;
130-->          }
131-->        else
132-->          {
133-->            while(dbresults(dbproc2)!=NO_MORE_RESULTS)
134-->            {
```

```
135-->              dbbind(dbproc2,1,STRINGBIND,0, FSI_NOMB);
136-->              dbbind(dbproc2,2,STRINGBIND,0, FSI_PATH);
137-->              dbbind(dbproc2,3,STRINGBIND,0, FSI_HASH);
138-->              dbbind(dbproc2,4,STRINGBIND,0, FSI_SIST_VER);
139-->              dbbind(dbproc2,5,STRINGBIND,0, FSI_NOM_SRV);
140-->              dbbind(dbproc2,6,STRINGBIND,0, FSI_PERM_ARC);
141-->              dbbind(dbproc2,7,STRINGBIND,0, FSI_PROP_ARC);
142-->              dbbind(dbproc2,8,STRINGBIND,0, FSI_GPO);
143-->              dbbind(dbproc2,9,STRINGBIND,0, FSI_FECH_ARC);
144-->              dbbind(dbproc2,10,STRINGBIND,0, FSI_FECH_AUD);
145-->              while(dbnextrow(dbproc2)!=NO_MORE_ROWS)
146-->                {
147-->                  if(d==0)
148-->                  {
149-->                    strcpy(nom_tab,FSI_NOMB);
150-->             strcpy(pat_tab,FSI_PATH);
151-->             strcpy(fir_tab,FSI_HASH);
152-->             strcpy(per_tab,FSI_PERM_ARC);
153-->             strcat(pat_tab,nom_tab);
154-->             d++;
155-->             }
156-->       if(strstr(FSI_PERM_ARC,"d"))
157-->       strcpy(tipo,"TIPO:            DIRECTORIO\n");
 158-->      else if(strstr(FSI_PERM_ARC,"D")) strcpy(tipo,"TIPO:
           DOOR\n");
159-->       else if(strstr(FSI_PERM_ARC,"l")) strcpy(tipo,"TIPO: LIGA
           SIMBOLICA\n");
160-->       else if(strstr(FSI_PERM_ARC,"b")) strcpy(tipo,"TIPO: ARCHIVO ESPECIAL
           DE BLOCK\n");
170-->       else if(strstr(FSI_PERM_ARC,"c")) strcpy(tipo,"TIPO: ARCHIVO ESPECIAL
           DE CARACTER\n");
171-->
172-->       else if(strstr(FSI_PERM_ARC,"p")) strcpy(tipo,"TIPO: ARCHIVO ESPECIA
173-->       FIFO(PIPE)\n");
174-->                  else
175-->                  if(strstr(FSI_PERM_ARC,"s")) strcpy(tipo,"TIPO:
                       DIRECCION DE FAMILIAS DE SOCKETS\n");
176-->                  else if(strstr(FSI_PERM_ARC,"-")) strcpy(tipo,"TIPO:
                       ARCHIVO ORDINARIO\n");
177-->                       if(strcmp(sys[c].l_hash,FSI_HASH)!=0 &&
                 strcmp(sys[c].l_perm,FSI_PERM_ARC)!=0 &&
178-->            strcmp(sys[c].l_usuario,FSI_PROP_ARC)!=0 &&
                 strcmp(sys[c].l_grupo,FSI_GPO)!=0)
179-->                  {
180-->                    fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
181-->                    fprintf(reporte,"%s",tipo);
182-->                    fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
183-->                    fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
184-->                    fprintf(reporte,"FIRMA UNIX:      %s\n",sys[c].l_hash);
185-->                    fprintf(reporte,"PERMISOS SYBASE: %s\n",FSI_PERM_ARC);
186-->                    fprintf(reporte,"PERMISOS UNIX:   %s\n",sys[c].l_perm);
187-->                    fprintf(reporte,"PROPIETARIO SYB: %s\n",FSI_PROP_ARC);
188-->                    fprintf(reporte,"PROPIETARIO
                         UNI: %s\n",sys[c].l_usuario);
189-->                    fprintf(reporte,"GRUPO SYBASE:    %s\n",FSI_GPO);
```

```
190-->                    fprintf(reporte,"GRUPO UNIX:      %s\n",sys[c].l_grupo);
191-->                    fprintf(reporte,"SERVIDOR:        %s\n",FSI_NOM_SRV);
192-->                    fprintf(reporte,"VERSION SO:      %s\n",FSI_SIST_VER);
193-->                    fprintf(reporte,"FECHA ARCHIVO:   %s\n",FSI_FECH_ARC);
194-->                    fprintf(reporte,"FECHA AUDITORIA: %s\n",FSI_FECH_AUD);
195-->                    fprintf(reporte,"OBSERVACIONES:   LAS FIRMAS, LOS
                          PERMISOS, LOS USUARIOS Y LOS GRUPOS SON
196-->                    DIFERENTES\n");
197-->                    fprintf(reporte,
198 ->
"================================================================================
==============================\n");
199-->                    fflush(reporte);
201-->                    if(strstr(FSI_PERM_ARC,"d"))
202-->                      {
203-->                        fprintf(errores,"1 %s%s\n",FSI_PATH,FSI_NOMB);
204-->                        fflush(errores);
205-->                      }
206-->
207-->                    else
208-->                      {
209-->                        fprintf(errores,"2 %s%s\n",FSI_PATH,FSI_NOMB);
210-->                        fflush(errores);
211-->                      }
212-->                   }
213-->                   else if(strcmp(sys[c].l_hash,FSI_HASH)!=0 &&
                          strcmp(sys[c].l_perm,FSI_PERM_ARC)!=0 &&
214-->                   strcmp(sys[c].l_usuario,FSI_PROP_ARC)!=0)
215-->                   {
216-->                     fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
217-->                     fprintf(reporte,"%s",tipo);
218-->                     fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
219-->                     fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
220-->                     fprintf(reporte,"FIRMA UNIX:      %s\n",sys[c].l_hash);
221-->                     fprintf(reporte,"PERMISOS SYBASE: %s\n",FSI_PERM_ARC);
222-->                     fprintf(reporte,"PERMISOS UNIX:   %s\n",sys[c].l_perm);
223-->                     fprintf(reporte,"PROPIETARIO SYB: %s\n",FSI_PROP_ARC);
224-->                     fprintf(reporte,"PROPIETARIO UNI:
                          %s\n",sys[c].l_usuario);
225-->                     fprintf(reporte,"GRUPO:           %s\n",FSI_GPO);
226-->                     fprintf(reporte,"SERVIDOR:        %s\n",FSI_NOM_SRV);
227-->                     fprintf(reporte,"VERSION SO:      %s\n",FSI_SIST_VER);
228-->                     fprintf(reporte,"FECHA ARCHIVO:   %s\n",FSI_FECH_ARC);
229-->                     fprintf(reporte,"FECHA AUDITORIA: %s\n",FSI_FECH_AUD);
230-->                     fprintf(reporte,"OBSERVACIONES:   LAS FIRMAS, LOS
                          PERMISOS Y LOS USUARIOS SON DIFERENTES\n");
231-->                     fprintf(reporte,
232-->
"================================================================================
==============================\n");
233-->                     fflush(reporte);
234-->                     if(strstr(FSI_PERM_ARC,"d"))
235-->                       {
236-->                         fprintf(errores,"1 %s%s\n",FSI_PATH,FSI_NOMB);
237-->                         fflush(errores);
```

```
238-->                      }
239-->                    else
240-->                      {
241-->                        fprintf(errores,"2 %s%s\n",FSI_PATH,FSI_NOMB);
242-->                        fflush(errores);
243-->                      }
244-->                  }
245-->                  else
246-->                  if(strcmp(sys[c].l_hash,FSI_HASH)!=0 &&
                       strcmp(sys[c].l_perm,FSI_PERM_ARC)!=0)
247-->                    {
248-->                      fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
249-->                      fprintf(reporte,"%s",tipo);
250-->                      fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
251-->                      fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
252-->                      fprintf(reporte,"FIRMA UNIX:      %s\n",sys[c].l_hash);
253-->                      fprintf(reporte,"PERMISOS SYBASE: %s\n",FSI_PERM_ARC);
254-->                      fprintf(reporte,"PERMISOS UNIX:   %s\n",sys[c].l_perm);
255-->                      fprintf(reporte,"PROPIETARIO:     %s\n",FSI_PROP_ARC);
256-->                      fprintf(reporte,"GRUPO:           %s\n",FSI_GPO);
257-->                      fprintf(reporte,"SERVIDOR:        %s\n",FSI_NOM_SRV);
258-->                      fprintf(reporte,"VERSION SO:      %s\n",FSI_SIST_VER);
259-->                      fprintf(reporte,"FECHA ARCHIVO:   %s\n",FSI_FECH_ARC);
260-->                      fprintf(reporte,"FECHA AUDITORIA: %s\n",FSI_FECH_AUD);
270-->                      fprintf(reporte,"OBSERVACIONES:   LAS FIRMAS Y LOS
                           PERMISOS SON DIFERENTES\n");
271-->                      fprintf(reporte,
272-->
"================================================================================
==============================\n");
273-->                      fflush(reporte);
274-->                      if(strstr(FSI_PERM_ARC,"d"))
275-->                        {
276-->                        fprintf(errores,"1 %s%s\n",FSI_PATH,FSI_NOMB);
277-->                        fflush(errores);
278-->                        }
279-->                      else
280-->                        {
281-->                        fprintf(errores,"2 %s%s\n",FSI_PATH,FSI_NOMB);
282-->                        fflush(errores);
283-->                        }
284-->                    }
285-->                  else
286-->                    if(strcmp(sys[c].l_hash,FSI_HASH)!=0)
287-->                      {
288-->                        fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
289-->                        fprintf(reporte,"%s",tipo);
290-->                        fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
291-->                        fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
292-->                        fprintf(reporte,"FIRMA UNIX:
                             %s\n",sys[c].l_hash);
293-->                        fprintf(reporte,"PERMISOS:
                             %s\n",FSI_PERM_ARC);
294-->                        fprintf(reporte,"PROPIETARIO:
                             %s\n",FSI_PROP_ARC);
```

```
295-->                          fprintf(reporte,"GRUPO:           %s\n",FSI_GPO);
296-->                          fprintf(reporte,"SERVIDOR:        %s\n",FSI_NOM_SRV);
297-->                          fprintf(reporte,"VERSION SO:
                                %s\n",FSI_SIST_VER);
298-->                          fprintf(reporte,"FECHA ARCHIVO:
                                %s\n",FSI_FECH_ARC);
299-->                          fprintf(reporte,"FECHA AUDITORIA:
                                %s\n",FSI_FECH_AUD);
301-->                          fprintf(reporte,"ONSERVACIONES:   LAS FIRMAS SON
                                DIFERENTES\n");
302-->                          fprintf(reporte,
"================================================================================
==============================\n");
304-->                          fflush(reporte);
305-->                          if(strstr(FSI_PERM_ARC,"d"))
306-->                          {
307-->                            fprintf(errores,"1 %s%s\n",FSI_PATH,FSI_NOMB);
308-->                            fflush(errores);
309-->                          }
310-->                          else
311-->                          {
312-->                            fprintf(errores,"2 %s%s\n",FSI_PATH,FSI_NOMB);
313-->                            fflush(errores);
314-->                          }
315-->                        }
316-->                      else if(strcmp(sys[c].l_perm,FSI_PERM_ARC)!=0)
317-->                        {
318-->                          fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
319-->                          fprintf(reporte,"%s",tipo);
320-->                          fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
321-->                          fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
322-->                          fprintf(reporte,"PERMISOS SYBASE:
                                %s\n",FSI_PERM_ARC);
323-->                          fprintf(reporte,"PERMISOS UNIX:
                                %s\n",sys[c].l_perm);
324-->                          fprintf(reporte,"PROPIETARIO:
                                %s\n",FSI_PROP_ARC);
325-->                          fprintf(reporte,"GRUPO:           %s\n",FSI_GPO);
326-->                          fprintf(reporte,"SERVIDOR:        %s\n",FSI_NOM_SRV);
327-->                          fprintf(reporte,"VERSION SO:
                                %s\n",FSI_SIST_VER);
328-->                          fprintf(reporte,"FECHA ARCHIVO:
                                %s\n",FSI_FECH_ARC);
329-->                          fprintf(reporte,"FECHA AUDITORIA:
                                %s\n",FSI_FECH_AUD);
330-->                          fprintf(reporte,"ONSERVACIONES:   LOS PERMISOS SON
                                DIFERENTES\n");
331-->                          fprintf(reporte,
332-->
"================================================================================
==============================\n");
333-->                          fflush(reporte);
334-->                          if(strstr(FSI_PERM_ARC,"d"))
335-->                          {
336-->                            fprintf(errores,"1 %s%s\n",FSI_PATH,FSI_NOMB);
```

```
337-->                          fflush(errores);
338-->                        }
339-->                        else
340-->                        {
341-->                          fprintf(errores,"2 %s%s\n",FSI_PATH,FSI_NOMB);
342-->                          fflush(errores);
343-->                        }
344-->                      }
345-->                    else
346-->                      if(strcmp(sys[c].l_usuario,FSI_PROP_ARC)!=0)
347-->                        {
348-->                        fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
349-->                        fprintf(reporte,"%s",tipo);
350-->                        fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
351-->                        fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
352-->                        fprintf(reporte,"FIRMA UNIX:
                             %s\n",sys[c].l_hash);
353-->                        fprintf(reporte,"PERMISOS:
                             %s\n",FSI_PERM_ARC);
354-->                        fprintf(reporte,"PROPIETARIO SYB:
                             %s\n",FSI_PROP_ARC);
355-->                         fprintf(reporte,"PROPIETARIO UNI:
                             %s\n",sys[c].l_usuario);
356-->                        fprintf(reporte,"GRUPO:   %s\n",FSI_GPO);
357-->                        fprintf(reporte,"SERVIDOR:   %s\n",FSI_NOM_SRV);
358-->                        fprintf(reporte,"VERSION SO:
                             %s\n",FSI_SIST_VER);
359-->                        fprintf(reporte,"FECHA ARCHIVO:
                             %s\n",FSI_FECH_ARC);
360-->                        fprintf(reporte,"FECHA AUDITORIA:
                             %s\n",FSI_FECH_AUD);
370-->                        fprintf(reporte,"ONSERVACIONES:   EL USUARIO ES
                             DISTINTO\n");
371-->                        fprintf(reporte, 72-->
"================================================================================
============================\n");
373-->                        fflush(reporte);
374-->                        if(strstr(FSI_PERM_ARC,"d"))
375-->                          {
376-->                            fprintf(errores,"1 %s%s\n",FSI_PATH,FSI_NOMB);
377-->                            fflush(errores);
378-->                          }
379-->
380-->                        else
381-->                          {
382-->                            fprintf(errores,"2 %s%s\n",FSI_PATH,FSI_NOMB);
383-->                            fflush(errores);
384-->                          }
385-->                        }
386-->                      else
387-->                        if(strcmp(sys[c].l_grupo,FSI_GPO)!=0)
388-->                        {
389-->                          fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
390-->                          fprintf(reporte,"%s",tipo);
391-->                          fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
```

```
392-->                          fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
393-->                          fprintf(reporte,"FIRMA UNIX:
                                %s\n",sys[c].l_hash);
394-->                          fprintf(reporte,"PERMISOS:
                                %s\n",FSI_PERM_ARC);
395-->                            fprintf(reporte,"PROPIETARIO:
                                %s\n",FSI_PROP_ARC);
396-->                          fprintf(reporte,"GRUPO SYBASE:    %s\n",FSI_GPO);
397-->                          fprintf(reporte,"GRUPO UNIX:
                                %s\n",sys[c].l_grupo);
398-->                            fprintf(reporte,"SERVIDOR:
                                %s\n",FSI_NOM_SRV);
399-->                          fprintf(reporte,"VERSION SO:
                                %s\n",FSI_SIST_VER);
401-->                          fprintf(reporte,"FECHA ARCHIVO:
                                %s\n",FSI_FECH_ARC);
402-->                          fprintf(reporte,"FECHA AUDITORIA:
                                %s\n",FSI_FECH_AUD);
403-->                          fprintf(reporte,"OBSERVACIONES:   EL GRUPO ES
                                DIFERENTE\n");
404-->                          fprintf(reporte,
"================================================================================
==============================\n");
406-->                          fflush(reporte);
407-->                          if(strstr(FSI_PERM_ARC,"d"))
408-->                            {
409-->                              fprintf(errores,"1 %s%s\n",FSI_PATH,FSI_NOMB);
410-->                              fflush(errores);
411-->                            }
412-->                          else
413-->                            {
414-->                              fprintf(errores,"2 %s%s\n",FSI_PATH,FSI_NOMB);
415-->                              fflush(errores);
416-->                            }
417-->                        }
418-->                }
419-->            }
420-->          }
421-->      }
422-->       else
423-->      {
424-->        if(strstr(sys[c].l_perm,"d")) strcpy(tipo,"TIPO:
             DIRECTORIO\n");
425-->        else if(strstr(sys[c].l_perm,"D")) strcpy(tipo,"TIPO:
             DOOR\n");
426-->        else if(strstr(sys[c].l_perm,"l")) strcpy(tipo,"TIPO: LIGA
             SIMBOLICA\n");
427-->        else if(strstr(sys[c].l_perm,"b")) strcpy(tipo,"TIPO: ARCHIVO
             ESPECIAL DE BLOCK\n");
428-->        else if(strstr(sys[c].l_perm,"c")) strcpy(tipo,"TIPO: ARCHIVO
             ESPECIAL DE CARACTER\n");
429-->        else if(strstr(sys[c].l_perm,"p")) strcpy(tipo,"TIPO: ARCHIVO
             ESPECIAL FIFO(PIPE)\n");
430-->        else if(strstr(sys[c].l_perm,"s")) strcpy(tipo,"TIPO: DIRECCION DE
             FAMILIAS DE SOCKETS\n");
```

```
431-->        else if(strstr(sys[c].l_perm,"-")) strcpy(tipo,"TIPO: ARCHIVO
              ORDINARIO\n");
432-->        fprintf(reporte,"NOMBRE:         %s\n",sys[c].l_nomb);
433-->        fprintf(reporte,"%s",tipo);
434-->        fprintf(reporte,"RUTA:           %s\n",sys[c].l_path);
435-->        fprintf(reporte,"FIRMA:          %s\n",sys[c].l_hash);
436-->        fprintf(reporte,"OBSERVACIONES:  NO EXISTE EN LA BASE DE
              DATOS\n",sys[c].l_nomb,sys[c].l_path,sys[c].l_hash);
437-->        fprintf(reporte, 38-->
"================================================================================
==============================\n");
439-->        fflush(reporte);
440-->        if(strstr(sys[c].l_perm,"d"))
441-->          {
442-->            fprintf(errores,"1 %s%s\n",sys[c].l_path,sys[c].l_nomb);
443-->            fflush(errores);
444-->          }
445-->
446-->        else
447-->          {
448-->            fprintf(errores,"2 %s%s\n",sys[c].l_path,sys[c].l_nomb);
449-->            fflush(errores);
450-->          }
451-->      }
452-->       dbfcmd(dbproc1, "select count(*) from fsi where fsi_nomb='%s'",
            sys[c].l_nomb);
453-->       if(vers==1) dbfcmd(dbproc1, " and fsi_sist_ver='%s'",num_op);
454-->       if (dbsqlexec(dbproc1)==FAIL)
455-->      {
456-->        dbcancel(dbproc1);
457-->        return -1;
458-->      }
459-->       else
460-->      {
470-->        while(dbresults(dbproc1)!=NO_MORE_RESULTS)
471-->          {
472-->            dbbind(dbproc1,1,INTBIND,0,(BYTE *) &DUP);
473-->            while(dbnextrow(dbproc1)!=NO_MORE_ROWS)
474-->            {
475-->            ;
476-->            }
477-->          }
478-->      }
479-->       if(DUP>1)
480-->      {
481-->        dbfcmd(dbproc2, "select * from fsi where fsi_nomb='%s'",
             sys[c].l_nomb);
482-->        if(vers==1) dbfcmd(dbproc1, " and fsi_sist_ver='%s'",num_op);
483-->        if (dbsqlexec(dbproc2)==FAIL)
484-->          {
485-->            dbcancel(dbproc2);
486-->            return -1;
487-->          }
488-->        else
489-->          {
```

```
490-->            while(dbresults(dbproc2)!=NO_MORE_RESULTS)
491-->            {
492-->              dbbind(dbproc2,1,STRINGBIND,0, FSI_NOMB);
493-->              dbbind(dbproc2,2,STRINGBIND,0, FSI_PATH);
494-->              dbbind(dbproc2,3,STRINGBIND,0, FSI_HASH);
495-->              dbbind(dbproc2,4,STRINGBIND,0, FSI_SIST_VER);
496-->              dbbind(dbproc2,5,STRINGBIND,0, FSI_NOM_SRV);
497-->              dbbind(dbproc2,6,STRINGBIND,0, FSI_PERM_ARC);
498-->              dbbind(dbproc2,7,STRINGBIND,0, FSI_PROP_ARC);
499-->              dbbind(dbproc2,8,STRINGBIND,0, FSI_GPO);
501-->              dbbind(dbproc2,9,STRINGBIND,0, FSI_FECH_ARC);
502-->              dbbind(dbproc2,10,STRINGBIND,0, FSI_FECH_AUD);
503-->              while(dbnextrow(dbproc2)!=NO_MORE_ROWS)
504-->                {
505-->         if(strstr(FSI_PERM_ARC,"d")) strcpy(tipo,"TIPO:
           DIRECTORIO\n");
506-->         else if(strstr(FSI_PERM_ARC,"D")) strcpy(tipo,"TIPO: DOOR\n");
507-->         else if(strstr(FSI_PERM_ARC,"l")) strcpy(tipo,"TIPO: LIGA
           SIMBOLICA\n");
508-->         else if(strstr(FSI_PERM_ARC,"b")) strcpy(tipo,"TIPO: ARCHIVO
           ESPECIAL DE BLOCK\n");
509-->         else if(strstr(FSI_PERM_ARC,"c")) strcpy(tipo,"TIPO: ARCHIVO
           ESPECIAL DE CARACTER\n");
510-->         else if(strstr(FSI_PERM_ARC,"p")) strcpy(tipo,"TIPO: ARCHIVO
           ESPECIAL FIFO(PIPE)\n");
511-->         else if(strstr(FSI_PERM_ARC,"s")) strcpy(tipo,"TIPO: DIRECCION
           DE FAMILIAS DE SOCKETS\n");
512-->         else if(strstr(FSI_PERM_ARC,"-")) strcpy(tipo,"TIPO: ARCHIVO
           ORDINARIO\n");
513-->
514-->         fprintf(reporte,
515-->
"*****************************************************************************
**************************\n");

516-->                   fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
517-->                   fprintf(reporte,"%s",tipo);
518-->                   fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
519-->                   fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
520-->                   fprintf(reporte,"PERMISOS SYBASE: %s\n",FSI_PERM_ARC);
521-->                   fprintf(reporte,"PROPIETARIO SYB: %s\n",FSI_PROP_ARC);
522-->                   fprintf(reporte,"GRUPO SYBASE:    %s\n",FSI_GPO);
523-->                   fprintf(reporte,"SERVIDOR:        %s\n",FSI_NOM_SRV);
524-->                   fprintf(reporte,"VERSION SO:      %s\n",FSI_SIST_VER);
525-->                   fprintf(reporte,"FECHA ARCHIVO:   %s\n",FSI_FECH_ARC);
526-->                   fprintf(reporte,"FECHA AUDITORIA: %s\n",FSI_FECH_AUD);
527-->                   fprintf(reporte,"OBSERVACIONES:   EL ARCHIVO ESTA
                        DUPLICADO\n");
528-->                 }
529-->              }
530-->            }
531-->
532-->                   printf(reporte,
533-->
```

```
"*****************************************************************************
*****************************\n");

534-->        fflush(reporte);
535-->       }
536-->      }
537-->
538-->    dbfcmd(dbproc3,"select * from fsi where fsi_path like \"%s%s%%\" and
          fsi_perm_arc not like 39--> \"d%%\"",sys[0].l_path,sys[0].l_nomb);
540-->    if(vers==1) dbfcmd(dbproc1, " and fsi_sist_ver='%s'",num_op);
541-->    if (dbsqlexec(dbproc3)==FAIL)
542-->      {
543-->        dbcancel(dbproc3);
544-->        return -1;
545-->      }
546-->    else
547-->      {
548-->        while(dbresults(dbproc3)!=NO_MORE_RESULTS)
549-->       {
550-->        dbbind(dbproc3,1,STRINGBIND,0, FSI_NOMB);
551-->        dbbind(dbproc3,2,STRINGBIND,0, FSI_PATH);
552-->        dbbind(dbproc3,3,STRINGBIND,0, FSI_HASH);
553-->        dbbind(dbproc3,4,STRINGBIND,0, FSI_SIST_VER);
554-->        dbbind(dbproc3,5,STRINGBIND,0, FSI_NOM_SRV);
555-->        dbbind(dbproc3,6,STRINGBIND,0, FSI_PERM_ARC);
556-->        dbbind(dbproc3,7,STRINGBIND,0, FSI_PROP_ARC);
557-->        dbbind(dbproc3,8,STRINGBIND,0, FSI_GPO);
558-->        dbbind(dbproc3,9,STRINGBIND,0, FSI_FECH_ARC);
559-->        dbbind(dbproc3,10,STRINGBIND,0, FSI_FECH_AUD);
560-->        while(dbnextrow(dbproc3)!=NO_MORE_ROWS)
570-->          {
571-->            sprintf(shell, "/tmp/veri.sh %s%s",FSI_PATH,FSI_NOMB);
572-->            system(shell);
573-->            if((checa_estd=fopen("/tmp/arc.era","r"))==NULL){
574-->            printf("Error al tratar de leer el archivo de salida:
                 arc.era\n");
575-->            exit(1);
576-->            }
577-->
578-->             while (!feof(checa_estd) && fgets(valu,sizeof(valu),checa_estd))
                 {
579-->            if(atoi(valu)==1) {
580-->            if(strstr(FSI_PERM_ARC,"d")) strcpy(tipo,"TIPO:
                 DIRECTORIO\n");
581-->              else if(strstr(FSI_PERM_ARC,"D")) strcpy(tipo,"TIPO:
                   DOOR\n");
582-->              else if(strstr(FSI_PERM_ARC,"l")) strcpy(tipo,"TIPO:
                   LIGA SIMBOLICA\n");
583-->              else if(strstr(FSI_PERM_ARC,"b")) strcpy(tipo,"TIPO:
                   ARCHIVO ESPECIAL DE BLOCK\n");
584-->              else if(strstr(FSI_PERM_ARC,"c")) strcpy(tipo,"TIPO:
                   ARCHIVO ESPECIAL DE CARACTER\n");
585-->              else if(strstr(FSI_PERM_ARC,"p")) strcpy(tipo,"TIPO:
                   ARCHIVO ESPECIAL FIFO(PIPE)\n");
```

```
586-->              else if(strstr(FSI_PERM_ARC,"s")) strcpy(tipo,"TIPO:
                    DIRECCION DE FAMILIAS DE SOCKETS\n");
587-->              else if(strstr(FSI_PERM_ARC,"-")) strcpy(tipo,"TIPO:
                    ARCHIVO ORDINARIO\n");
588-->              fprintf(reporte,"NOMBRE:          %s\n",FSI_NOMB);
589-->              fprintf(reporte,"%s",tipo);
590-->              fprintf(reporte,"RUTA:            %s\n",FSI_PATH);
591-->              fprintf(reporte,"FIRMA SYBASE:    %s\n",FSI_HASH);
592-->               //           fprintf(reporte,"FIRMA UNIX:
                    %s\n",sys[c].l_hash);
593-->              fprintf(reporte,"PERMISOS:        %s\n",FSI_PERM_ARC);
594-->              fprintf(reporte,"PROPIETARIO:     %s\n",FSI_PROP_ARC);
595-->              fprintf(reporte,"GRUPO SYBASE:    %s\n",FSI_GPO);
596-->              //            fprintf(reporte,"GRUPO UNIX:
                    %s\n",sys[c].l_grupo);
597-->              fprintf(reporte,"SERVIDOR:        %s\n",FSI_NOM_SRV);
598-->              fprintf(reporte,"VERSION SO:      %s\n",FSI_SIST_VER);
599-->              fprintf(reporte,"FECHA ARCHIVO:   %s\n",FSI_FECH_ARC);
601-->              fprintf(reporte,"FECHA AUDITORIA: %s\n",FSI_FECH_AUD);
602-->              fprintf(reporte,"OBSERVACIONES:   EXISTE EN LA BASE DE DATOS
                    PERO NO EN EL DIRECTORIO\n");
603-->              fprintf(reporte,"                 POSIBLES CAUSAS: EL ARCHIVO
                    FUE ELIMINADO O SE LE CAMBIO DE NOMBRE\n");
604-->              fprintf(reporte,
"================================================================================
============================\n");
606-->              fflush(reporte);
607-->              fprintf(errores,"3 %s%s\n",FSI_PATH,FSI_NOMB);
608-->              fflush(errores);
609-->
610-->
611-->            }
612-->
613-->            }
614-->            fclose(checa_estd);
615-->
616-->
617-->
618-->
619-->
620-->          }
621-->       }
622-->     }
```

En la línea 103.

Mediante la función dbfcmd se envía un comando, para realizar un select a la tabla fsi, para generar resultados dependiendo de los parámetros de entrada.

En la línea 105

Se detectan errores de sintaxis.

En la línea 114
Asociamos la columna 1 del resultado del buffer de comandos con la variable EXISTE.

En la línea 122
Se valida el resultado de la variable, dependiendo del resultado realiza una nueva consulta.

De las líneas 126 a la 129
Se hace una revisión para que no existan errores.

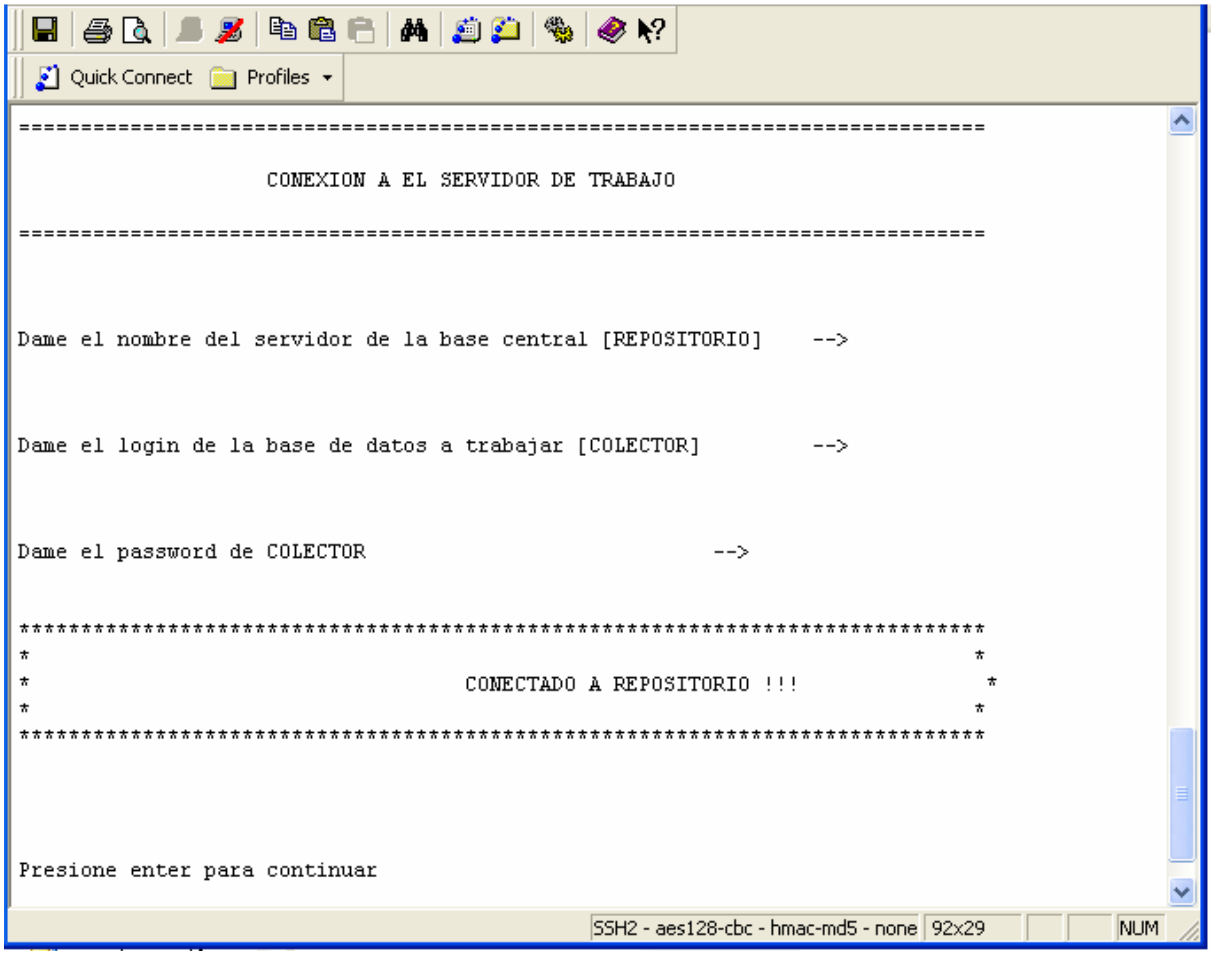
De las líneas 134 a la 144
Se asigna el resultado de la consulta a unas variables.

De las líneas 149 a la 152
Se copian los resultados del select a variables.

La línea 153
Concatena cadenas.

A partir de la 179
Se generan reportes dependiendo del parámetro en la línea 452.

## *3.3 Operación del ANA*

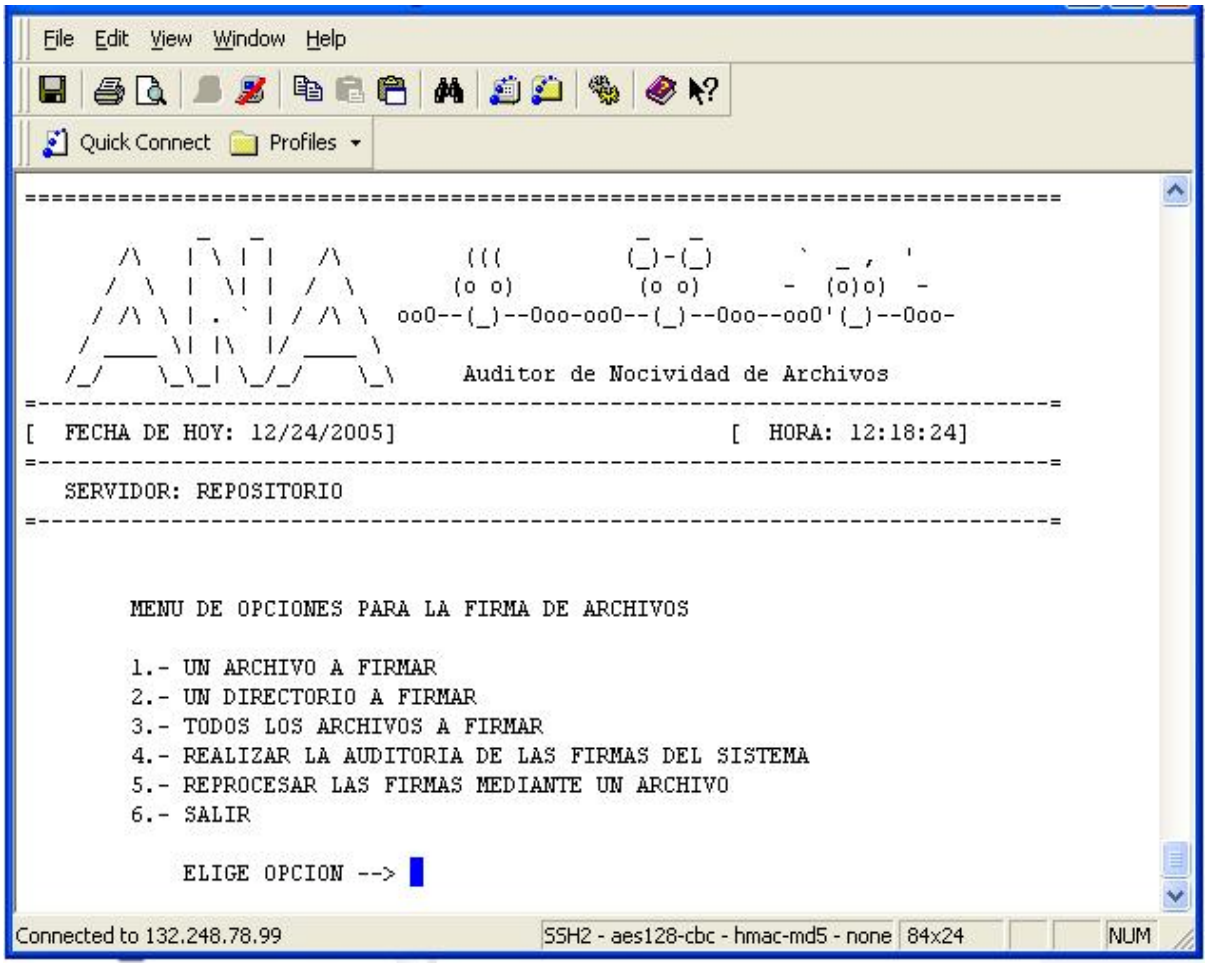



En esta pantalla se muestra el acceso al programa ANA que como se había descrito anteriormente, pide el nombre del servidor de base de datos con el cual vamos a trabajar, como podemos observar, el programa ya tiene el valor definido que es el servidor llamado REPORSITORIO, el cual es el servidor que debe estar siempre activo, alternativamente se dará la posibilidad de entrar a otro servidor.

El siguiente dato que nos solicita es el login con el cual vamos a entrar al servidor de base de datos y al igual que en el anterior, si no se proporciona ningún valor hay uno por default llamado COLECTOR; con el que podremos ejecutar todas las opciones que se muestran más adelante.

Por último nos solicitará el password, el cual por el momento es nulo, es decir, no requiere que se le dé algún valor a este campo de captura, así que con presionar la tecla ENTER será más que suficiente para acceder al uso de la aplicación.

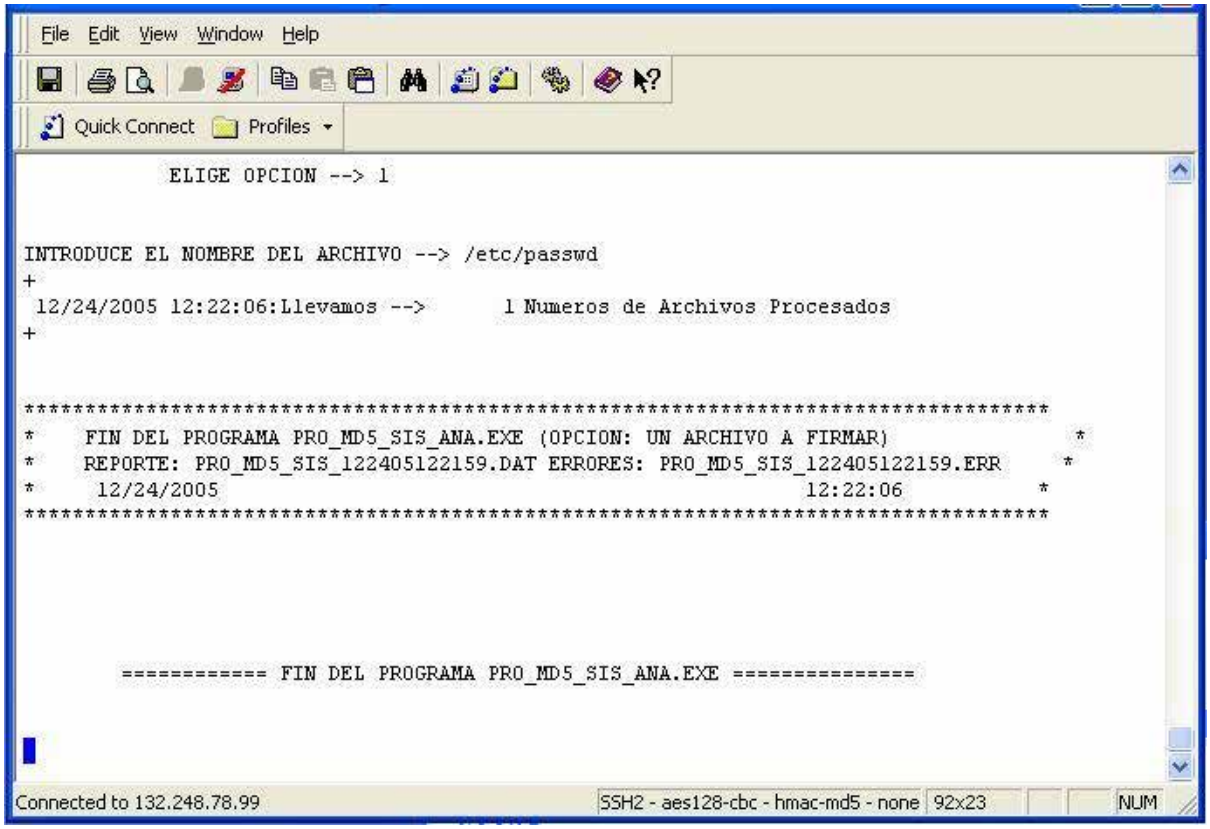
Posteriormente y si no envía ningún mensaje de error estaremos conectados al sistema y sólo estará esperando de nuevo un ENTER para iniciar el trabajo.




En esta imagen se presenta el menú de opciones del programa; el que compone de 6 posibles opciones, intentamos que sea un sistema amigable para el operador y por ello sólo se tiene que ingresar la información que se va solicitando para el proceso.

Además de presentar la fecha y hora en la cual se está ejecutando, se especificará que servidor de base de datos es con que estamos operando.

Ahora realizaremos una explicación abordando opción por opción.



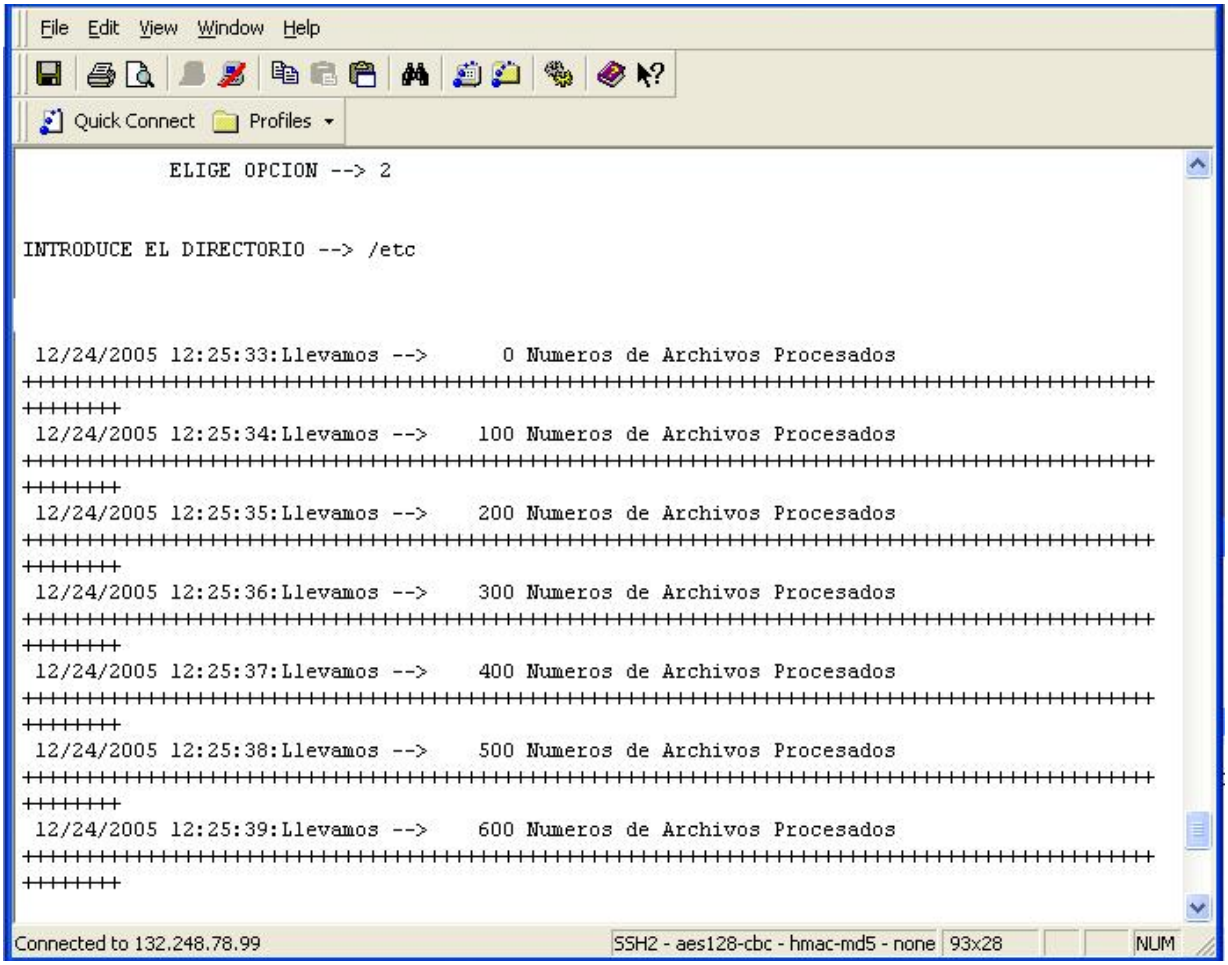

***La opción 1***

Nos permite firmar un sólo archivo, en este momento hay que mencionar que se captura de manera automática la información general del archivo como es: dueño, permisos, tamaño del archivo, nombre, grupo al que pertenece etc. y se genera el hash md5 del mismo archivo, toda esta información es almacenada en las tablas definidas en el servidor de base de datos, incluyendo la versión del sistema operativo y la ruta en la que se encuentra físicamente el archivo, podríamos decir que está tomándole una foto digital para su posterior reconocimiento.

El programa muestra el avance en número de archivos y despliega caracteres “+” por cada archivo que se firma.

Posteriormente se generarán dos archivos más, el primero contiene un diagnóstico del momento de la carga, por lo que si el archivo ya exigía el programa avisa que ya fue capturado en algún momento, en caso contrario dará un informe de cuales fueron los datos que se guardaron en la base de datos, en su anterior corrida.

También se generará un reporte de posibles errores que se podrían haber producido al momento de la carga, (un archivo con extensión .ERR), en el cual dará un breve recuento de lo ocurrido al momento de cargar la información.



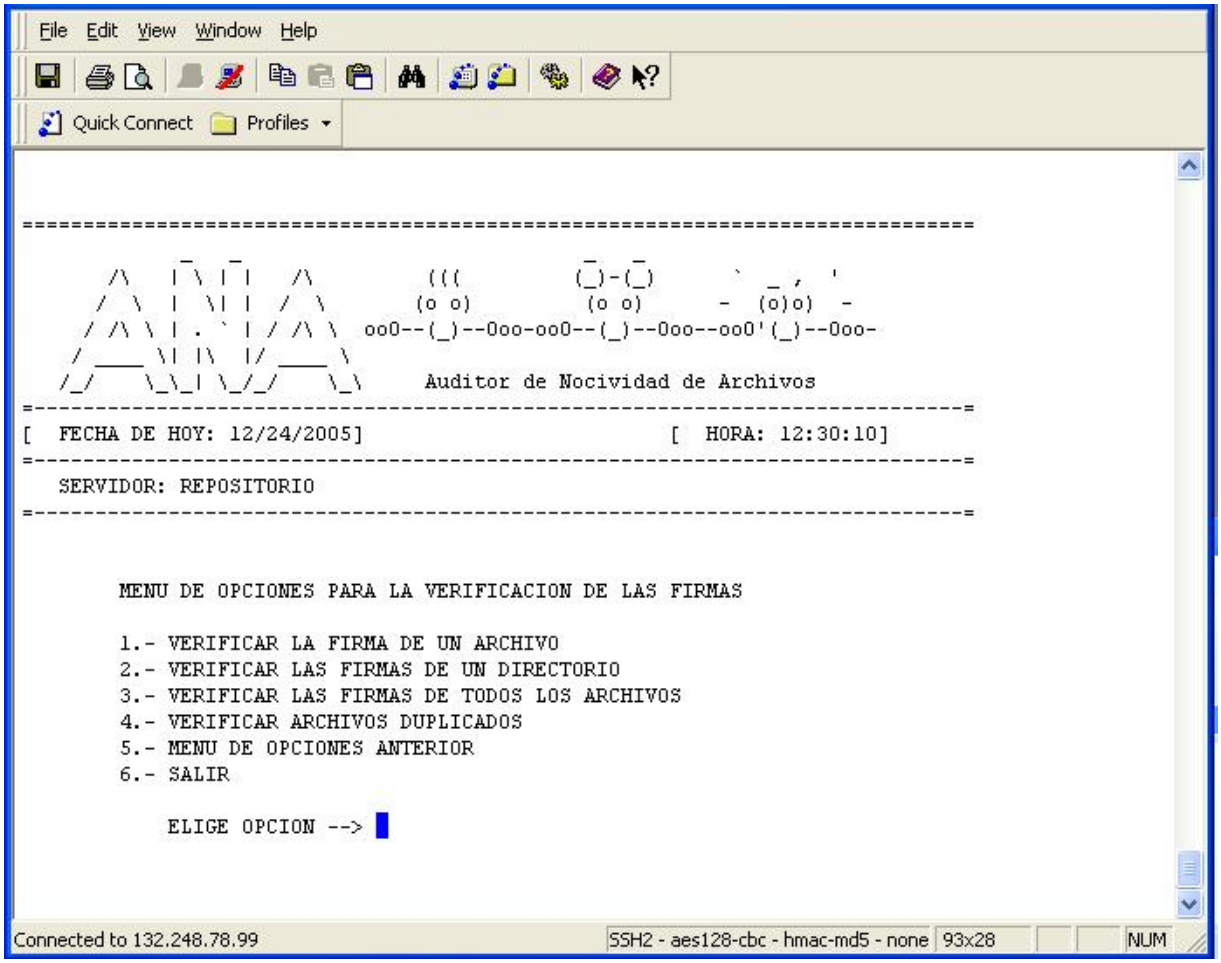

***La opción 2***

Se muestra en la imagen, para este caso se captura el nombre de un directorio y de igual manera que en la opción 1 por medio de signos "+" muestra el avance en carga del programa al servidor de base de datos; y el programa hace de manera interactiva el rastreo de los archivos, esto significa que si el directorio tiene subdirectorio y éstos a su vez de igual forma contienen subdirectorios, el programa va recorriendo todas las ramas del árbol hasta llegar a los archivos finales. Y al igual que la opción 1 genera dos archivos.

Los nombres de los archivos son los mismos pero cambian las extensiones, están compuestos por el prefijo PRO_MD5_SIS_ posteriormente pone el mes, día y año en el cual fue ejecutado el programa e incluye la hora con minutos y segundos en el se ejecutó.
.

***La opción 3***

Realiza el mismo proceso que la opción 2, pero para toda la máquina unix y arranca desde root (/); guarda la información en el servidor de todos los archivos que existan en el momento de la ejecución.




La imagen arriba muestra el resultado de la parte de auditoría y verificación de firmas.

Podemos observar que ese submenú está compuesto de 6 posibles acciones, por ejemplo verificar la firma de un sólo archivo o directorio, la de todos los archivos (que nos permite

auditar TODA la máquina) y la opción 4 que nos permite realizar una tarea que analizaremos más adelante.

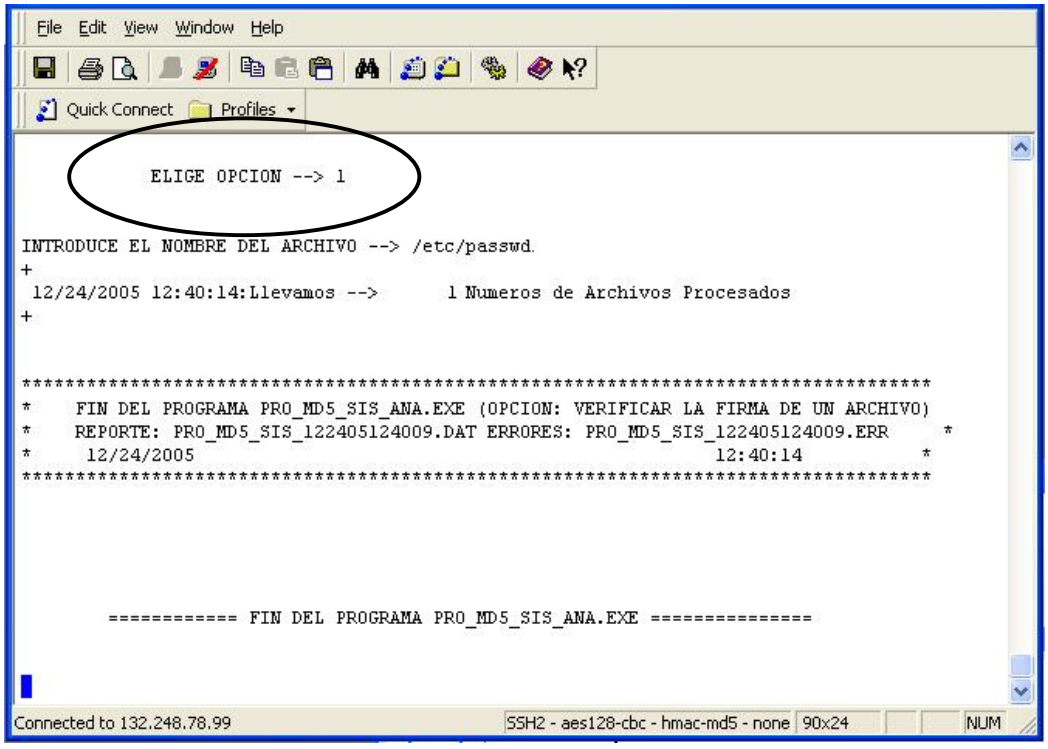
***La opción 5***

Regresa al menú anterior

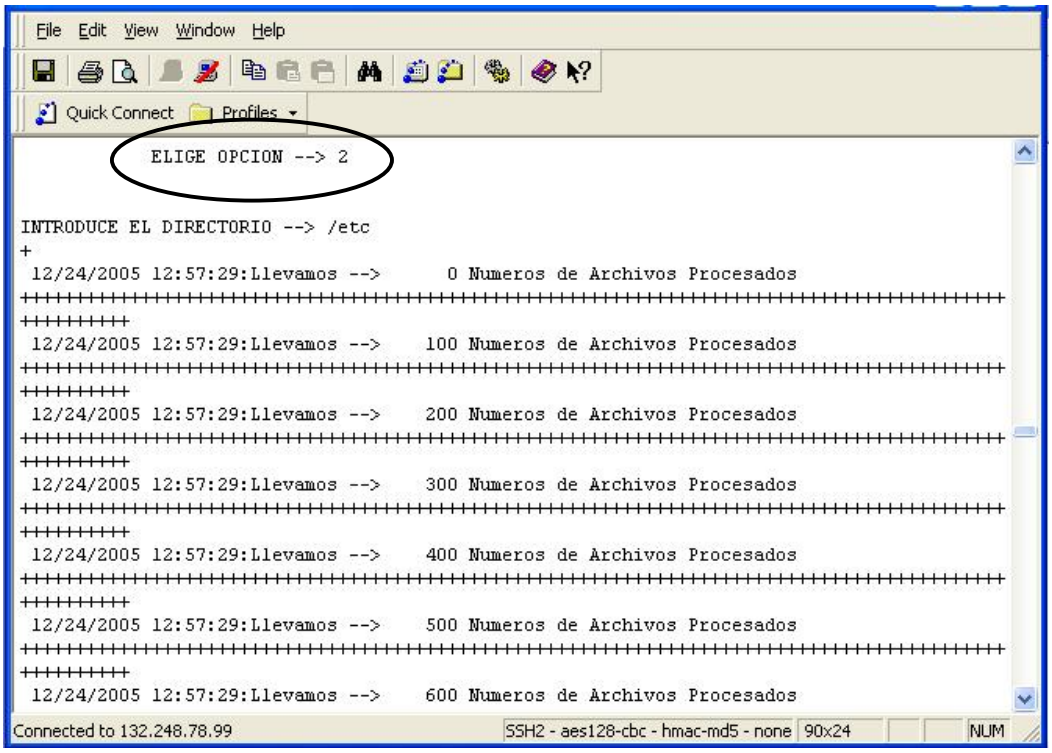
***La opción6***

Abandona el programa

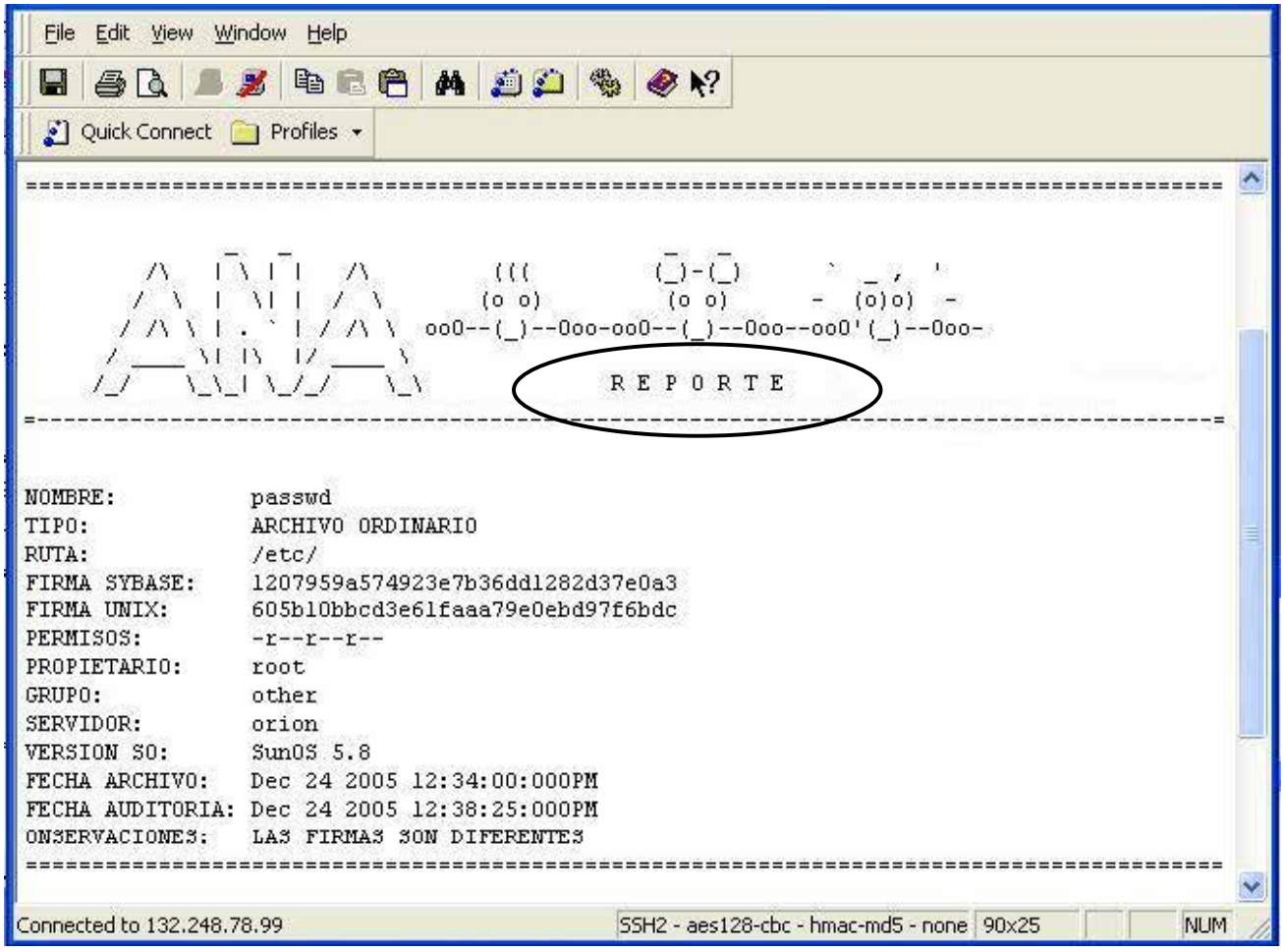
Las tres primeras opciones son muy similares en su ejecución que los anteriores vistos de generar la firma así que sólo se mencionará y se mostraran sus respectivas ejecuciones.



Verificación de las firmas de un solo archivo



Resultado de la ejecución de verificación de firmas de un directorio.



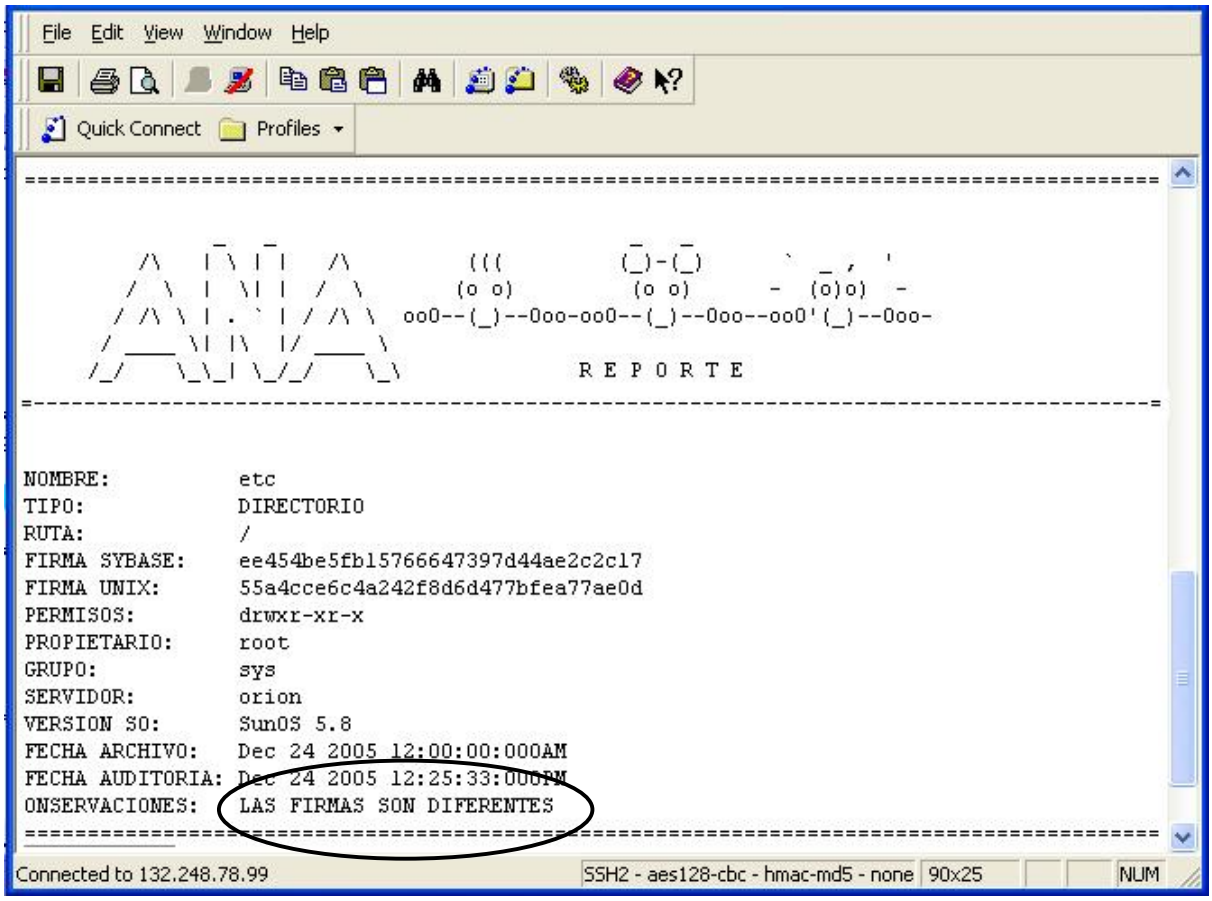

En este cuadro se observa la salida de la primera auditoría, ofrece la información general como es el nombre del archivo, el tipo de archivo considerados todos los tipos que existen en unix system5 tanto de BLOCK, como CARÁCTER y SOCKETS.

También indica la ruta en la que está ubicado físicamente el archivo, de manera que si fuera una liga simbólica, este campo mostrará la referencia física.

Posteriormente muestra la firma (hash md5) que está almacenada en la base de datos, y en el siguiente renglón la firma que se está generando en ese momento, que es la que realmente genera el error; los permisos que tenían al momento de ejecutar la auditoría el propietario o dueño del archivo, el grupo, el servidor en el cual se encuentra el archivo, la

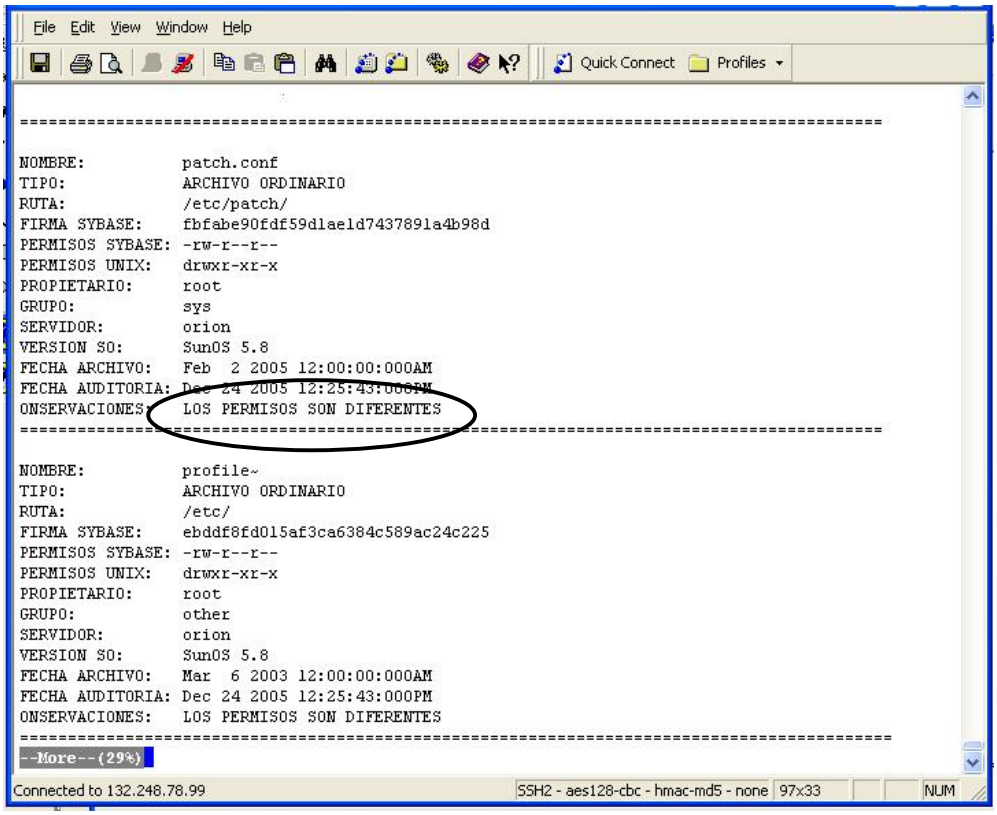
versión del sistema unix, la fecha hora minuto y segundo en la cual se hizo la carga del archivo, y la fecha hora minuto y segundo en la cual se realizo la auditoría.

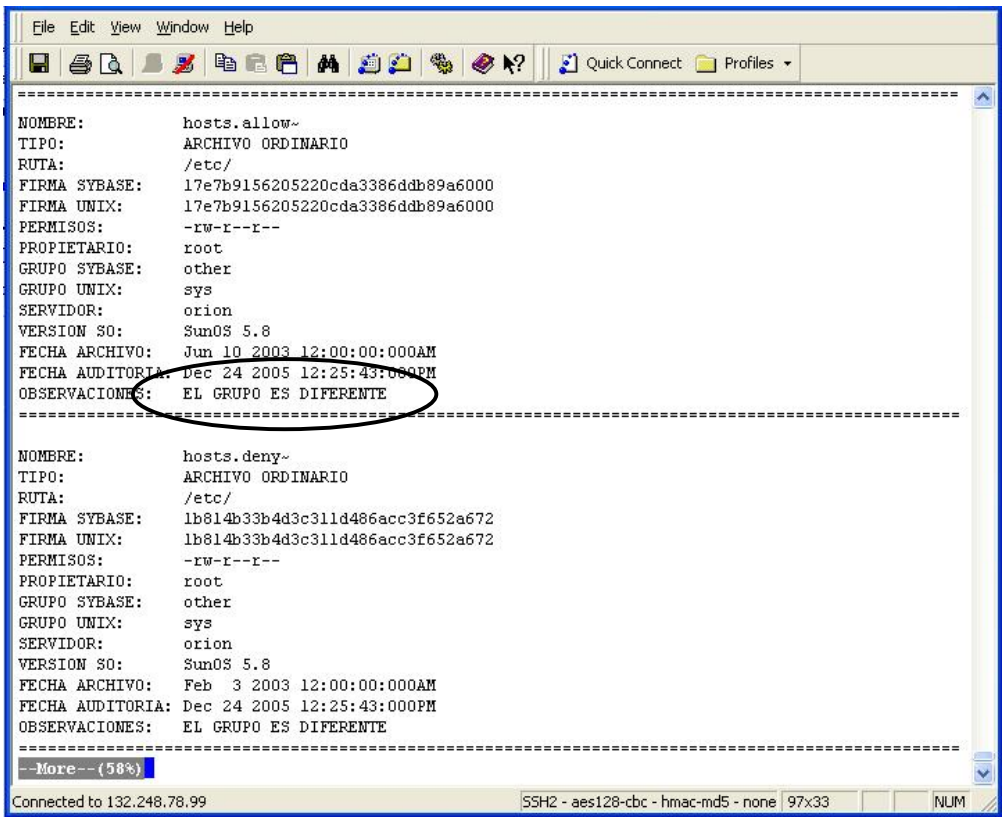
Finalmente en las últimas líneas explicará brevemente cuál fue la causa del error. El programa puede tener varias líneas, es decir si las firmas no coinciden, y los permisos o el dueño no es el mismo las observaciones nos lo describirán.

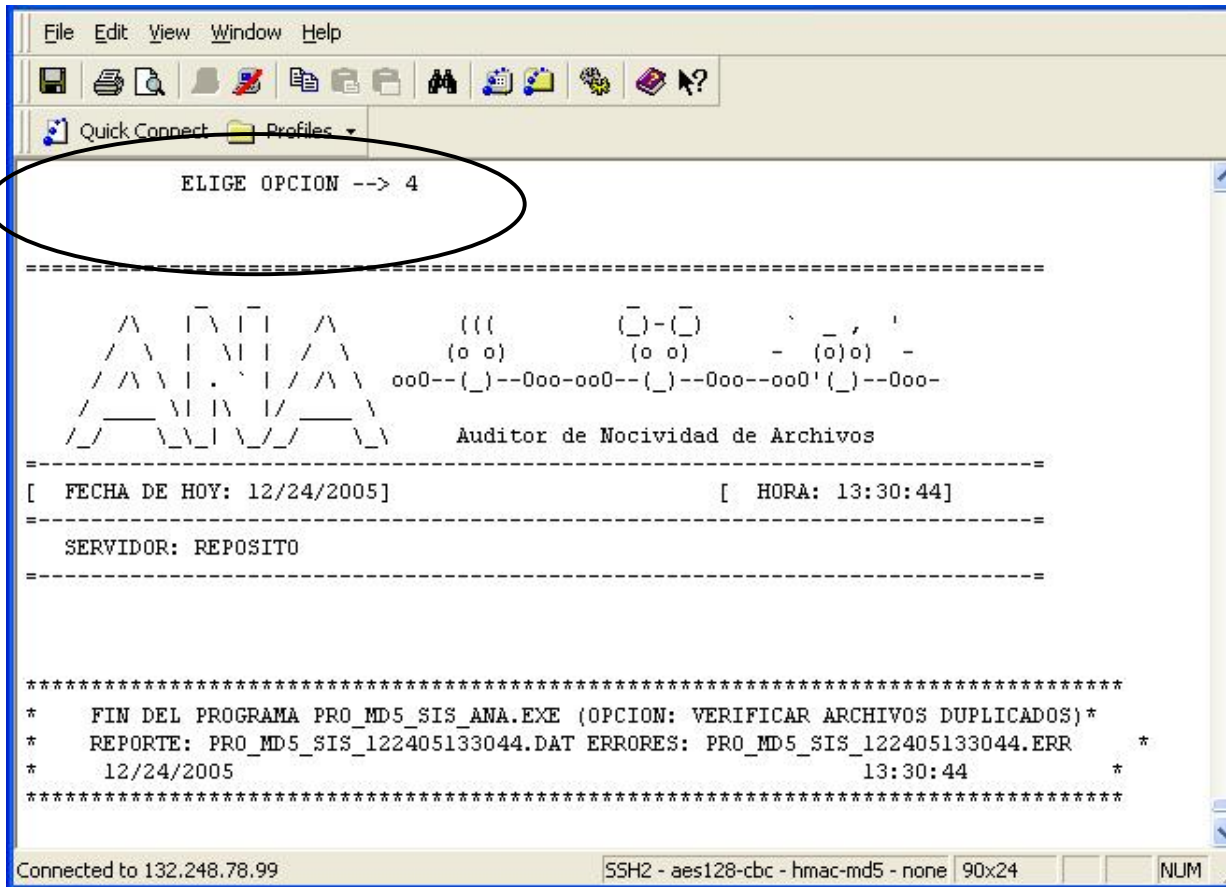



En la auditoría del directorio, se tendrán campos similares y pueden igual que en el caso anterior, avisar si se borró, modificó o se agrego un archivo dentro del directorio, y avisaría de manera global, que hay algo diferente.

En el reporte nos mostrará a detalle las posibles causas que pudieron generar el mensaje de error, como se muestra en la siguiente página.


File Edit View Window Help
Quick Connect
Profiles
NOMBRE: patch.conf
TIPO: ARCHIVO ORDINARIO
RUTA: /etc/patch/
FIRMA SYBASE: fbfabe90fdf59d1ae1d7437891a4b98d
PERMISOS SYBASE: -rw-r--r--
PERMISOS UNIX: drwxr-xr-x
PROPIETARIO: root
GRUPO: sys
SERVIDOR: orion
VERSION SO: SunOS 5.8
FECHA ARCHIVO: Feb 2 2005 12:00:00:000AM
FECHA AUDITORIA: Dec 24 2005 12:25:43:000PM
ONSERVACIONES: LOS PERMISOS SON DIFERENTES
NOMBRE: profile~
TIPO: ARCHIVO ORDINARIO
RUTA: /etc/
FIRMA SYBASE: ebddf8fd015af3ca6384c589ac24c225
PERMISOS SYBASE: -rw-r--r--
PERMISOS UNIX: drwxr-xr-x
PROPIETARIO: root
GRUPO: other
SERVIDOR: orion
VERSION SO: SunOS 5.8
FECHA ARCHIVO: Mar 6 2003 12:00:00:000AM
FECHA AUDITORIA: Dec 24 2005 12:25:43:000PM
ONSERVACIONES: LOS PERMISOS SON DIFERENTES
--More--(29%)
Connected to 132.248.78.99
SSH2 - aes128-cbc - hmac-md5 - none
97x33
NUM

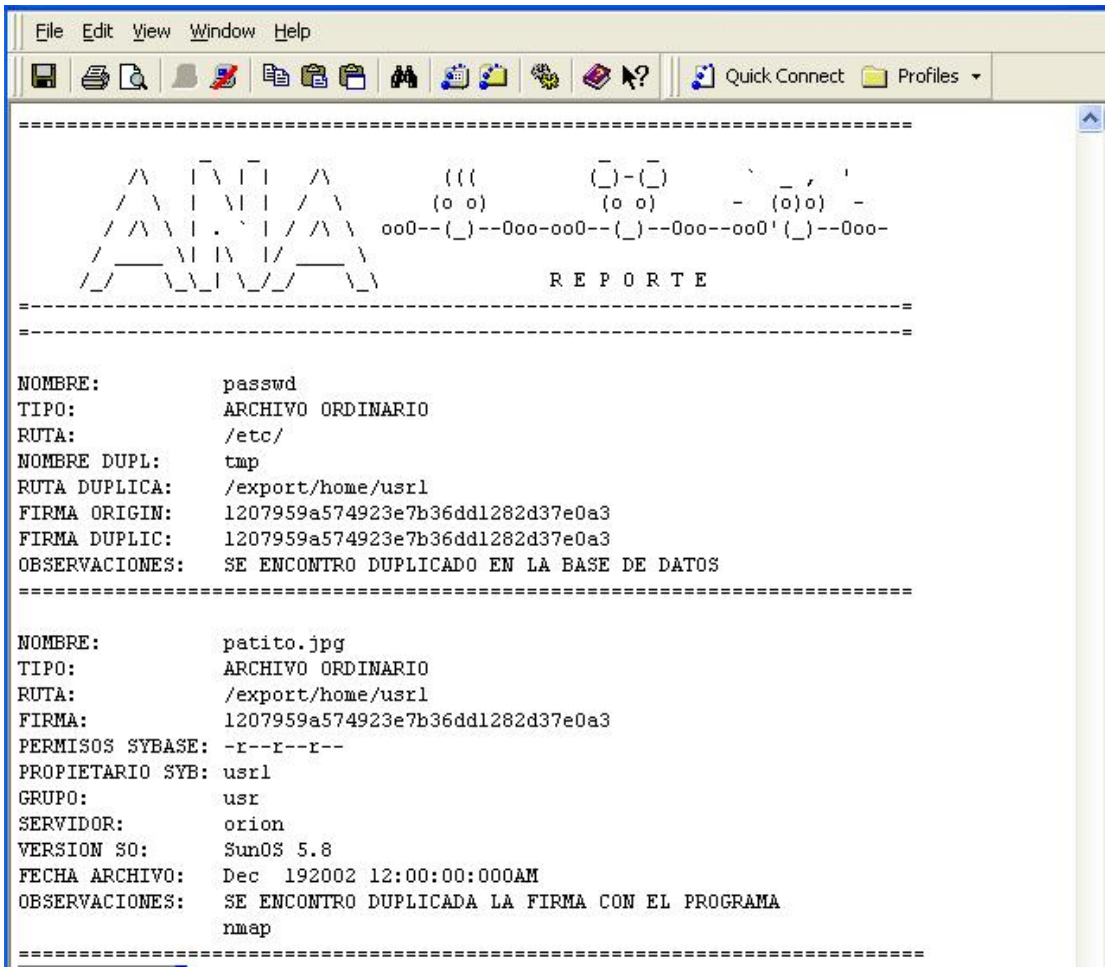

File Edit View Window Help
Quick Connect
Profiles
NOMBRE: hosts.allow~
TIPO: ARCHIVO ORDINARIO
RUTA: /etc/
FIRMA SYBASE: 17e7b9156205220cda3386ddb89a6000
FIRMA UNIX: 17e7b9156205220cda3386ddb89a6000
PERMISOS: -rw-r--r--
PROPIETARIO: root
GRUPO SYBASE: other
GRUPO UNIX: sys
SERVIDOR: orion
VERSION SO: SunOS 5.8
FECHA ARCHIVO: Jun 10 2003 12:00:00:000AM
FECHA AUDITORIA: Dec 24 2005 12:25:43:000PM
OBSERVACIONES: EL GRUPO ES DIFERENTE
NOMBRE: hosts.deny~
TIPO: ARCHIVO ORDINARIO
RUTA: /etc/
FIRMA SYBASE: 1b814b33b4d3c311d486acc3f652a672
FIRMA UNIX: 1b814b33b4d3c311d486acc3f652a672
PERMISOS: -rw-r--r--
PROPIETARIO: root
GRUPO SYBASE: other
GRUPO UNIX: sys
SERVIDOR: orion
VERSION SO: SunOS 5.8
FECHA ARCHIVO: Feb 3 2003 12:00:00:000AM
FECHA AUDITORIA: Dec 24 2005 12:25:43:000PM
OBSERVACIONES: EL GRUPO ES DIFERENTE
--More--(58%)
Connected to 132.248.78.99
SSH2 - aes128-cbc - hmac-md5 - none
97x33
NUM




***La Opción 4***

La ejecución de la opción 4 del submenú de verificación, hace un chequeo sólo de la información que está capturada y que está siendo auditada en ese momento, es decir es en ese momento cuando se hace una identificación de posibles anormalidades dentro del sistema unix. Es específicamente cuando se hace una identificación a nivel de firmas (hash md5) y trata de ubicar posibles copias no autorizadas, o de posibles programas que sean maliciosos o nocivos para el sistema en su conjunto.

Observemos la siguiente imagen.

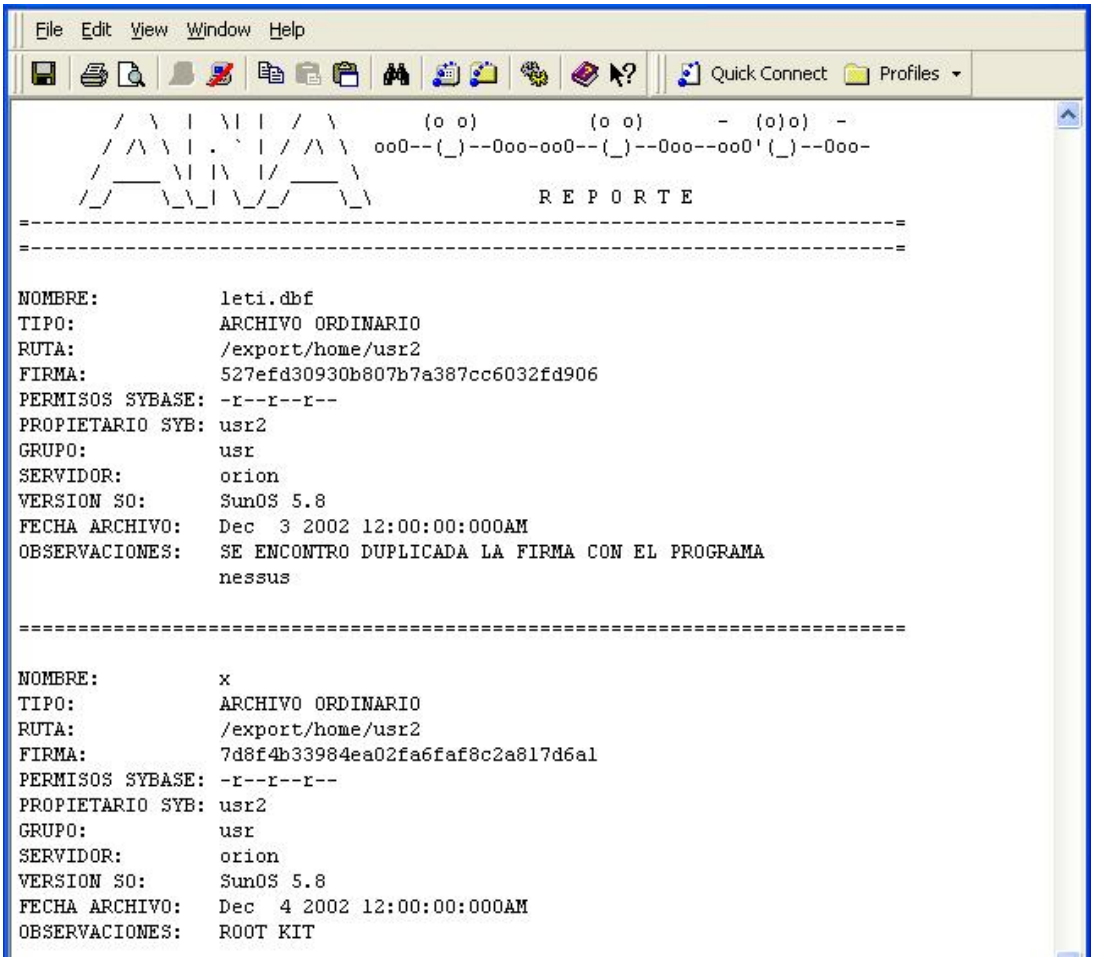



Aquí podemos ver parte del reporte generado por la opción 4 del submenú de verificación y como se ejemplifica tenemos dos errores o violaciones al sistema.

En el primer caso se hace mención que el archivo passwd ha sido copiado y está ubicado en el directorio de un usuario con otro nombre tmp; es importante hacer notar que en el campo de la firma original y la firma duplicada coinciden al 100% y esto es concluyente en el campo de observaciones.

El segundo caso es más crítico todavía, ya que la firma coincidió con una herramienta que sirve para escanear puertos unix, como podemos ver este caso ejemplifica también claramente cuando un usuario usa herramientas para hackers y como trata de pasar desapercibido cambiándole el nombre por algo que a simple vista es inofensivo, como lo es una imagen de un pato, aquí hay que hacer mención que ***ANA*** trata de tener la mayor

cantidad de hash de los programas binarios más conocidos para diferentes plataformas y que esta información es capturada por otra vía.




En esta imagen podemos ver otra parte del reporte y podemos observar que no importa como se llama el archivo; ***ANA*** es capaz de detectar programas con alta nocividad para los sistemas unix, y recordemos que según estadísticas el 80% de los ataques a las empresas es generado por personal interno y no tanto por gente externa a la empresa.

El programa detecta dos herramientas más que son muy delicadas para el manejo de un usuario, Nessus es una herramienta que se explicó dentro de este trabajo y que sirve para buscar posibles fallas de seguridad en un sistema unix.

Observamos en la imagen la detección de Un *root kit,* que es un conjunto de herramientas usadas frecuentemente por los intrusos o crackers que consiguen acceder ilícitamente a un sistema informático. Este tipo de herramientas sirven para esconder los procesos y archivos que permiten al intruso mantener el acceso al sistema, a menudo con fines maliciosos.

# CONCLUSIONES

# CONCLUSIONES

Por principio podemos decir que el objetivo de esta tesis, se alcanzó en su totalidad, al generar un software, que responde a las necesidades de las tareas de seguridad y de la posibilidad de auditar un sistema unix, además se ha logrado reunir información útil para cualquier usuario encargado de la administración de un equipo de cómputo, por un lado se han documentado algunos tipos de ataques nocivos y a la par, la tesis ofrece una documentación de diversas herramientas de cómputo que pueden ser útiles, como alternativas de prevención para el mejoramiento de la seguridad de un sistema.

La documentación de estas herramientas, obedece a su relación con nuestro reto de construir una herramienta "mejorada", en la que se ofrezca una alternativa para efectuar una auditoría confiable. El auditor construido **(*ANA),*** no ha perdido de vista su implicación en el mundo de software libre, y considera que con el paso del tiempo, logrará robustecerse, a través de su base de datos, para ofrecer un servicio de buena calidad.

Como decíamos antes, el programa ha aprovechado las características de las herramientas de seguridad de software libre como es el caso de MD5, que generará las firmas para un diagnóstico y detección de errores, pero incrementará sus capacidades con la "memoria" (base de datos), que se le ha implementado; este repositorio, ha tomado también la referencia de otra herramienta, para almacenar información súmamente útil, como lo son las firmas, me refiero específicamente a la herramienta snefru.

Esta herramienta de seguridad snefru, ha aportado al sistema la lógica para realizar el diagnóstico, pero snefru trabaja de manera local en un equipo recientemente configurado, y también de manera local se genera su diagnóstico, por lo tanto aún cuando es una herramienta súmamente eficiente, y como hemos analizado, utilizada por multitud de usuarios, ofrece ese servicio, con una certeza muy frágil.

Nuestro ***A***uditor de ***N***ocividad de ***A***rchivos a diferencia de esas condiciones, trabaja de manera local y también remota, cuenta con la base de datos que se le ha implementado y al que llamamos "repositorio" o bitácora de firmas, que detectará que la firmas generadas son diferentes, ***ANA*** no da por hecho que la firma almacenada es la correcta, lo que parecería disminuir la confianza, pero fortalece la precaución, pues ofrece la posibilidad de conocer que existe una diferencia, por mínima que ésta sea entre las firmas generadas, y ese diagnóstico lo entrega de manera inmediata una vez que se corre el programa, dicho de otra forma, la firma que se generó por el auditor que en ese momento se está corriendo y la firma generada en una previa corrida deben ser analizadas con detenimiento, pues son susceptibles de haber sido invadidas por software nocivo. Si extrapolamos este servicio a la red, podremos asegurar que la base da datos, "la memoria de ***ANA***", será a lo largo del tiempo altamente confiable.

***ANA***, como su nombre lo indica, realiza de manera constante este proceso al sistema en que resida, su administrador debe estar actualizado con las referencias de nuevos software

intrusos, más aún, si consideramos su uso en la red, de la cual han hecho presa innumerables hackers, como vimos en el capítulo I.

***ANA*** aplicará todos los principios de auditoría a un sistema estable, siempre y cuando su administrador sea lo suficientemente responsable para ejecutarlo, ejecutándolo como un proceso parcial o completo de así desearlo, entonces, al revisar el árbol de directorios que consideremos vulnerable, de manera general o parcial, podremos tener una respuesta que de igual forma garantizará el correcto funcionamiento del sistema, que auditemos.

El auditor logrará detectar al software nocivo, pero además, al reconocerlo y gracias al repositorio de información, ofrecerá los antecedentes a manera de bitácora, por lo que de ser necesario los reportará al administrador, de esta forma se podrán tomar las medidas necesarias, y detectaremos el ataque de manera oportuna, para hacer los ajustes antes de que se registren fallas irreversibles.

***ANA*** creada de inicio para un problema específico, puede ser puesta a disposición del público en la red, logrando concentrar un gran cúmulo de información, en beneficio de todos los usuarios que solicitarán su servicio, lógicamente al ser inspirada en software libre, correspondería para ser distribuida de manera gratuita, como el software en el que se inspiró.

Por lo tanto, nuestra confianza depositada en una alarma que detecte el menor cambio por el resultado de la auditoría, nos garantiza, la tranquilidad de manejo en condiciones normales de operación, pero más allá de esto, su bitácora automática nos ofrecerá la posibilidad de disminuir una de las más largas tareas de revisión, en una herramienta que logrará mediante una corrida, detectar al software intruso.

Por último cabe mencionar, que este concentrado es un cúmulo de información que puede ser utilizada a manera de introducción para la especialización, de los ingenieros en computación recién egresados, para su introducción en el área de seguridad informática.

# GLOSARIO DE TÉRMINOS

# GLOSARIO DE TÉRMINOS

| CONCEPTO | SIGNIFICADO |
|---|---|
| **Antivirus** | Son todos aquellos programas que permiten analizar memoria y unidades de disco en busca de virus. Una vez el antivirus ha detectado alguno de ellos, informa al usuario procediendo inmediatamente y de forma automática a desinfectar los archivos, directorios, o discos que hayan sido víctimas del virus. Aplicación cuya finalidad es la detección y eliminación de virus, troyanos y gusanos informáticos. |
| **Apuntador** | Un puntero o apuntador es una variable que contiene la dirección de una variable |
| **Archivo** | Es una estructura que organiza registros. Es sólo un conjunto de información con un nombre (llamado *nombre del archivo).* Grupo de datos relacionados entre sí que se procesan juntos |
| **Auditoría** | Es una técnica especializada, orientada al examen y evaluación de los sistemas de información automatizadas, verificando la corrección y confiabilidad de la información procesada por medios electrónicos, de acuerdo con normas técnicas y reglamentarias. |
| **Auditoría Informática** | es la revisión y la evaluación de los controles, sistemas, procedimientos informáticos, la utilización de los equipos de cómputo, eficiencia, seguridad, y el procesamiento de la información, a fin de que por medio del señalamiento de cursos alternativos se logre una utilización más eficiente y segura de la información que servirá para una adecuada toma de decisiones |
| **Autenticación De Usuario** | Es el proceso de determinar si una persona o una empresa están autorizadas para llevar a cabo una acción dada. Algunos sistemas de autenticación incluyen tanto identificación como autorización, mientras que otros solamente incluyen uno u otro. Cuando usted inserta su tarjeta débito en un ATM, se le pide que ingrese su Número de Identificación Personal (conocido como PIN o NIP). El NIP le indica al Cajero Automático que está autorizado para llevar a cabo la transacción solicitada. |
| **Backup** | Copia de seguridad o respaldo de una base de datos, servidor, computadora o archivos. |
| **Bibliotecas** | Colección o conjunto de programas desarrollados por un mismo fabricante que suelen ser compatibles e ínter operables entre sí. |

| | |
|---|---|
| **Buffer** | Espacio de memoria que se utiliza como regulador y sistema de almacenamiento intermedio entre dispositivos de un sistema informático. Así, por ejemplo, las impresoras suelen contar con un buffer donde se almacena temporalmente la información a imprimir, liberando a la memoria del ordenador de dichos datos, y permitiendo que el usuario pueda seguir trabajando mientras se imprimen los datos. También existen buffers entre diferentes dispositivos internos del ordenador. |
| **Cache De Memoria** | Llamada también a veces almacenamiento caché ó RAM caché, es una parte de memoria RAM estática de alta velocidad (SRAM) más que la lenta y barata RAM dinámica (DRAM) usada como memoria principal. La memoria caché es efectiva dado que los programas acceden una y otra vez a los mismos datos o instrucciones. |
| **Ciberespacio** | Según la definición de la Unesco, el "ciberespacio" es un nuevo ambiente humano y tecnológico de expresión, información y transacciones económicas. Consiste en personas de todos los países, de todas las culturas e idiomas, de todas las edades y profesiones proporcionando y requiriendo información; una red mundial de computadoras interconectadas por la infraestructura de telecomunicaciones que permite la información en tránsito sea procesada y transmitida digitalmente. |
| **Cinta** | Las unidades de cintas magnéticas son utilizadas para copias de archivos o backup debido a su alta capacidad en un espacio reducido y bajo costo por Mbyte en comparación a los disquetes y los discos ópticos. Puede almacenar fácilmente varios miles de MB. La cinta contiene información en forma secuencial, por lo cual, la lectura debe hacerse de la misma manera, haciendo muy lento el proceso de búsqueda de archivos y la transferencia de información entre ésta y la unidad de proceso. |
| **Comando** | Instrucción que un usuario da al sistema operativo de la computadora para realizar determinada tarea. |
| **Compactación** | Proceso de reducir el tamaño de un archivo para ahorrar espacio o para transmitirlo a mayor velocidad. |
| **Compilador** | Programa traductor que genera lenguaje máquina a partir de un lenguaje de programación de alto nivel basado en el lenguaje. Programa capaz de traducir un código fuente, escrito en el lenguaje de alto nivel que sea, a un código objeto escrito en lenguaje de maquina. |

| | |
|---|---|
| **Configurar** | Adaptar una aplicación software o un elemento hardware al resto de los elementos del entorno y a las necesidades específicas del usuario. Es una tarea esencial antes de trabajar con cualquier nuevo elemento. La tendencia actual es a reducir las necesidades de configuración mediante sistemas que permiten al nuevo elemento detectar en qué entorno se instala, configurándose automáticamente sin requerir la participación del usuario. Cuando ésta es necesaria, se intenta facilitar al máximo el proceso de configuración. |
| **Consulta** | Interrogación realizada a una base de datos, en la que se requiere una información o informaciones concretas en función de unos criterios de búsqueda definidos. |
| **Criptografía** | La criptografía, en el contexto de redes informáticas, es la ciencia que estudia los métodos y procedimientos, mediante algoritmos matemáticos, para modificar los datos de tal manera que solamente las personas que tengan la llave adecuada puedan a) tener acceso a la versión original de los mismos (confidencialidad) y b) asegurar que estos datos no fueron modificados entre el remitente y el destinatario (integridad). |
| **Dbms** | (DataBase Management System) Sistema de Administración de Base de Datos. Conjunto de programas necesarios para facilitar el almacenamiento, el acceso a bases de datos, la seguridad, y el mantenimiento de su integridad. |
| **Dml** | (Data Management Language)Lenguaje de Manipulación de Datos |
| **Dos** | Disk Operating System -- DOS (Sistema Operativo en Disco) DOS fue el primer sistema operativo para ordenadores personales. Se basa en mandatos que se escriben línea por línea y fue desarrollado por Bill Gates para IBM, si bien antes de la aparición de los ordenadores personales IBM desarrolló otro DOS para anteriores ordenadores. No confundir con DoS (*Denial of Service*), con o minúscula. |
| **Encriptar** | Técnica por la que la información se hace ilegible para terceras personas. Para poder acceder a ella es necesaria una clave que sólo conocen el emisor y el receptor. Se usa para evitar el robo de información sensible, como números de tarjetas de crédito |
| **Firewall** | El firewall es un dispositivo de control de acceso que determina qué conexiones son permitidas “ingresar a” y “salir de” una red en base a un conjunto de reglas predefinidas. |

| | |
|---|---|
| **Gusano** | Es un programa que se mantiene y duplica solo, usualmente consume memoria, causando por tanto que el equipo deje de responder. Compare con virus. |
| **Hackers** | Persona muy aficionada y hábil en informática, que entra ilegalmente en sistemas y redes ajenas: el hacker se adentra en un sistema como desafio intelectual. |
| **Hardware** | Dícese de cualquier componente físico relacionado con el sector informático. Antónimo 'software' (Soft = Blando) por oposición a 'hardware' (Hard = Duro). Componentes materiales del ordenador pantalla, chips, etc. Conjunto de dispositivos de los que consiste un sistema. Comprende componentes tales como el teclado, el Mouse, las unidades de disco y el monitor. |
| **Huecos De Seguridad** | Son imperfecciones en el diseño del software, y representan el blanco de los ataques informáticos. |
| **Ids** | (Intrusion Detection Systems). Un sistema de detección de intrusos (IDS) es un proceso o dispositivo activo que analiza la actividad del sistema y de la red por entradas no autorizadas y/o actividades maliciosas. La forma en que un IDS detecta las anomalías pueden variar ampliamente; sin embargo, el objetivo final de cualquier IDS es el de atrapar a los perpetradores en el acto antes de que hagan algún daño a sus recurso |
| **Índices** | Se usan en las aplicaciones de bases de datos para indicar la operación de ordenar los registros contenidos en ella de manera especial, en función de unos parámetros definidos previamente. |
| **Interfaz.** | Conexión mecánica o eléctrica que permite el intercambio de información entre dos dispositivos o sistemas. Habitualmente se refiere al 'software' y 'hardware' necesarios para unir dos elementos de proceso en un sistema o bien para describir los estándares recomendados para realizar dichas interconexiones. Es más conocido por su denominación inglesa 'interface'. |
| **Intrusión** | Conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso. |
| **Log** | Archivo que registra movimientos y actividades de un determinado programa (log file). |
| **Login** | Entrada de identificación. |
| **Password** | O contraseña. Se denomina así al método de seguridad que se utiliza para identificar a un usuario. Es frecuente su uso en redes, sistemas operativos y DBMS. Se utiliza para dar acceso a personas con determinados permisos. |

| | |
|---|---|
| **Plataforma** | Se refiere a la arquitectura física, de hardware de la computadora. |
| **Www** | Web o la web, la "red" o www de "World Wide Web", es básicamente un medio de comunicación de texto, gráficos y otros objetos multimedia a través de Internet, es decir, la web es un sistema de hipertexto que utiliza Internet como su mecanismo de transporte o desde otro punto de vista, una forma gráfica de explorar Internet. |
| **Roles** | Son conjuntos de privilegios que pueden concederse 'de golpe' a un usuario. Los Privilegios pueden ser de Objetos (para INSERT, SELECT, UPDATE, DELETE, EXECUTE...) o del Sistema (para crear tablas, vistas...). |
| **Servidor Proxy** | Componente de un firewall que maneja el tráfico de Internet hacia y desde una red de área local (LAN) y puede desempeñar otras funciones, como el almacenaje de un documento y control de acceso. |
| **Software** | Dícese de cualquier componente lógico (programas, aplicaciones) relacionado con el sector informático. Componentes inmateriales del ordenador programas, sistemas operativos, etc. Conjunto de instrucciones mediante las cuales la computadora puede realizar tareas. Los programas, los sistemas operativos y las aplicaciones son ejemplos de software. |
| **Software Libre** | "Software Libre" se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:<br><br>o La libertad de usar el programa, con cualquier propósito (libertad 0).<br>o La libertad de estudiar como funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.<br>o La libertad de distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2).<br>o La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto. |
| **SQL** | (Structured Query Language). Es un estándar en el lenguaje de acceso a bases de datos. Originalmente, era un lenguaje de acceso al sistema de gestión de bases de datos denominado DB2 en plataformas 390 de IBM |

| | |
|---|---|
| **Tabla** | Es una unidad donde se almacena el conjunto de datos para la base de datos. Estos datos estarán ordenados en columnas verticales, a las cuales se les denomina campos. |
| **Unix** | Potente y complejo sistema operativo multiproceso/multitarea y multiusuario orientado a comunicaciones y gran devorador de 'RAM'. Fue creado en 1969 por Ken Thompson y Dennis Ritchie (de la empresa norteamericana 'AT&T Laboratories') coincidiendo con el nacimiento de 'Internet'. |
| **Utilerías** | Herramientas de hardware o de software con un propósito o utilidad especifica. |
| **Vacunación** | Mediante esta técnica el programa antivirus almacena información sobre cada uno de los archivos. En caso de haberse detectado algún cambio entre la información guardada y la información actual del archivo, el antivirus avisa de lo ocurrido. Existen dos tipos de vacunaciones: Interna (la información se guarda dentro del propio archivo, de tal forma que al ejecutarse él mismo comprueba si ha sufrido algún cambio) y Externa (la información que guarda en un archivo especial y desde él se contrasta la información). |
| **Variable** | Es un identificador que se utiliza para representar un dato individual; es decir, una cantidad numérica o carácter de forma parecida a la constante pero en este caso, su valor es variable, asignado en alguna parte del programa. El valor que el programa asigna a la variable puede ser recuperado referenciando al nombre de la variable. Sin embargo, el tipo de dato asociado a la variable no puede cambiar, esto se refiere a que no se puede asignar por ejemplo un carácter a una variable de tipo int. |
| **Virus** | Programa que está diseñado para copiarse a sí mismo sin conocimiento del usuario y con la intención de infectar el sistema operativo y/o aplicaciones, cuyos efectos pueden variar dependiendo de cada virus: mostrar un mensaje, sobrescribir archivos, borrar archivos, enviar información confidencial mediante emails a terceros, etc. |
| **Vulnerabilidades** | Por vulnerabilidad entendemos la exposición latente a un riesgo. En el área de informática, existen varios riesgos tales como: ataque de virus, códigos maliciosos, gusanos, caballos de troya y hackers. |

# BIBLIOGRAFÍA

Horton Mike, Mugge Clinton, *Claves Hackers,*
Madrid McGraw Hill / Interamericana, 2004, 288 pp**.**

Keigh J. Jones, Shema Mike, Jonson Bradley C. Superutilidades Hackers
Madrid Mc. Graw Hilll / Interamericana de España, 2003, 721 pp.

Morant R. José Luis, *Seguridad y protección de la información.*
Madrid McGraw Hill / Interamericana, 1994, 566 pp.

Rosales Herrera Humberto David. *Determinación de riesgos en los centros de cómputos*.
México Trillas. /1996, 245 pp.

Montoya Edwin, Cañón Jorge Alonso "Riesgos Políticas y Herramientas de Seguridad en Redes"
*Revista Universidad Eafit*, España, Julio-Agosto- Septiembre de 1997, pp 69-86

Villalón Huerta Antonio, "Seguridad en UNIX y redes",
*Tesis*, España, Julio 2002, pp 1-17

Cerini María Dolores, Prá Pablo Ignacio, "Plan De Seguridad Informática"
*Tesis Facultad de Ingeniería*, Argentina, 2002, pp 10-15

Núñez Sandoval Alejandro, "Curso Herramientas de Seguridad en Unix"
*Departamento de Seguridad UNAM*, México 1995

# SECCIÓN DE ANEXOS

# ANEXO I. Recomendaciones de Seguridad en Sistemas de Cómputo[1]

En los últimos años el tema de la seguridad en las redes de computadores se ha tornado en un asunto de primera importancia dado el incremento de prestaciones de las mismas, así como la imparable ola de ataques o violaciones a las barreras de acceso a los sistemas implementados en aquellas.

Los "incidentes de seguridad" reportados continúan creciendo cada vez más y a un ritmo más acelerado, a la par de la masificación del Internet y de la complejidad del software desarrollado.[2]

## 1. Efectuar un análisis de riesgos

Esto se suele mencionar en la literatura como el primer paso a realizarse cuando se plantea la seguridad en un sistema.[3] La idea es muy sencilla: trazar todos los elementos que conforman nuestro sistema (hardware y software) y observar cuáles involucran más o menos riesgo. Esto desembocará en un plan de seguridad cuyo objetivo es disminuir el riesgo total del sistema, que se puede modelar como la suma de los riesgos de sus componentes:

**RIESGO TOTAL** = RIESGO (componente 1) + RIESGO (componente 2) …

El riesgo de cada componente está en función directa a las pérdidas que ocasionaría el que éste deje de operar, así como en función de cuán vulnerable es dicho componente en este momento. Por ejemplo, una base de datos de clientes involucra un gran riesgo debido al gran valor que la información representa para una organización; pero una simple PC Windows de la misma organización conectada directamente al Internet (sin firewall/ roxy de por medio) también lo representa, debido a que puede ser objeto de un ataque desde el exterior, con el posible riesgo de fácil propagación hacia otros computadores de nuestra red. El riesgo no es fácil de cuantificar, siendo en general un estimador subjetivo. A modo de ejemplo podríamos plantear una fórmula como la que sigue:

**RIESGO**(componente) = P * V

Donde P=pérdida, es la pérdida en dinero que implicaría la inoperatividad del componente hasta su reparación, aunque se pueden agregar otros estimadores como el desprestigio ante nuestros clientes. A veces ayuda considerarlo como "el valor" que representa para la organización. V=vulnerabilidad, es tanto o más subjetiva puesto que no hay una manera segura de establecer para todos los casos si los supuestos mecanismos de protección (del componente) son o no realmente confiables.

---

[1] Diego Bravo Estrada, *15 Recomendaciones de seguridad en sistemas distribuidos de cómputo (V0.11)*

[2] El software se torna más inseguro y complejo http://www.counterpane.com/crypto-gram-0003.html#SoftwareComplexityandSecurity, El CERT/CC http://www.cert.org/

[3] 5 Steps to enterprise security http://www.eweek.com/category/0,3660,s=25132,00.asp

Así por ejemplo, podríamos suponer que la vulnerabilidad de ciertos documentos importantes es muy baja, puesto que están protegidos por cierto antivirus. Sin embargo, esto realmente estará en función de diversas características del antivirus, como pueden ser: recientes actualizaciones, ejecución automática, capacidad de eliminar virus locales, licencias no caducadas, configuración adecuada, etc. Pero por algo se debe comenzar. Por ejemplo, se puede asignar números como 1, 2, 3, 4, para señalar una vulnerabilidad mínima, regular, importante y peligrosa, respectivamente. Para una extensa (y compleja) explicación,[4] ésto normalmente demanda el acopio de mucha información, la que muchas veces no está a cargo de una sola persona. Esto implica que todo el staff encargado de las distintas partes del sistema debe colaborar en el análisis.

### 2. Lo más valioso debe alejarse de lo más vulnerable

En la fórmula del "riesgo" propuesta arriba, es evidente que los componentes de nuestro sistema con algo valor y alta vulnerabilidad serán de lejos los que presenten mayor riesgo. Sin embargo, en muchos casos no es sencillo disminuir el valor de cierto componente (y por tanto la pérdida en caso de problemas), y tampoco se puede eliminar completamente la vulnerabilidad del mismo (por ejemplo, si está de cara a Internet.)

En este caso lo que conviene es separar o dividir este componente en dos partes suficientemente alejadas e independientes a fin de que el riesgo total disminuya. Por ejemplo, los portales de comercio electrónico deben dar cara a Internet (siendo vulnerables en principio) y a la vez manejar información muy costosa (como transacciones con tarjeta de crédito.) Esto los convierte en un sistema de alto riesgo. Sin embargo es casi universal la separación que se efectúa entre los componentes dedicados a dar cara a Internet (como los Web Servers) y los componentes que manipulan la información comercial (generalmente sistemas DBMS.) En términos prácticos, esto significa que el *hacker* no puede acceder directamente al DBMS (lo que sería catastrófico), y sólo podría atacar al Web Server, lo que en principio no acarrea mayores consecuencias.

### 3. Mantener las cosas simples

Un sistema complejo es más difícil de asegurar y potencialmente proporciona una mayor cantidad de puertas abiertas a los atacantes. En general, es recomendable intentar dividir el problema mediante la simplificación de la configuración, para así identificar los puntos o rutas de control vulnerables para incrementar la seguridad.

Un ejemplo frecuentemente citado es la seguridad de una red de estaciones Windows manipulada por usuarios inexpertos. En muchos casos no resulta práctico efectuar un profundo trabajo de seguridad en las estaciones (más allá de la instalación del antivirus, por ejemplo), puesto que por un lado, son muchas, y por otro, los usuarios suelen modificar la configuración en tanto instalan y desinstalan su software sin dar aviso a nadie. En estos casos es aconsejable centrarnos en un único punto que centralice la seguridad de todas las

[4] ] Extenso documento sobre administración de riesgos, http://cisac.stanford.edu/docs/soohoo.pdf

estaciones. Una configuración de red que obligue a que todo su tráfico pase a través de un *gateway* permitiría crear un único punto de control para todas las estaciones, con lo cual simplificamos la administración. Todo esto generalmente conlleva a situaciones en las que hay que hacer un compromiso.

**4. Asegurar la seguridad en todos los niveles**

Ésto se puede expresar más sencillamente como: no confiar el sistema a un único mecanismo de seguridad.

La información fluye a través de los distintos componentes y/o capas del sistema y son muchas las instancias en las que se puede mejorar su seguridad.

La recomendación estipula que utilicemos todas estas instancias a pesar de que en principio puedan parecer redundantes.

Por lo general los administradores tienden a preocuparse por un único punto de acceso desde donde supuestamente hay una gran probabilidad de ser atacados (por ejemplo, la conexión a Internet.) Por tanto se invierte esfuerzo y dinero en controlar este único punto bajo la asunción de que es la única puerta de entrada a los maleantes y que por tanto, tras asegurarla, todo el sistema quedará seguro. Esto tiene dos problemas:

- ✧ Muchos ataques o “vulnerabilidades” se originan (de forma inocente o intencional) desde dentro de la organización
- ✧ El sistema que controla la “puerta” siempre puede fallar

Esto obliga a implementar la seguridad no en un único punto evidentemente vulnerable, sino en todos los lugares por donde fluye la información al interior de cada componente involucrado.

Un ejemplo típico lo constituye un filtro de paquetes TCP/IP, operando a nivel de capas 3 y 4 del modelo OSI.

Estos por lo general hacen un gran servicio restringiendo accesos a servicios de red internos y se suelen colocar en la “puerta” o “*gateway*” que da cara a Internet o a una red insegura a modo de “firewall”. Tomando literalmente la idea de “niveles” o “capas”, podríamos pensar qué hacer en este *gateway* para mejorar la seguridad.

Son 7 las capas OSI y podríamos intentar añadir instancias de control adicionales a las mencionadas. Por ejemplo, en la capa 7 (aplicación) es frecuente y muy recomendable el empleo de proxys HTTP. Algunos *firewalls* proporcionan control a nivel de la capa 2 (específicamente, del MAC Address.) Sin embargo, ésto se debe complementar con otras medidas que van más allá del gateway y que se implementan a lo largo de todo el sistema

(pudiéndose considerar como niveles o “capas” adicionales), tales como redes perimétricas, el uso de encriptación, etc.

### 5. Encriptar tanto como sea posible

La encriptación es un tema complejo pero cuya implementación resulta cada vez más sencilla conforme aparecen más productos. Los cambios en la legislación norteamericana con respecto a la exportación de productos que encriptan, son un incentivo claro para que los desarrolladores y vendedores se interesen más en el tema.

Los canales de comunicación más vulnerables o de mayor cercanía al público requieren una encriptación “más fuerte”, es decir, más difícil de descifrar por los curiosos o atacantes. Cierta información conlleva más riesgo que otra, y por tanto requerirá un nivel de encriptación diferenciado.

Las herramientas capaces de hacer esto son muchas, dependiendo del contexto en que nos encontremos. Por ejemplo, los sistemas DBMS más avanzados incorporan la encriptación como una opción normal para los datos almacenados, generalmente bajo esquemas propietarios.

La tecnología de encriptación de información destinada a pasar a través de la red ha evolucionado bastante, haciéndose popular el término VPN[5] para hacer referencia a canales que encriptan la información de un modo más o menos transparente. Hay soluciones propietarias así como estándares relativamente implementados como IP Sec.

Ciertas aplicaciones estándar han recibido soluciones de encriptación también estándar. El caso del Web encriptado bajo SSL (HTTPS) junto con la industria de certificados digitales es el caso más conspicuo.

De igual modo los estándares para correo electrónico PGP (o derivados) y S/MIME son integrados cada vez con mayor frecuencia en las aplicaciones de los usuarios finales.

En nuestra organización deberíamos encriptar todo lo que sea posible. La razón de ésto es evidente si de lo que se trata es de enviar un mensaje privado por Internet. Sin embargo, al interior de la organización la encriptación puede ayudar también. Pues es usual que cierta información sea manejada exclusivamente por un número reducido de personas.

Un caso más dramático se presenta cuando un *hacker* ha logrado infiltrarse en la organización: si la información está encriptada, los datos que robe le serán inútiles dado que no posee las claves correspondientes. Naturalmente hay que sopesar los inconvenientes que trae la encriptación en términos de incomodidad de uso, costo de licencias, ciclos de CPU,

[5] ] VPN Source Page, http://www.internetweek.com/VPN/default.html; VPN Labs, http://www.vpnlabs.org/

etcétera; con el hecho de que cierta información es definitivamente de carácter público y por tanto no tiene sentido que esté encriptada.

Por ejemplo, la información que se publica en Internet vía el Web Server de la empresa. [6]

### 6. No confiar en la autenticación estándar

Las redes locales, y por extensión, el Internet, NO fueron diseñados en principio para ser seguros.

La mayoría del software y hardware creados hasta incluso mediados de los años 90, no tuvieron la seguridad como objetivo de primer orden. E implica que muchos de los sistemas que compramos y usamos en la actualidad tienen serias deficiencias, pero que por mantener la compatibilidad no se han podido corregir.

Un caso crítico es lo concerniente a la autenticación de los accesos, que e la gran mayoría de los casos se limita a proporcionar una contraseña por parte de quien pretende acceder al recurso correspondiente, cosa que funciona bien en un ambiente seguro en el que todos estamos de acuerdo en no ir más allá de este mecanismo. Pero, ésto no siempre es cierto (y menos en Internet).

Son muchos los mecanismos que puede emplear un hacker para “capturar” las contraseñas de usuarios reales, así como para “adivinar” las mismas. Otros esquemas pretenden ser más seguros basándose en el análisis de la dirección de red de quien intenta conectarse, pero de igual modo los hackers pueden “simular” una dirección de red sin mayores inconvenientes.

En tanto los ataques son cada vez más sofisticados, es de rigor que la autenticación también sea cada vez más sofisticada, lo cual implica abandonar para siempre diversos mecanismos estándar muy arraigados.

El ejemplo más paradigmático tal vez sea el comando “telnet” de los sistemas Unix, utilizado para obtener una sesión en un sistema remoto. Este comando se distribuye (y se seguirá distribuyendo sin duda) en prácticamente todas las encarnaciones de los sistemas Unix y sus clones (como Linux), es tan popular y útil, que incluso los sistemas Windows lo proporcionan, sin embargo, ningún administrador serio recomendaría su uso para una conexión remota, y quizá tampoco al interior de la red. La razón estriba en que toda la información de autenticación (el nombre de usuario y su contraseña) viaja sin encriptación tal cual es, y cualquier persona con acceso a un nodo intermedio o al canal de la conexión puede extraer esta información con relativa facilidad. Actualmente diversos productos y estándares promueven la seguridad en la autenticación. Como muestra podemos mencionar a SSH, S/Key, Kerberos, Tcpwrappers, etc.

[6] Cryptogram newsletter, http://www.counterpane.com/crypto-gram.html

## 7. No usar la configuración "estándar"

Por lo general los sistemas operativos y las aplicaciones se instalan con una configuración determinada y de carácter genérico. En muchos casos el administrador no tiene necesidad de modificarla pues el sistema aparentemente funciona bien. Sin embargo, los atacantes al no tener conocimiento de la configuración de nuestro sistema, asumen que ésta es justamente la "estándar", y programan sus ataques basados en dicha asunción.

Alguien escribió acerca de un administrador cuyo servidor web IIS fue atacado con éxito por "CodeRed", pero las páginas web publicadas no fueron alteradas debido a que antes había decidido modificar en la configuración el directorio donde se guardan los documentos html, ciertamente este tipo de cambios tienden a complicar el sistema puesto que sólo el administrador es consciente de tales, esta complejidad también la padecerá el atacante.

Lamentablemente muchas herramientas "automáticas" de configuración, en su afán de hacer la vida del administrador más sencilla, no permiten hacer modificaciones radicales. Esta tendencia a la homogeneidad en la configuración va en aumento.

## 8. La seguridad hacia el interior

Algunos reportes[7] han puesto de relieve que en una gran cantidad de casos la mayor amenaza de ataques al sistema no proviene de fuera, sino que parte desde el interior de la organización.

Muchos ataques exitosos desde el exterior han necesitado de cierta ayuda inicial activada en el interior de la organización, donde por lo general nadie sospecha de este tipo de prácticas. Entonces; el análisis de riesgos debe incluir posibles ataques originados en el interior, incluyéndose el robo de contraseñas, la modificación de archivos de configuración, la desactivación de las barreras de protección, etc. Un caso muy común de este tipo de ataque lo constituye el trabajador despedido o castigado que decide tomar venganza. Antes de retirarse definitivamente puede efectuar este tipo de tareas maliciosas e incluso ponerse en combinación con un atacante externo. En ciertos casos la simple introducción intencional de un virus puede acarrear efectos devastadores.

La única manera de reducir el riesgo en estos casos, consiste en planificar el acceso al sistema de modo tal que ningún elemento crítico dependa de una sola persona. Dicho de otro modo, para dañar un sistema, no debe bastar con un único individuo disconforme.

Esto muchas veces no es posible con los sistemas operativos comerciales actuales (por ejemplo, con las estaciones Windows XP) y se hace aún más difícil si el despedido es el administrador! Algunos sistemas intentan por tanto dividir las responsabilidades en diversos administradores, pero en el sector comercial todavía falta mucho camino por recorrer. Los

[7] El enemigo más peligroso está en el interior, http://oraclepressoffice.bulletonline.com/showrelease.php?id=128

usuarios de Linux podrían interesarse en el sistema LIDS a fin de implementar esta recomendación.

### 9. Educar a los usuarios

Una de las mayores ayudas que puede recibir un hacker que intenta infiltrarse en el sistema de una organización consiste en obtener información acerca de éste.

Las prácticas empleadas por el atacante del sistema comprenden muchas veces la interacción encubierta con los usuarios de la organización a los cuales se les extrae (sin que tomen conciencia de ésto) una serie de datos útiles para el hacker. El caso más evidente consiste en obtener "como jugando" una contraseña de parte de este incauto. Y ciertamente las contraseñas de los usuarios comunes suelen ser muy malas, por lo que pueden liberarlas inadvertidamente al interlocutor en medio de una conversación aparentemente inocente. Esto se suele denominar "Ingeniería Social".

La única forma de combatirlo es educando a los usuarios. Es necesario que sean conscientes de este tipo de ataque (o entrevista), y que una de las mejores medidas a tomarse es el uso de contraseñas suficientemente complicadas como para que no surjan de improviso en cualquier diálogo.

Otra recomendación (o norma obligatoria si se desea) consiste en que los usuarios no divulguen detalles acerca de la configuración del sistema. Esta es una práctica increíblemente extendida: Auxiliares, programadores, ingenieros, e incluso administradores, que en su deseo de hacer alarde del "super sistema" que utiliza su empresa, empiezan de pronto a nombrar todos y cada uno de sus componentes, tanto de hardware como de software, con número de versión incluido.

Esto puede ser inmediatamente aprovechado por el atacante, pues empezará a "disparar" los ataques ya conocidos para ese hardware/software específico.

### 10. No confiar (totalmente) en nosotros mismos

Puede sonar extraño, sin embargo es necesario que otra persona verifique la seguridad de nuestro sistema.

Como se sabe, existen empresas consultoras especializadas en auditar nuestra organización con este fin. Si esta última opción no es posible (por ejemplo, por el costo involucrado) entonces es de rigor solicitar a otro administrador o ingeniero que verifique las medidas que hemos considerado.

En sistemas y redes complejas es muy posible que una sola persona (nosotros) hayamos dejado pasar alguna puerta insegura. Mientras más personas verifiquen nuestro trabajo, habrá más probabilidades de que éste esté adecuadamente realizado. Esta es la idea que

está detrás de mucho software Open Source, siendo el Kernel de Linux el caso más conspicuo.
Naturalmente evitaremos mostrar estos detalles a personas ajenas a la organización, a fin de disminuir el conocimiento que se tiene en el exterior acerca de nuestro sistema.[8]

**11. Ejecutar sólo los servicios imprescindibles**

Algunas personas tienen la manía de instalar los sistemas con la mayor cantidad posible de opciones que puedan entrar en el disco duro.

Los administradores de sistemas seguros deben ir exactamente en sentido inverso: en un sistema de alto riesgo es de rigor que se ejecute únicamente lo imprescindible.

El ejemplo más conocido corresponde a los servicios de red, los cuales muchas veces vienen configurados para estar activos tan pronto como se instala un sistema operativo, creándose automáticamente nuevas oportunidades para los atacantes. El administrador debería asegurarse de que sólo esté instalado lo imprescindible para que el sistema siga en operación. Debe descartar incluso el software aparentemente más inofensivo.

Sistemas que en apariencia no tienen nada de riesgosos han presentado serias amenazas para la seguridad, como por ejemplo, las conexiones X Window, las bibliotecas compartidas, los módulos del kernel, etc.

**12. Mantenerse al día con las actualizaciones**

Esta recomendación cada vez es más crítica. El software pese a los esfuerzos y la propaganda, continuará teniendo errores y puertas ocultas, y al parecer la tendencia sigue en aumento con la complejidad del mismo, lo que implica que los vendedores deberán proporcionar parches o versiones mejoradas a sus clientes cada vez que se descubra alguna vulnerabilidad. [9]

Lamentablemente no se toma muy en cuenta, tanto por las empresas de software como por los administradores. Por ejemplo: desde que se descubre una vulnerabilidad hasta que ésta puede ser aprovechada por un atacante puede pasar mucho tiempo (incluso puede ser que nunca se logre aprovechar) lo que motiva a que las casas de software tiendan a esperar más de la cuenta para admitir la vulnerabilidad a la vez que presentan el parche correspondiente.

En sistemas operativos que involucran muchos componentes (como casi todos en la actualidad) es de esperarse que las vulnerabilidades surjan con una frecuencia extremadamente alta.

[8] Outsourcing para auditorías de seguridad, http://www.networknews.co.uk/Analysis/1129412

[9] Consecuencias de no aplicar parches, http://zdnet.com.com/2100-11-527502.html?legacy=zdnn

Desde el punto de vista de la casa de software, esto significaría incomodar con demasiada frecuencia a los administradores por lo que usualmente se distribuyen parches a intervalos relativamente extensos y que corrigen de un sólo golpe todas vulnerabilidades halladas hasta la fecha, generalmente acompañadas de aditamentos al sistema de modo tal que no luzca como un simple parche (piénsese en los "service pack".) lógicamente, los administradores que se limitan a esperar el lanzamiento del próximo super-parche corren un gran riesgo en este lapso.

En un ambiente de alta seguridad, los parches de cada componente deben aplicarse tan pronto como sea posible, si no deseamos que la administración se convierta en un parchar y parchar, deberíamos asegurarnos que nuestro sistema realmente tenga pocos componentes (y por tanto, menos necesidad de parches.)

Es necesaria además una rigurosa política de actualización, hoy existen múltiples canales en Internet proporcionados por los proveedores de software, y solucionadotes de problemas, de igual modo entonces, mantenerse informado acerca de las nuevas vulnerabilidades descubiertas, se logra con diversas publicaciones electrónicas especializadas.[10]

### 13. Escaneo regular

Un "scanner" es un programa que intenta indagar acerca de qué servicios proporciona un computador de la red. Una vez que se conocen los servicios, un atacante puede centrar sus ataques hacia los mismos.

Esta herramienta es empleada regularmente por los hackers. Sin embargo, se pueden emplear en beneficio propio, puesto que así nosotros podremos descubrir si hemos dejado ciertas puertas abiertas. El scanner se limita a proporcionar esta información y por tanto su uso es seguro e inofensivo.

El "escaneo" se debe hacer desde fuera de nuestra red, desde un computador de Internet, así como desde su interior contra nuestras estaciones de trabajo.[11]

### 14. Descargas de software de Internet

Como regla general, el software no debería ser descargado de Internet, sino adquirido de una fuente confiable. Sin embargo en muchos casos es imprescindible por lo que deberíamos seguir algunas reglas mínimas para no terminar descargando un virus o un "caballo de Troya".

En primer lugar, debemos asegurarnos que el software que descargamos es realmente lo que hemos pretendido descargar.

[10] Securityfocus Mailing lists, http://online.securityfocus.com/archive; Cert Mailing lists, http://www.cert.org/contact_cert/certmaillist.html

[11] ] The Art of port scanning, http://www.dlhoffman.com/publiclibrary/software/nmap_doc.html

La idea es que un atacante podría modificar los datos que estamos recibiendo e introducir modificaciones en aquello que descargamos. Luego nosotros confiadamente ejecutamos el archivo en cuestión y… la historia es conocida.

Para evitar esto, cada vez con mayor frecuencia los portales de descarga incluyen una serie de alternativas para verificar que lo que hemos descargado no ha sido alterado. Un método común consiste en el "checksum MD5" que el usuario debería aplicar a todo archivo que descargue, a fin de obtener una especie de "firma" que ha de coincidir con la que se anuncia en el portal.

Es asimismo muy recomendable que los portales de descarga utilicen certificados digitales, de modo que no seamos presas de un portal ficticio implementado por atacantes (esto último es realmente es muy difícil, pero no imposible).

**15. Establecer planes de contingencia y sistemas de respaldo**

No existe ninguna garantía de que nuestro sistema sea invulnerable. Más allá de las medidas que podamos adoptar, siempre existirá la posibilidad de ser atacados. Esto nos obliga a tener presentes ciertas medidas de contingencia traducidas preferentemente en políticas de seguridad bien establecidas.

En otras palabras, debemos imaginarnos sucesivamente un conjunto de escenarios de ataques exitosos. ¿Qué hacemos si…

- ✧ Sospechamos que un hacker está atacando el firewall
- ✧ Sospechamos que ya ha tomado control del firewall
- ✧ Comprobamos que ya ha tomado control del firewall
- ✧ Sospechamos que el servidor de base de datos ha sido alterado
- ✧ Descubrimos que las PCs Windows han sido infectadas con un virus

Las posibilidades evidentemente son muchas, y las acciones a tomarse también, sin embargo es mejor tener ésto por escrito a no tener absolutamente nada el día que las quejas sobre sucesos extraños en el sistema empiecen a acaecer. Ahora bien, cualquier administrador medianamente decente tendrá establecida una política de backups para situaciones críticas. En el caso de ataques a través de la red las consecuencias pueden obligar a requerir de estos backups, por lo que es imperativo que éstos estén actualizados y preparados para su uso.

Este tipo de análisis no debe ser extraño puesto que los sistemas pueden tener problemas por múltiples motivos que no tienen nada que ver con los ataques (por ejemplo, un fallo de hardware) que deberían también poseer planes de contingencia.

Los administradores más minuciosos podrían efectuar simulacros de desastres: No hay nada más frustrante en medio de un desastre que descubrir que los backups diarios nunca se grabaron por problemas en la unidad de cinta.

### 16. Mantener contacto con el proveedor de líneas de comunicación

Diversos problemas pueden aparecer en las comunicaciones desde nuestra red hacia el exterior en los cuales sólo el proveedor del enlace puede ayudarnos. Ciertos tipos de ataques necesitan de la participación de uno o más proveedores de Internet para contrarrestarlos, como son ciertos tipos de ataque DoS (Deny of Service) ante los que nuestro firewall podría quedar indefenso.

Sólo mediante el concurso de uno o más nodos de alto nivel del Internet se podría bloquear al atacante[12]. Por suerte los ataques más sofisticados de este tipo no son muy frecuentes.

### 17. No permitir conexiones directas desde la red interna a Internet

Asumimos que en Internet están los malos, y por tanto nuestra red interna no debería tomar contacto con ellos. Pero el Internet es irresistible e imprescindible para muchas personas de nuestra organización, por lo que debemos buscar algún mecanismo de protección.

Una de las soluciones más generalizadas a este problema la constituyen los sistemas denominados Proxy, en este caso, los usuarios de nuestra red local (a los que protegemos), se conectan aparentemente a Internet, pero realmente lo hacen hacia nuestro programa Proxy. La función de éste es básicamente conectarse a Internet en beneficio de los usuarios internos que lo solicitan. El resultado es que los posibles atacantes observarán al Proxy conectándose, pero no podrán acceder a la red interna. Obviamente nos aseguraremos que el Proxy esté bien seguro. Esto es más sencillo que cuidar a cada estación con usuarios incautos y sus sistemas operativos inseguros.

En resumen: cuando sea posible, debemos usar Proxy para dar acceso a Internet para la red interna. Además de ofrecer otras ventajas, como son: ahorro del ancho de banda y mayor velocidad de acceso a las páginas web más populares ( Proxy-caché).[13]

### 18. Uso de red perimétrica o zona desmilitarizada

Si tenemos servidores que dan cara a Internet, entonces deberá permitir ciertas conexiones del exterior a sus servidores, será problemático y por lo tanto debe ser controlado.

[12] Deny Of Service resources, http://www.denialinfo.com/; istributed Deny Of Service resources, http://staff.washington.edu/dittrich/misc/ddos/; Overview of Scans and DDos Attacks, http://www.nipc.gov/ddos.pdf

[13] La seguridad en Internet, http://www.cert.org/encyc_article/tocencyc.html

Muchos especialistas recomiendan mantener estos servidores protegidos mediante *firewalls*, pero a la vez separados de la red interna donde se encuentra, por ejemplo, nuestra base de datos.

La idea es que estos servidores están en una zona de peligrosidad intermedia protegidos de Internet, pero no al nivel de nuestro sistema más interno. Es por eso que se les suele asignar una red especial independiente denominada "perimétrica" o zona desmilitarizada (DMZ) con la intención de reducir la vulnerabilidad de nuestros datos más importantes mediante el alejamiento de los computadores que son blancos directo de las conexiones exteriores.

## 19. Prácticas de programación segura

Si la organización desarrolla software de cualquier tipo, y en especial, si este software dará cara a Internet, entonces debemos asegurarnos de que los desarrolladores conozcan las recomendaciones de seguridad correspondientes al lenguaje de programación o herramienta de desarrollo empleada así como del ambiente (sistema operativo o red) en que se ejecutan. De igual modo los diseñadores de la arquitectura deben considerar la seguridad de la misma desde el principio del proyecto. [14] Los programas que dan cara a Internet son un caso muy especial donde la seguridad es de suma importancia a juzgar por las malas experiencias producidas.[15]

## 20. Vigilancia

La vigilancia del buen funcionamiento del sistema es un asunto más complicado de lo que parece. El problema es que los ataques frecuentemente están disfrazados de conexiones más o menos válidas, y por otro lado, los sistemas de cómputo normalmente no avisan cuando son alterados, a no ser que ésto se haya establecido de antemano.

Los ataques generalmente tratan de aparentar que no ha ocurrido nada, a fin de conseguir hacer más y más modificaciones sin ser detectados y detenidos.

Todo ésto obliga a considerar de antemano ciertos indicadores que nos ayuden a determinar si un sistema ha sido comprometido o está en proceso de serlo, y en ciertos casos puede requerir de mucha sagacidad y un profundo conocimiento de los mecanismos internos del sistema, así como la activación de una mayor descripción de eventos.

Pero generalmente este análisis no se pone en práctica hasta que los síntomas aparecen y el ataque ya está muy avanzado, lo que ha motivado el incremento de productos que intentan adelantarse y dar aviso cuando algo anda mal.

[14] Shmoo: Cómo escribir código seguro, http://www.shmoo.com/securecode/; (Libro) John Viega and Gary McGraw: Building Secure Software, http://cseng.aw.com/book/0,3828,020172152X,00.html; [24] Buffer overflows:, http://www.enseirb.fr/~glaume/bof/report.html.
[15] IBM: Programación segura de CGI, http://www-106.ibm.com/developerworks/library/secure-cgi/

A estos sistemas se les conoce como sistemas de detección de intrusiones (IDS o NIDS.) El administrador precavido hará bien en considerarlos como parte de las medidas de seguridad.[16]

## 21. Establecimiento de políticas

Para terminar, una recomendación que en cierto modo engloba a todas las anteriores.

El establecimiento de políticas corresponde a un mecanismo que permite asegurar que la seguridad se mantenga en todas las situaciones y se deriva del "compromiso con la seguridad" de la organización. La idea detrás de todo esto es que nadie puede saber de antemano lo que piensa el administrador o el encargado de la seguridad sin ser informado.

Muchas organizaciones no lo tienen en cuenta y se da el caso en que un gerente se limita a reunir a los empleados y decirles "no hagan nada que pueda atentar contra la seguridad de la organización, o serán castigados…" El problema es que la gente normalmente no piensa en términos de seguridad sino en términos de cumplimiento de obligaciones y logro de resultados, y el camino más corto no siempre es el más seguro.

Las políticas justamente intentan abordar estos problemas mediante el establecimiento de normas que han de cumplir todos los miembros de la organización. Las políticas son documentos destinados a personas con responsabilidades claramente definidas y detallan cuidadosamente su alcance y aplicación. Explican tanto procedimientos informáticos como logísticos en términos de la jerarquía de la organización. [17]

Una ventaja colateral es la posibilidad de identificar responsables en caso de ataques exitosos (¡y así salvar su cabeza el administrador!) El cumplimiento de las políticas pasa por hacer tomar consciencia de su importancia a los destinatarios, para lo cual es imprescindible que su aplicación sea impulsada desde el más alto nivel jerárquico. Es recomendable, por ejemplo, que sea leída y firmada a modo de compromiso por parte del destinatario, y su difusión puede estar acompañada por charlas informativas por parte del administrador o del staff responsable.

La actualización responsable de las políticas es crítica para su cumplimiento, por lo que en ciertos casos se indica la designación de un responsable de las mismas o incluso de un equipo responsable, especialmente durante su elaboración y establecimiento inicial.

[16] Previniendo y detectando ataques con IDS's, http://online.securityfocus.com/infocus/1558; Snort: Network instrusion detection system, http://www.snort.org/, Tripwire: Host based IDS, http://www.tripwire.org/

[17] Recursos sobre políticas de seguridad, http://secinf.net/ipolicye.html

# ANEXO II. Políticas de Seguridad

Para lograr un efectivo control sobre todos los componentes que conforman la red y asegurar que su conectividad a otras redes no es algo frágil, es necesario primero que todo establecer con exactitud los recursos de la red y servicios que desea proteger, de tal manera que esté preparado para conectar nuestra red con el resto del mundo. Esto implica el estudio y definición de los aspectos necesarios para la planeación de la seguridad de la red, análisis de riesgos, identificación de recursos y amenazas, uso de la red y responsabilidades, planes de acción y contingencia.

Es importante tener una política de seguridad de red efectiva y bien pensada que pueda proteger la inversión y recursos de información de nuestra empresa. Sobre todo es importante que la organización defina claramente y valore que tan importantes son los recursos e información que se tienen en la red corporativa y dependiendo de ésto justificará si es necesario que se preste la atención y esfuerzos suficientes para lograr un nivel adecuado de protección.

**La Definición de Políticas de Seguridad**

Una Política de seguridad es un conjunto de leyes, reglas y prácticas que regulan como una organización maneja, protege y distribuye información sensitiva. Este documento se convierte en el primer paso para construir barreras de protección efectivas.

Una política de seguridad en un sitio es requerida para establecer a lo largo de una organización, un programa la interacción que hay entre los usuarios internos y externos con la red de computadoras de la empresa, como se implementará la arquitectura de la topología de red y donde se localizarán los puntos especiales de atención en cuanto a protección se refiere.

La definición de una política de seguridad de red no es algo en lo que se pueda establecer un orden lógico o secuencia aceptada de estados debido a que la seguridad es algo muy subjetivo, cada negocio tiene diferentes expectativas, diferentes metas, diferentes formas de valorar lo que va por su red, cada empresa tiene distintos requerimientos para almacenar , enviar y comunicar información de manera electrónica; por esto nunca existirá una política de seguridad aplicable a dos organizaciones diferentes.

Además, así como los negocios evolucionan para adaptarse a los cambios en las condiciones de mercado, la política de seguridad debe evolucionar para satisfacer las condiciones cambiantes de la tecnología.

Una política de seguridad de red efectiva es algo que todos los usuarios y administradores pueden aceptar y estar dispuestos a reforzar, siempre y cuando la política no disminuya la capacidad de la organización, es decir la política de seguridad debe ser de tal forma que no obstruya a los usuarios, y puedan cumplir con sus tareas de forma efectiva.

**Ventajas en el establecimiento de políticas de seguridad.**

Existen muchos factores que justifican el establecimiento de políticas de seguridad para un sitio específico, pero los más determinantes son:

- Ayudan a la organización a darle valor a la información.
- Es una infraestructura desde la cual otras estrategias de protección pueden ser desarrolladas
- Proveen unas claras y consistentes reglas para los usuarios de la red corporativa y su interacción con el entorno.
- Contribuyen a la efectividad y direccionan la protección total de la organización
- Pueden ayudar a responder ante requerimientos legales
- Ayudan a prevenir incidentes de seguridad.
- Proveen una guía cuando un incidente ocurre.
- Es una planeación estratégica del papel que juega la arquitectura de red al interior de la organización.
- Ayuda en la culturización de los usuarios para el uso de servicios de red e inculca el valor real que ellos representan.

**Características de las políticas de seguridad.**

Una política de seguridad es un plan elaborado de acuerdo con los objetivos generales de la organización y en el cual se ve reflejado el sentir corporativo acerca de los servicios de red y recursos que se desean proteger de manera efectiva y que representan activos importantes para el normal cumplimiento de la misión institucional. Por esto la política de seguridad debe cumplir con ciertas características propias de este tipo de planes como son:

- Debe ser simple y entendible (específica).
- Debe estar siempre disponible.
- Se puede aplicar en cualquier momento a la mayoría de las situaciones contempladas.
- Debe ser practicable y desarrollable.
- Se debe poder hacer cumplir.
- Debe ser consistente con otras políticas organizacionales.
- Debe ser estructurada.
- Se establece como una guía, no como una cadena de la cual se tenga que atar para siempre.
- Debe ser cambiante con la variación tecnológica.
- Una política debe considerar las necesidades y requerimientos de seguridad de todas las redes interconectadas como una unidad corporativa.

**Desarrollo de una política de seguridad**

El objetivo perseguido para desarrollar una política de seguridad de red oficial corporativa es definir las expectativas de la organización acerca del uso de la red y definir procedimientos

para prevenir y responder a incidentes de seguridad provenientes del cada día más avanzado mundo de la comunicación global.

Definir una política de seguridad de red significa desarrollar procedimientos y planes que salvaguardan los recursos de la red contra pérdidas y daños, por lo tanto es muy importante analizar entre otros los siguientes aspectos:

- Determinar los objetivos y directrices de la organización
- La política de seguridad debe estar acorde con otras políticas, reglas, regulaciones o leyes ya existentes en la organización, por lo tanto es necesario identificarlas y tenerlas en cuenta al momento de desarrollar la política de seguridad de redes.
- Identificación de los recursos disponibles.
- ¿Qué recursos se quieren proteger?
- ¿De quién se necesitan proteger los recursos?
- Identificación de posibles amenazas
- ¿Qué tan reales son las amenazas?
- ¿Qué tan importante es el recurso?
- ¿Qué medidas se puede implantar para proteger sus bienes de una manera económica y oportuna?
- Verificación frecuente de la política de seguridad de red para ver si los objetivos y circunstancias han cambiado.

En general el costo de proteger las redes de una amenaza debe ser menor que el costo de la recuperación, si es que se ve afectado por la amenaza de seguridad. La política de seguridad debe ser comunicada a cada quien que usa una computadora en la red con el fin de que sea ampliamente conocida y se pueda obtener una retroalimentación de los usuarios de la misma para efectos de revisiones periódicas y detección de nuevas amenazas o riesgos.

**Identificación de elementos a proteger**

El primer paso en la creación de la política de seguridad es crear una lista de todas las cosas que necesitan ser protegidas, esta lista debe ser regularmente actualizada.

Algunos elementos a considerar son:

- *Hardware: CPUs*, Tarjetas, teclados, terminales, estaciones de trabajo, PCs, impresoras, unidades de almacenamiento, líneas de comunicación, enrutadores, servidores de acceso remoto.
- *Software:* Programas fuentes, programas objeto, utilidades, programas de diagnóstico, sistemas operacionales, programas de comunicaciones.
- *Datos*: Durante la ejecución, almacenados en línea, almacenados fuera de línea, respaldos, rastreos de auditoría, bases de datos, en tránsito sobre los medios físicos de comunicación.
- *Personas*: Usuarios, personal externo con uso de servidores locales.
- *Documentació*n: de programas, hardware, procedimientos de administración loca.

- *Otros*: Medios magnéticos, papel, formas, etc.

**Análisis de riesgos**

El análisis de riesgos involucra la determinación de ¿qué se necesita proteger?, ¿contra que se necesita proteger?, ¿qué se necesita para protegerlo? Y ¿cómo protegerlo?. Es el proceso de examinar los posibles riesgos y clasificarlos por nivel de severidad, esto involucra hacer decisiones costo beneficio.

Algunos riesgos incluyen:

- Acceso no autorizado.
- Servicios no disponibles
- Descubrimiento de información sensitiva.

Como resultado de este proceso se obtiene un esquema para pesar el riesgo contra la importancia del recurso lo cual permitirá determinar cuanto esfuerzo se debe gastar en proteger el recurso.

**Definición de políticas de uso aceptable**

Los procedimientos antes descritos son una parte de la solución para el desarrollo de una política de seguridad pero no lo son en la totalidad, uno de los aspectos importantes para complementar es como los usuarios interactúan con la red y de allí se extraen elementos adicionales.

Algunos otros factores a analizar son:

- ¿A quien se le permite el uso de los recursos?
- ¿Cuál es el uso apropiado de para los recursos?
- ¿Quién está autorizado para garantizar acceso a aprobar el uso de recursos?
- ¿Quién puede tener privilegios para la administración del sistema?
- ¿Cuáles son los derechos y responsabilidades de los usuarios?
- ¿Cuáles son los derechos y responsabilidades de los administradores del sistema?
- ¿Qué es información altamente sensible?

**Auditoría y Revisión**

Es necesario contar con herramientas que ayuden a determinar si hay una violación a las políticas de seguridad, para ello se debe aprovechar al máximo las herramientas incluidos en los sistemas operacionales y utilidades de la red. La mayoría de sistemas operacionales cuentan con bastantes rastros o archivos de “log” con el fin de informar de la actividad del sistema, la examinación de estos “logs” son el primer paso efectivo para detectar el uso no autorizado del sistema.

Para tal efecto se pueden tomar acciones.

- ✧ Comparar listas de usuarios actualmente conectados con listas históricas, para detectar anomalías o comportamientos irregulares.
- ✧ Examinar las facilidades de "log" del sistema para verificar mensajes de error inusuales del software del sistema operacional como por ejemplo, un número grande de intentos fallidos de conexión a un usuario específico.
- ✧ Comparación de los procesos que están siendo ejecutados en las máquinas a tiempos diferentes pueden mostrar diferencias que lleven a detectar programas no autorizados o extraños en ejecución, quizás lanzados por intrusos.

**Comunicación a los usuarios**

La política de seguridad debe ser informada a todos los usuarios de la red para que conozcan acerca del uso apropiado que rige su acceso o estación de trabajo específica.

También debe ir acompañada por una campaña educacional que indique como se espera que sean usados todos los recursos involucrados en la red corporativa y como se pueden proteger por ellos mismos de accesos no autorizados.

Implantar una política de seguridad de red efectiva es un esfuerzo colectivo y como tal se deben proveer los medios para que los usuarios participen activamente en la definición de la misma y hagan aportes de lo que ellos mismos perciben de su interacción con la red.

Si los usuarios perciben que la política reduce su productividad, se debe permitir que participen. Si es necesario se pueden añadir recursos adicionales a la red para asegurar que los usuarios pueden continuar con su trabajo sin pérdida en la productividad. Para crear una política de seguridad de red efectiva es necesario encontrar un balance entre la protección y la productividad.

**Asegurar responsabilidades en torno a la política de seguridad**

Un aspecto importante en torno a la política de seguridad de red es asegurar que todos saben cuál es la responsabilidad para mantener la seguridad, por lo tanto la política debe poder garantizar que cada tipo de problema tiene a alguien que puede manejarlo de manera responsable. Así mismo pueden existir varios niveles de responsabilidad asociados con una política de seguridad de red. Por ejemplo cada usuario de la red está responsabilizado por su clave de acceso, un usuario que pone en riesgo su cuenta de acceso aumenta la probabilidad de comprometer otras cuentas y recursos. Por otro lado los administradores de red y de sistema son responsables de mantener la seguridad general de la red.

# ANEXO. III La Auditoría Informática

## 1. Planeación de la Auditoría en Informática

Para hacer una adecuada planeación de la auditoría en informática, hay que seguir una serie de pasos previos que permitirán dimensionar el tamaño y características de área dentro del organismo a auditar, sus sistemas, organización y equipo.

En el caso de la auditoría en informática, la planeación es fundamental, pues habrá que hacerla desde el punto de vista de los dos objetivos:

- Evaluación de los sistemas y procedimientos.
- Evaluación de los equipos de cómputo.

Para hacer una planeación eficaz, lo primero que se requiere es obtener información general sobre la organización y sobre la función de informática a evaluar, para ello es preciso hacer una investigación preliminar y algunas entrevistas previas, con base en esto planear el programa de trabajo, el cual deberá incluir tiempo, costo, personal necesario y documentos auxiliares a solicitar o formular durante el desarrollo de la misma.

### La investigación preliminar

Se deberá observar el estado general del área, su situación dentro de la organización, si existe la información solicitada, si es o no necesaria y la fecha de su última actualización.

Se debe hacer la investigación preliminar solicitando y revisando la información de cada una de las áreas basándose en los siguientes puntos:
Administración

Se recopila la información para obtener una visión general del departamento por medio de observaciones, entrevistas preliminares y solicitud de documentos para poder definir el objetivo y alcances del departamento.

☑ Para analizar y dimensionar la estructura por auditar se debe solicitar a nivel del área de informática:

### Objetivos

A corto y largo plazo.

### Recursos materiales y técnicos

Solicitar documentos sobre los equipos, número de ellos, localización y características.

- ✧ Estudios de viabilidad.
- ✧ Número de equipos, localización y las características (de los equipos instalados y por instalar y programados)

- ✧ Fechas de instalación de los equipos y planes de instalación.
- ✧ Contratos vigentes de compra, renta y servicio de mantenimiento.
- ✧ Contratos de seguros.
- ✧ Convenios que se tienen con otras instalaciones.
- ✧ Configuración de los equipos y capacidades actuales y máximas.
- ✧ Planes de expansión.
- ✧ Ubicación general de los equipos.
- ✧ Políticas de operación.
- ✧ Políticas de uso de los equipos.

**Sistemas**

Descripción general de los sistemas instalados y de los que estén por instalarse que contengan volúmenes de información.

- ✧ Manual de formas.
- ✧ Manual de procedimientos de los sistemas.
- ✧ Descripción genérica.
- ✧ Diagramas de entrada, archivos, salida.
- ✧ Salidas.
- ✧ Fecha de instalación de los sistemas.
- ✧ Proyecto de instalación de nuevos sistemas.

☑ En el momento de hacer la planeación de la auditoría o bien su realización, debemos evaluar que pueden presentarse las siguientes situaciones.

*Se solicita la información y se ve que:*

- ✧ No tiene y se necesita.
- ✧ No se tiene y no se necesita.

*Se tiene la información pero:*

- ✧ No se usa.
- ✧ Es incompleta.
- ✧ No esta actualizada.

- ✧ No es la adecuada.
- ✧ Se usa, está actualizada, es la adecuada y está completa.

En el caso de No se tiene y no se necesita, se debe evaluar la causa por la que no es necesaria. En el caso de No se tiene pero es necesaria, se debe recomendar que se elabore de acuerdo con las necesidades y con el uso que se le va a dar. En el caso de que se tenga la información pero no se utilice, se debe analizar por que no se usa. En caso de que se tenga la información, se debe analizar si se usa, si está actualizada, si es la adecuada y si está completa.

☑ El éxito del análisis crítico depende de las consideraciones siguientes

- ✧ Estudiar hechos y no opiniones (no se toman en cuenta los rumores ni la información sin fundamento)
- ✧ Investigar las causas, no los efectos.
- ✧ Atender razones, no excusas.
- ✧ No confiar en la memoria, preguntar constantemente.
- ✧ Criticar objetivamente y a fondo todos los informes y los datos recabados.

**Personal participante**

Una de las partes más importantes dentro de la planeación de la auditoría en informática es el personal que deberá participar y sus características.

Uno de los esquemas generalmente aceptados para tener un adecuado control es que el personal que intervengan esté debidamente capacitado, con alto sentido de moralidad, al cual se le exija la optimización de recursos (eficiencia) y se le retribuya o compense justamente por su trabajo.

Con estas bases se debe considerar las características de conocimientos, práctica profesional y capacitación que debe tener el personal que intervendrá en la auditoría. En primer lugar se debe pensar que hay personal asignado por la organización, con el suficiente nivel para poder coordinar el desarrollo de la auditoría, proporcionar toda la información que se solicite y programar las reuniones y entrevistas requeridas.

Éste es un punto muy importante ya que, de no tener el apoyo de la alta dirección, ni contar con un grupo multidisciplinario en el cual estén presentes una o varias personas del área a auditar, sería casi imposible obtener información en el momento y con las características deseadas.

También se debe contar con personas asignadas por los usuarios para que en el momento que se solicite información o bien se efectúe alguna entrevista de comprobación de hipótesis, nos proporcionen aquello que se esta solicitando, y complementen el grupo multidisciplinario,

ya que se debe analizar no sólo el punto de vista de la dirección de informática, sino también el del usuario del sistema.

☑ Para completar el grupo, como colaboradores directos en la realización de la auditoría se deben tener personas con las siguientes características:

- ✧ Técnico en informática.
- ✧ Experiencia en el área de informática.
- ✧ Experiencia en operación y análisis de sistemas.
- ✧ Conocimientos de los sistemas más importantes.

En caso de sistemas complejos se deberá contar con personal con conocimientos y experiencia en áreas específicas como base de datos, redes, etc.

Lo anterior no significa que una sola persona tenga los conocimientos y experiencias señaladas, pero si deben intervenir una o varias personas con las características apuntadas.

Una vez que se ha hecho la planeación, se puede utilizar el formato que aparece a continuación: "*Programa de Auditoría en Sistemas*", en él figura el organismo, las fases y subfases que comprenden la descripción de la actividad, el número de personas participantes, las fechas estimadas de inicio y terminación, el número de días hábiles y el número de días/hombre estimado,

**PROGRAMA DE AUDITORÍA EN SISTEMAS**

**HOJA NO._____ DE _____**

**INSTITUCIÓN: _____ FECHA DE FORMULACIÓN_____**

| fase | descripción | actividad | número de personal | período estimado | | Días hab est. | Días hom. est. |
|---|---|---|---|---|---|---|---|
| | | | participante | inicio | termino | | |
| | | | | | | | |

El control del avance de la auditoría lo podemos llevar mediante el *siguiente formato: "Avance del Cumplimiento del Programa de Auditoría en Sistemas"*, el cual nos permite cumplir con los procedimientos de control y asegurarnos que el trabajo se está llevando a cabo de acuerdo con el programa de auditoría, con los recursos estimados y en el tiempo señalado en la planeación.

**AVANCE DEL CUMPLIMIENTO DEL PROGRAMA DE AUDITORÍA EN SISTEMAS**

**HOJA No._____ DE _____**

**INSTITUCIÓN: _____ NUMERO_____ PERIODO QUE REPORTA_____**

| Fase | Situación de la auditoría | | | Periodo real de la auditoría | | Días reales utilizados | Grado de avance | Días hom. Est. | Explicación de las variaciones en relación con lo programado |
|---|---|---|---|---|---|---|---|---|---|
| | No iniciada | En proceso | Terminada | Iniciada | Terminada | | | | |
| | | | | | | | | | |

El hecho de contar con la información del avance nos permite revisar el trabajo elaborado por cualquiera de los asistentes. Como ejemplo de propuesta de auditoría en informática véase el siguiente:

**Propuesta de Servicios de Auditoría en Informática**

**I. ANTECEDENTES**
(Anotar los antecedentes específicos del proyecto de Auditoría)

**II. OBJETIVOS**
(Anotar el objetivo de la Auditoría)

**III. ALCANCES DEL PROYECTO**

El alcance del proyecto comprende:

***1.- Evaluación de la Dirección de Informática en lo que corresponde a:***

- Capacitación
- Planes de trabajo
- Controles
- Estándares

***2.- Evaluación de los Sistemas***

- Evaluación de los diferentes sistemas en operación (flujo de información, procedimientos, documentación, redundancia, organización de archivos, estándares de programación, controles, utilización de los sistemas)
- Evaluación del avance de los sistemas en desarrollo y congruencia con el diseño general
- Evaluación de prioridades y recursos asignados (humanos y equipos de cómputo)
- Seguridad física y lógica de los sistemas, su confidencialidad y respaldos

***3.- Evaluación de los equipos***

- Capacidades
- Utilización
- Nuevos Proyectos
- Seguridad física y lógica
- Evaluación física y lógica

## IV. METODOLOGIA

La metodología de investigación a utilizar en el proyecto se presenta a continuación:

1. Para la evaluación de la Dirección de Informática se llevarán a cabo las siguientes actividades:
   - Solicitud de los estándares utilizados y programa de trabajo
   - Aplicación del cuestionario al personal
   - Análisis y evaluación del a información
   - Elaboración del informe
2. Para la evaluación de los sistemas tanto en operación como en desarrollo se llevarán a cabo las siguientes actividades:
   - Solicitud del análisis y diseño del os sistemas en desarrollo y en operación
   - Solicitud de la documentación de los sistemas en operación (manuales técnicos, de operación del usuario, diseño de archivos y programas)
   - Recopilación y análisis de los procedimientos administrativos de cada sistema (flujo de información, formatos, reportes y consultas)
   - Análisis de llaves, redundancia, control, seguridad, confidencial y respaldos
   - Análisis del avance de los proyectos en desarrollo, prioridades y personal asignado

- Entrevista con los usuarios de los sistemas
- Evaluación directa de la información obtenida contra las necesidades y requerimientos del usuario
- Análisis objetivo de la estructuración y flujo de los programas
- Análisis y evaluación de la información recopilada
- Elaboración del informe

**3.** Para la evaluación de los equipos se levarán a cabo las siguientes actividades:

- Solicitud de los estudios de viabilidad y características de los equipos actuales, proyectos sobre ampliación de equipo, su actualización
- Solicitud de contratos de compra y mantenimientos de equipo y sistemas
- Solicitud de contratos y convenios de respaldo
- Solicitud de contratos de Seguros
- Elaboración de un cuestionario sobre la utilización de equipos, memoria, archivos, unidades de entrada/salida, equipos periféricos y su seguridad
- Visita técnica de comprobación de seguridad física y lógica de la instalaciones de la Dirección de Informática
- Evaluación técnica del sistema electrónico y ambiental de los equipos y del local utilizado
- Evaluación de la información recopilada, obtención de gráficas, porcentaje de utilización de los equipos y su justificación
- Elaboración y presentación del informe final ( conclusiones y recomendaciones)

## V. TIEMPO Y COSTO

**1.** (Poner el tiempo en que se llevará a cabo el proyecto, de preferencia indicando el tiempo de cada una de las etapas, costo del proyecto

## 2. Evaluación de Sistemas

La elaboración de sistemas debe ser evaluada con mucho detalle, para lo cual se debe revisar si existen realmente sistemas entrelazados como un todo o bien si existen programas aislados. Otro de los factores a evaluar es si existe un plan estratégico para la elaboración de los sistemas o si se están elaborados sin el adecuado señalamiento de prioridades y de objetivos.

☑ El plan estratégico deberá establecer los servicios que se presentarán en un futuro contestando preguntas como las siguientes:

- ✧ ¿Cuáles servicios se implementarán?
- ✧ ¿Cuándo se pondrán a disposición de los usuarios?
- ✧ ¿Qué características tendrán?
- ✧ ¿Cuántos recursos se requerirán?

☑ La estrategia de desarrollo deberá establecer las nuevas aplicaciones, recursos y la arquitectura en que estarán fundamentados:

- ✧ ¿Qué aplicaciones serán desarrolladas y cuando?
- ✧ ¿Qué tipo de archivos se utilizarán y cuando?
- ✧ ¿Qué bases de datos serán utilizarán y cuando?
- ✧ ¿Qué lenguajes se utilizarán y en que software?
- ✧ ¿Qué tecnología será utilizada y cuando se implementará?
- ✧ ¿Cuantos recursos se requerirán aproximadamente?
- ✧ ¿Cuál es aproximadamente el monto de la inversión en hardware y software?

☑ En lo referente a la consulta a los usuarios, el plan estratégico debe definir los requerimientos de información de la dependencia.

- ✧ ¿Qué estudios van a ser realizados al respecto?
- ✧ ¿Qué metodología se utilizará para dichos estudios?
- ✧ ¿Quién administrará y realizará dichos estudios?

En el área de auditoría interna debe evaluarse cuál ha sido la participación del auditor y los controles establecidos.

☑ Por último, el plan estratégico determina la planeación de los recursos.

- ✧ ¿Contempla el plan estratégico las ventajas de la nueva tecnología?
- ✧ ¿Cuál es la inversión requerida en servicios, desarrollo y consulta a los usuarios?

El proceso de planeación de sistemas deberá asegurarse de que todos los recursos requeridos estén claramente identificados en el plan de desarrollo de aplicaciones y datos. Estos recursos (hardware, software y comunicaciones) deberán ser compatibles con la arquitectura y la tecnología, conque se cuenta actualmente.

Los sistemas deben evaluarse de acuerdo con el ciclo de vida que normalmente siguen: requerimientos del usuario, estudio de factibilidad, diseño general, análisis, diseño lógico, desarrollo físico, pruebas, implementación, evaluación, modificaciones, instalación, mejoras.

Y se vuelve nuevamente al ciclo inicial, el cual a su vez debe comenzar con el de factibilidad.

La primera etapa a evaluar del sistema es el estudio de factibilidad, el cual debe analizar si el sistema es factible de realizarse, cuál es su relación costo/beneficio y si es recomendable elaborarlo.

Se deberá solicitar el estudio de factibilidad de los diferentes sistemas que se encuentren en operación, así como los que estén en la fase de análisis para evaluar si se considera la disponibilidad y características del equipo, los sistemas operativos y lenguajes disponibles, la necesidad de los usuarios, las formas de utilización de los sistemas, el costo y los beneficios que reportará el sistema, el efecto que producirá en quienes lo usarán y el efecto que éstos tendrán sobre el sistema y la congruencia de los diferentes sistemas.

En el caso de sistemas que estén funcionando, se deberá comprobar si existe el estudio de factibilidad con los puntos señalados y compararse con la realidad con lo especificado en el estudio de factibilidad

Por ejemplo en un sistema que el estudio de factibilidad señaló determinado costo y una serie de beneficios de acuerdo con las necesidades del usuario, debemos comparar cual fue su costo real y evaluar si se satisficieron las necesidades indicadas como beneficios del sistema.

Para investigar el costo de un sistema se debe considerar, con una exactitud razonable, el costo de los programas, el uso de los equipos (compilaciones, programas, pruebas, paralelos), tiempo, personal y operación, cosa que en la práctica son costos directos, indirectos y de operación.

Los beneficios que justifiquen el desarrollo de un sistema pueden ser el ahorro en los costos de operación, la reducción del tiempo de proceso de un sistema. Mayor exactitud, mejor servicio, una mejoría en los procedimientos de control, mayor confiabilidad y seguridad.

**3. Evaluación del Análisis**

En esta etapa se evaluarán las políticas, procedimientos y normas que se tienen para llevar a cabo el análisis.

☑ Se deberá evaluar la planeación de las aplicaciones que pueden provenir de tres fuentes principales:

- ✧ La planeación estratégica: agrupadas las aplicaciones en conjuntos relacionados entre sí y no como programas aislados. Las aplicaciones deben comprender todos los sistemas que puedan ser desarrollados en la dependencia, independientemente de los recursos que impliquen su desarrollo y justificación en el momento de la planeación.
- ✧ Los requerimientos de los usuarios.
- ✧ El inventario de sistemas en proceso al recopilar la información de los cambios que han sido solicitados, sin importar si se efectuaron o se registraron.

☑ La situación de una aplicación en dicho inventario puede ser alguna de las siguientes:

- ✧ Planeada para ser desarrollada en el futuro.
- ✧ En desarrollo.
- ✧ En proceso, pero con modificaciones en desarrollo.
- ✧ En proceso con problemas detectados.
- ✧ En proceso sin problemas.
- ✧ En proceso esporádicamente.

**Nota:** Se deberá documentar detalladamente la fuente que generó la necesidad de la aplicación. La primera parte será evaluar la forma en que se encuentran especificadas las políticas, los procedimientos y los estándares de análisis, si es que se cumplen y si son los adecuados para la dependencia.

Es importante revisar la situación en que se encuentran los manuales de análisis y si están acordes con las necesidades de la dependencia. En algunas ocasiones se tiene una microcomputadora, con sistemas sumamente sencillos y se solicita que se lleve a cabo una serie de análisis que después hay que plasmar en documentos señalados en los estándares, lo cual hace que esta fase sea muy compleja y costosa. Los sistemas y su documentación deben estar acordes con las características y necesidades de una dependencia específica.

Se debe evaluar la obtención de datos sobre la operación, flujo, nivel, jerarquía de la información que se tendrá a través del sistema. Se han de comparar los objetivos de los sistemas desarrollados con las operaciones actuales, para ver si el estudio de la ejecución deseada corresponde al actual.

La auditoría en sistemas debe evaluar los documentos y registros usados en la elaboración del sistema, así como todas las salidas y reportes, la descripción de las actividades de flujo de la información y de procedimientos, los archivos almacenados, su uso y su relación con otros archivos y sistemas, su frecuencia de acceso, su conservación, su seguridad y control, la documentación propuesta, las entradas y salidas del sistema y los documentos fuentes a usarse.

☑ Con la información obtenida podemos contestar a las siguientes preguntas:

- ✧ ¿Se está ejecutando en forma correcta y eficiente el proceso de información?
- ✧ ¿Puede ser simplificado para mejorar su aprovechamiento?
- ✧ ¿Se debe tener una mayor interacción con otros sistemas?
- ✧ ¿Se tiene propuesto un adecuado control y seguridad sobre el sistema?
- ✧ ¿Está en el análisis la documentación adecuada?

## 4. Evaluación del Diseño Lógico del Sistema

En esta etapa se deberán analizar las especificaciones del sistema.

¿Qué deberá hacer?, ¿Cómo lo deberá hacer?, ¿Secuencia y ocurrencia de los datos, el proceso y salida de reportes?

Una vez que hemos analizado estas partes, se deberá estudiar la participación que tuvo el usuario en la identificación del nuevo sistema, la participación de auditoría interna en el diseño de los controles y la determinación de los procedimientos de operación y decisión.

Al tener el análisis del diseño lógico del sistema debemos compararlo con lo que realmente se está obteniendo en la cual debemos evaluar lo planeado, cómo fue planeado y lo que realmente se está obteniendo.

☑ Los puntos a evaluar son:

- ✧ Entradas.
- ✧ Salidas.
- ✧ Procesos.
- ✧ Especificaciones de datos.
- ✧ Especificaciones de proceso.
- ✧ Métodos de acceso.
- ✧ Operaciones.
- ✧ Manipulación de datos (antes y después del proceso electrónico de datos).
- ✧ Proceso lógico necesario para producir informes.
- ✧ Identificación de archivos, tamaño de los campos y registros.
- ✧ Proceso en línea o lote y su justificación.
- ✧ Frecuencia y volúmenes de operación.
- ✧ Sistemas de seguridad.
- ✧ Sistemas de control.
- ✧ Responsables.
- ✧ Número de usuarios.

☑ Dentro del estudio de los sistemas en uso se deberá solicitar:

- ✧ Manual del usuario.
- ✧ Descripción de flujo de información y/o procesos.

- ✧ Descripción y distribución de información.
- ✧ Manual de formas.
- ✧ Manual de reportes.
- ✧ Lista de archivos y especificaciones.

☑ Lo que se debe determinar en el sistema:

*En el procedimiento:*

- ✧ ¿Quién hace, cuando y como?
- ✧ ¿Qué formas se utilizan en el sistema?
- ✧ ¿Son necesarias, se usan, están duplicadas?
- ✧ ¿El número de copias es el adecuado?
- ✧ ¿Existen puntos de control o faltan?

*En la gráfica de flujo de información:*

- ✧ ¿Es fácil de usar?
- ✧ ¿Es lógica?
- ✧ ¿Se encontraron lagunas?
- ✧ ¿Hay faltas de control?

*En el diseño:*

- ✧ ¿Cómo se usará la herramienta de diseño si existe?
- ✧ ¿Qué también se ajusta la herramienta al procedimiento?

**5. Evaluación del Desarrollo del Sistema**

En esta etapa del sistema se deberán auditar los programas, su diseño, el leguaje utilizado, interconexión entre los programas y características del hardware empleado (total o parcial) para el desarrollo del sistema. Al evaluar un sistema de información se tendrá presente que todo sistema debe proporcionar información para planear, organizar y controlar de manera eficaz y oportuna, para reducir la duplicidad de datos y de reportes y obtener una mayor seguridad en la forma más económica posible. De ese modo contará con los mejores elementos para una adecuada toma de decisiones. Al tener un proceso distribuido, es preciso considerar la seguridad del movimiento de la información entre nodos.

El proceso de planeación de sistemas debe definir la red óptima de comunicaciones, los tipos de mensajes requeridos, el tráfico esperado en las líneas de comunicación y otros factores

que afectan el diseño. Es importante considerar las variables que afectan a un sistema: ubicación en los niveles de la organización, el tamaño y los recursos que utiliza.

☑ Las características que deben evaluarse en los sistemas son:

- ✧ Dinámicos (susceptibles de modificarse).
- ✧ Estructurados (las interacciones de sus componentes o subsistemas deben actuar como un todo)
- ✧ Integrados (un solo objetivo). En él habrá sistemas que puedan ser interrelacionados y no programas aislados.
- ✧ Accesibles (que estén disponibles).
- ✧ Necesarios (que se pruebe su utilización).
- ✧ Comprensibles (que contengan todos los atributos).
- ✧ Oportunos (que esté la información en el momento que se requiere).
- ✧ Funcionales (que proporcionen la información adecuada a cada nivel).
- ✧ Estándar (que la información tenga la misma interpretación en los distintos niveles).
- ✧ Modulares (facilidad para ser expandidos o reducidos).
- ✧ Jerárquicos (por niveles funcionales).
- ✧ Seguros (que sólo las personas autorizadas tengan acceso).
- ✧ Únicos (que no duplique información).

## 6. Control de Proyectos

Debido a las características propias del análisis y la programación, es muy frecuente que la implantación de los sistemas se retrase y se llegue a suceder que una persona lleva trabajando varios años dentro de un sistema o bien que se presenten irregularidades en las que los programadores se ponen a realizar actividades ajenas a la dirección de informática. Para poder controlar el avance de los sistemas, ya que ésta es una actividad de difícil evaluación, se recomienda que se utilice la técnica de administración por proyectos para su adecuado control.

Para tener una buena administración por proyectos se requiere que el analista o el programador y su jefe inmediato elaboren un plan de trabajo en el cual se especifiquen actividades, metas, personal participante y tiempos. Este plan debe ser revisado periódicamente (semanal, mensual, etc.) para evaluar el avance respecto a lo programado.

La estructura estándar de la planeación de proyectos deberá incluir la facilidad de asignar fechas predefinidas de terminación de cada tarea. Dentro de estas fechas debe estar el calendario de reuniones de revisión, las cuales tendrán diferentes niveles de detalle.

## CUESTIONARIO

1. ¿Existe una lista de proyectos de sistema de procedimiento de información y fechas programadas de implantación que puedan ser considerados como plan maestro?
2. ¿Está relacionado el plan maestro con un plan general de desarrollo de la dependencia?
3. ¿Ofrece el plan maestro la atención de solicitudes urgentes de los usuarios?
4. ¿Asigna el plan maestro un porcentaje del tiempo total de producción al reproceso o fallas de equipo?
5. Escribir la lista de proyectos a corto plazo y largo plazo.
6. Escribir una lista de sistemas en proceso periodicidad y usuarios.
7. ¿Quién autoriza los proyectos?
8. ¿Cómo se asignan los recursos?
9. ¿Cómo se estiman los tiempos de duración?
10. ¿Quién interviene en la planeación de los proyectos?
11. ¿Cómo se calcula el presupuesto del proyecto?
12. ¿Qué técnicas se usan en el control de los proyectos?
13. ¿Quién asigna las prioridades?
14. ¿Cómo se asignan las prioridades?
15. ¿Cómo se controla el avance del proyecto?
16. ¿Con qué periodicidad se revisa el reporte de avance del proyecto?
17. ¿Cómo se estima el rendimiento del personal?
18. ¿Con que frecuencia se estiman los costos del proyecto para compararlo con lo presupuestado?
19. ¿Qué acciones correctivas se toman en caso de desviaciones?
20. ¿Qué pasos y técnicas siguen en la planeación y control de los proyectos?
21. Enumérelos secuencialmente.
    - ☐ Determinación de los objetivos.
    - ☐ Señalamiento de las políticas.
    - ☐ Designación del funcionario responsable del proyecto.
    - ☐ Integración del grupo de trabajo.
    - ☐ Integración de un comité de decisiones.
    - ☐ Desarrollo de la investigación.
    - ☐ Documentación de la investigación.

- ☐ Factibilidad de los sistemas.
- ☐ Análisis y valuación de propuestas.
- ☐ Selección de equipos.

22. ¿Se llevan a cabo revisiones periódicas de los sistemas para determinar si aún cumplen con los objetivos para los cuales fueron diseñados?

| | | |
|---|---|---|
| De análisis | SÍ ( ) | NO ( ) |
| De programación | SÍ ( ) | NO ( ) |
| Observaciones | | |

23. Incluir el plazo estimado de acuerdo con los proyectos que se tienen en que el departamento de informática podría satisfacer las necesidades de la dependencia, según la situación actual.

## 7. Control de Diseño de Sistemas y Programación

El objetivo es asegurarse de que el sistema funcione conforme a las especificaciones funcionales, a fin de que el usuario tenga la suficiente información para su manejo, operación y aceptación.

☑ Las revisiones se efectúan en forma paralela desde el análisis hasta la programación y sus objetivos son los siguientes:

- ✧ *Etapa de análisis* Identificar inexactitudes, ambigüedades y omisiones en las especificaciones.
- ✧ *Etapa de diseño* Descubrir errores, debilidades, omisiones antes de iniciar la codificación.
- ✧ *Etapa de programación* Buscar la claridad, modularidad y verificar con base en las especificaciones.

Esta actividad es muy importante ya que el costo de corregir errores es directamente proporcional al momento que se detectan: si se descubren en el momento de programación será más alto que si se detecta en la etapa de análisis. Esta función tiene una gran importancia en el ciclo de evaluación de aplicaciones de los sistemas de información y busca comprobar que la aplicación cumple las especificaciones del usuario, que se haya desarrollado dentro de lo presupuestado, que tenga los controles necesarios y que efectivamente cumpla con los objetivos y beneficios esperados.

El siguiente cuestionario se presenta como ejemplo para la evaluación del diseño y prueba de los sistemas:

## CUESTIONARIO

1. ¿Quiénes intervienen al diseñar un sistema?
   - ☐ Usuario.
   - ☐ Analista.
   - ☐ Programadores.
   - ☐ Operadores.
   - ☐ Gerente de departamento.
   - ☐ Auditores internos.
   - ☐ Asesores.
   - ☐ Otros.
2. ¿Los analistas son también programadores?

   SÍ ( ) NO ( )
3. ¿Qué lenguaje o lenguajes conocen los analistas?
4. ¿Cuántos analistas hay y qué experiencia tienen?
5. ¿Qué lenguaje conocen los programadores?
6. ¿Cómo se controla el trabajo de los analistas?
7. ¿Cómo se controla el trabajo de los programadores?
8. Indique qué pasos siguen los programadores en el desarrollo de un programa:
   - ☐ Estudio de la definición
   - ☐ Discusión con el analista
   - ☐ Diagrama de bloques
   - ☐ Tabla de decisiones
   - ☐ Prueba de escritorio
   - ☐ Codificación
   - ☐ ¿Es enviado a captura o los programadores capturan?
   - ☐ ¿Quién los captura?_____
   - ☐ Compilación
   - ☐ laborar datos de prueba
   - ☐ Solicitar datos al analista
   - ☐ Correr programas con datos
   - ☐ Revisión de resultados

- ☐ Corrección del programa
- ☐ Documentar el programa
- ☐ Someter resultados de prueba
- ☐ Entrega del programa

9. ¿Qué documentación acompaña al programa cuando se entrega?

_____
_____
_____
_____
_____
_____
_____

Difícilmente se controla realmente el flujo de la información de un sistema que desde su inicio ha sido mal analizado, mal diseñado, mal programado e incluso mal documentado.

El excesivo mantenimiento de los sistemas generalmente ocasionado por un mal desarrollo, se inicia desde que el usuario establece sus requerimientos (en ocasiones sin saber qué desea) hasta la instalación del mismo, sin que se haya establecido un plan de prueba del sistema para medir su grado de confiabilidad en la operación que efectuará.

Para verificar si existe esta situación, se debe pedir a los analistas y a los programadores las actividades que están desarrollando en el momento de la auditoría y evaluar si están efectuando actividades de mantenimiento o de realización de nuevos proyectos. En ambos casos se deberá evaluar el tiempo que llevan dentro del mismo sistema, la prioridad que se le asignó y cómo está en el tiempo real en relación al tiempo estimado en el plan maestro.

### 8. Instructivos de Operación

Se debe evaluar los instructivos de operación de los sistemas para evitar que los programadores tengan acceso a los sistemas en operación, y el contenido mínimo de los instructivos de operación se puedan verificar mediante el siguiente cuestionario.

☑ El instructivo de operación deberá comprender.

- ✧ Diagrama de flujo por cada programa.
- ✧ Diagrama particular de entrada/salida
- ✧ Mensaje y su explicación

- ✧ Parámetros y su explicación
- ✧ Diseño de impresión de resultados
- ✧ Cifras de control
- ✧ Fórmulas de verificación
- ✧ Observaciones
- ✧ Instrucciones en caso de error
- ✧ Calendario de proceso y resultados

**9. Forma de Implementación**

La finalidad de evaluar los trabajos que se realizan para iniciar la operación de un sistema, esto es, la prueba integral del sistema, adecuación, aceptación por parte del usuario, entrenamiento de los responsables del sistema etc.

☑ Los puntos a tomar en cuenta para la prueba de un sistema son:

- ✧ Prueba particular de cada programa
- ✧ Prueba por fase validación, actualización
- ✧ Prueba integral del paralelo
- ✧ Prueba en paralelo sistema

**10. Entrevistas a Usuarios**

La entrevista se deberá llevar a cabo para comprobar datos proporcionados y la situación de la dependencia en el departamento de Sistemas de Información.

Su objeto es conocer la opinión que tienen los usuarios sobre los servicios proporcionados, así como la difusión de las aplicaciones de la computadora y de los sistemas en operación.

Las entrevistas se deberán hacer, en caso de ser posible, a todos los usuarios o bien en forma aleatoria a algunos de los usuarios, tanto de los más importantes como de los de menor importancia, en cuanto al uso del equipo.

☑ Desde el punto de vista del usuario los sistemas deben:

- ✧ Cumplir con los requerimientos totales del usuario.
- ✧ Cubrir todos los controles necesarios.
- ✧ No exceder las estimaciones del presupuesto inicial.
- ✧ Serán fácilmente modificables.

Para que un sistema cumpla con los requerimientos del usuario, se necesita una comunicación completa entre usuarios y responsable del desarrollo del sistema.

En esta misma etapa debió haberse definido la calidad de la información que será procesada por la computadora, estableciéndose los riesgos de la misma y la forma de minimizarlos. Para ello se debieron definir los controles adecuados, estableciéndose además los niveles de acceso a la información, es decir, quién tiene privilegios de consulta, modificar o incluso borrar información.

Esta etapa habrá de ser cuidadosamente verificada por el auditor interno especialista en sistemas y por el auditor en informática, para comprobar que se logro una adecuada comprensión de los requerimientos del usuario y un control satisfactorio de información.

☑ Para verificar si los servicios que se proporcionan a los usuarios son los requeridos y se están proporcionando en forma adecuada, cuando menos será preciso considerar la siguiente información.

- ✧ Descripción de los servicios prestados.
- ✧ Criterios de evaluación que utilizan los usuarios para evaluar el nivel del servicio prestado.
- ✧ Reporte periódico del uso y concepto del usuario sobre el servicio.
- ✧ Registro de los requerimientos planteados por el usuario.

Con esta información se puede comenzar a realizar la entrevista para determinar si los servicios proporcionados y planeados por la dirección de Informática cubren las necesidades de información de las dependencias.

A continuación se presenta una guía de cuestionario para aplicarse durante la entrevista con el usuario.

## CUESTIONARIO

1. ¿Considera que el Departamento de Sistemas de Información de los resultados esperados?

   Si ( )  No ( )

   ¿Por qué? _____

2. ¿Cómo considera usted, en general, el servicio proporcionado por el Departamento de Sistemas de Información?

   ☐ Deficiente

   ☐ Aceptable

- ☐ Satisfactorio
- ☐ Excelente

¿Por qué? _____

3. ¿Cubre sus necesidades el sistema que utiliza el departamento de cómputo?
   - ☐ No las cubre
   - ☐ Parcialmente
   - ☐ La mayor parte
   - ☐ Todas

   ¿Por qué? _____
4. ¿Hay disponibilidad del departamento de cómputo para sus requerimientos?
   - ☐ Generalmente no existe
   - ☐ Hay ocasionalmente
   - ☐ Regularmente
   - ☐ Siempre

   ¿Por qué? _____
5. ¿Son entregados con puntualidad los trabajos?
   - ☐ Nunca
   - ☐ Rara vez
   - ☐ Ocasionalmente
   - ☐ Generalmente
   - ☐ Siempre

   ¿Por qué? _____
6. ¿Que piensa de la presentación de los trabajadores solicitados al departamento de cómputo?
   - ☐ Deficiente ( )
   - ☐ Aceptable ( )
   - ☐ Satisfactorio ( )
   - ☐ Excelente ( )

   ¿Por qué? _____
7. ¿Que piensa de la asesoría que se imparte sobre informática?
   - ☐ No se proporciona ( )
   - ☐ Es insuficiente ( )

- ☐ Satisfactoria ( )
- ☐ Excelente ( )

¿Por qué? _____

8. ¿Que piensa de la seguridad en el manejo de la información proporcionada por el sistema que utiliza?
   - ☐ Nula ( )
   - ☐ Riesgosa ( )
   - ☐ Satisfactoria ( )
   - ☐ Excelente ( )
   - ☐ Lo desconoce ( )

   ¿Por qué? _____
9. ¿Existen fallas de exactitud en los procesos de información?

   ¿Cuáles? _____
10. ¿Cómo utiliza los reportes que se le proporcionan? _____
11. ¿Cuáles no Utiliza? _____
12. De aquellos que no utiliza ¿por que razón los recibe? _____

13. ¿Que sugerencias presenta en cuanto a la eliminación de reportes modificación, fusión, división de reporte?

_____
_____
_____
_____

14. ¿Se cuenta con un manual de usuario por Sistema?

SI ( ) NO ( )

15. ¿Es claro y objetivo el manual del usuario?

SI ( ) NO ( )

16. ¿Que opinión tiene el manual? **NOTA:** Pida el manual del usuario para evaluarlo

_____
_____
_____

17. ¿Quién interviene de su departamento en el diseño de sistemas?

_____
_____
_____

18. ¿Que sistemas desearía que se incluyeran?

_____
_____
_____

19. Observaciones:

_____
_____
_____

## 11. Controles

Los datos son uno de los recursos más valiosos de las organizaciones y, aunque son intangibles, necesitan ser controlados y auditados con el mismo cuidado que los demás inventarios de la organización, por lo cual se debe tener presente:

a) La responsabilidad de los datos es compartida conjuntamente por alguna función determinada y el departamento de cómputo.
b) Un problema de dependencia que se debe considerar es el que se origina por la duplicidad de los datos y consiste en poder determinar los propietarios o usuarios posibles (principalmente en el caso de redes y banco de datos) y la responsabilidad de su actualización y consistencia.
c) Los datos deberán tener una clasificación estándar y un mecanismo de identificación que permita detectar duplicidad y redundancia dentro de una aplicación y de todas las aplicaciones en general.
d) Se deben relacionar los elementos de los datos con las bases de datos donde están almacenados, así como los reportes y grupos de procesos donde son generados.

### Control de los datos fuente y manejo cifras de control

La mayoría de los Delitos por computadora son cometidos por modificaciones de datos fuente al:

- ✧ Suprimir u omitir datos.

- ✧ Adicionar Datos.
- ✧ Alterar datos.
- ✧ Duplicar procesos.

Esto es de suma importancia en caso de equipos de cómputo que cuentan con sistemas en línea, en los que los usuarios son los responsables de la captura y modificación de la información al tener un adecuado control con señalamiento de responsables de los datos (uno de los usuarios debe ser el único responsable de determinado dato), con claves de acceso de acuerdo a niveles.

El primer nivel es el que puede hacer únicamente consultas. El segundo nivel es aquel que puede hacer captura, modificaciones y consultas y el tercer nivel es el que solo puede hacer todos lo anterior y además puede realizar bajas.

**NOTA**: Debido a que se denomina de diferentes formas la actividad de transcribir la información del dato fuente a la computadora, en el presente trabajo se le denominará captura o captación considerándola como sinónimo de digitalizar (capturista, digitalizadora).

☑ Lo primero que se debe evaluar es la entrada de la información y que se tengan las cifras de control necesarias para determinar la veracidad de la información, para lo cual se puede utilizar el siguiente cuestionario:

## CUESTIONARIO

1. Indique el porcentaje de datos que se reciben en el área de captación. _____%
2. Indique el contenido de la orden de trabajo que se recibe en el área de captación de datos:
   - ☐ Número de folio
   - ☐ Número(s) de formato(s)
   - ☐ Fecha y hora de Recepción
   - ☐ Nombre Depto.
   - ☐ Nombre del documento
   - ☐ Nombre responsable
   - ☐ Volumen aproximado de registro
   - ☐ Clave de cargo (Número de cuenta)
   - ☐ Número de registros
   - ☐ Fecha y hora de entrega de doctos y registros captados
   - ☐ Clave del capturista

- ☐ Fecha estimada de entrega

3. Indique cuál(es) control(es) interno(s) existe(n) en el área de captación de datos:

- ☐ Firmas de autorización ( )
- ☐ Recepción de trabajos ( )
- ☐ Control de trabajos atrasados ( )
- ☐ Revisión del documento ( )
- ☐ Avance de trabajos ( )
- ☐ fuente (legibilidad, verificación de datos completos, etc.) ( )
- ☐ Prioridades de captación ( )
- ☐ Errores por trabajo ( )
- ☐ Producción de trabajo ( )
- ☐ Corrección de errores ( )
- ☐ Producción de cada operador
- ☐ Entrega de trabajos
- ☐ Verificación de cifras de control de entrada con las de salida.
- ☐ Costo Mensual por trabajo

4. ¿Existe un programa de trabajo de captación de datos?

¿Se elabora ese programa para cada turno?

- ☐ Diariamente
- ☐ Semanalmente
- ☐ Mensualmente

La elaboración del programa de trabajos se hace:

- ☐ Internamente
- ☐ Se les señalan a los usuarios las prioridades

c) ¿Que acción(es) se toma(n) si el trabajo programado no se recibe a tiempo?

_____
_____
_____

5. ¿Quién controla las entradas de documentos fuente? _____

6. ¿En que forma las controla? _____

7. ¿Que cifras de control se obtienen?

| Sistema | Cifras que se Obtienen | Observaciones |
|---|---|---|
| | | |

8. ¿Qué documento de entrada se tienen?

| Sistemas Documentos | Depto. que proporciona el documento | periodicidad Observaciones |
|---|---|---|
| | | |

9. ¿Se anota que persona recibe la información y su volumen?

SI ( ) NO ( )

10. ¿Se anota a que capturista se entrega la información, el volumen y la hora?

SI ( ) NO ( )

11. ¿Se verifica la cantidad de la información recibida para su captura?

SI ( ) NO ( )

12. ¿Se revisan las cifras de control antes de enviarlas a captura?

SI ( ) NO ( )

13. ¿Para aquellos procesos que no traigan cifras de control se ha establecido criterios a fin de asegurar que la información es completa y valida?

SI ( ) NO ( )

14. ¿Existe un procedimiento escrito que indique como tratar la información inválida (sin firma ilegible, no corresponden las cifras de control)?

_____
_____
_____

15. En caso de resguardo de información de entrada en sistemas, ¿Se custodian en un lugar seguro?

_____
_____
_____

16. Si se queda en el departamento de sistemas, ¿Por cuanto tiempo se guarda?

_____
_____
_____

17. ¿Existe un registro de anomalías en la información debido a mala codificación?

_____
_____
_____

18. ¿Existe una relación completa de distribución de listados, en la cual se indiquen personas, secuencia y sistemas a los que pertenecen?

_____
_____
_____

19. ¿Se verifica que las cifras de las validaciones concuerden con los documentos de entrada?

_____
_____
_____

20. ¿Se hace una relación de cuando y a quién fueron distribuidos los listados?

_____
_____
_____

21. ¿Se controlan separadamente los documentos confidenciales?

_____
_____
_____

22. ¿Se aprovecha adecuadamente el papel de los listados inservibles?

_____
_____
_____

23. ¿Existe un registro de los documentos que entran a capturar?

    _____
    _____
    _____

24. ¿Se hace un reporte diario, semanal o mensual de captura?

    _____
    _____
    _____

25. ¿Se hace un reporte diario, semanal o mensual de anomalías en la información de entrada?

    _____
    _____
    _____

26. ¿Se lleva un control de la producción por persona?

    _____
    _____
    _____

27. ¿Quién revisa este control?

    _____
    _____
    _____

28. ¿Existen instrucciones escritas para capturar cada aplicación o, en su defecto existe una relación de programas?

    _____
    _____
    _____

**Control de operación**

La eficiencia y el costo de la operación de un sistema de cómputo se ven fuertemente afectados por la calidad e integridad de la documentación requerida para el proceso en la computadora.

El objetivo del presente ejemplo de cuestionario es señalar los procedimientos e instructivos formales de operación, analizar su estandarización y evaluar el cumplimiento de los mismos.

## CUESTIONARIO

1. ¿Existen procedimientos formales para la operación del sistema de computo?

SI ( ) NO ( )

2. ¿Están actualizados los procedimientos?

SI ( ) NO ( )

3. Indique la periodicidad de la actualización de los procedimientos:

- ☐ Semestral
- ☐ Anual
- ☐ Cada vez que haya cambio de equipo

4. Indique el contenido de los instructivos de operación para cada aplicación:

- ☐ Identificación del sistema
- ☐ Identificación del programa
- ☐ Periodicidad y duración de la corrida
- ☐ Especificación de formas especiales
- ☐ Especificación de cintas de impresoras
- ☐ Etiquetas de archivos de salida, nombre,
- ☐ archivo lógico, y fechas de creación y expiración
- ☐ Instructivo sobre materiales de entrada y salida
- ☐ Altos programados y la acciones requeridas
- ☐ Instructivos específicos
- ☐ a los operadores en caso de falla del equipo
- ☐ Instructivos de reinicio
- ☐ Procedimientos de recuperación para proceso de
- ☐ gran duración o criterios
- ☐ Identificación de todos los
- ☐ dispositivos de la máquina a ser usados
- ☐ Especificaciones de resultados
- ☐ (cifras de control, registros de salida por archivo, etc.)

5. ¿Existen órdenes de proceso para cada corrida en la computadora (incluyendo pruebas, compilaciones y producción)?

SI ( ) NO ( )

6. ¿Son suficientemente claras para los operadores estas órdenes?

SI ( ) NO ( )

7. ¿Existe una estandarización de las ordenes de proceso?

SI ( ) NO ( )

8. ¿Existe un control que asegure la justificación de los procesos en el computador? (Que los procesos que se están autorizados y tengan una razón de ser procesados.

SI ( ) NO ( )

9. ¿Cómo programan los operadores los trabajos dentro del departamento de cómputo?

- ☐ Primero que entra, primero que sale
- ☐ se respetan las prioridades,
- ☐ Otra (especifique) _____

10. ¿Los retrasos o incumplimiento con el programa de operación diaria, se revisa y analiza?

SI ( ) NO ( )

11. ¿Quién revisa este reporte en su caso?

_____
_____
_____

12. Analice la eficiencia con que se ejecutan los trabajos dentro del departamento de cómputo, tomando en cuenta equipo y operador, a través de inspección visual, y describa sus observaciones.

_____
_____
_____

13. ¿Existen procedimientos escritos para la recuperación del sistema en caso de falla?

_____
_____
_____

14. ¿Cómo se actúa en caso de errores?

_____
_____
_____

15. ¿Existen instrucciones especificas para cada proceso, con las indicaciones pertinentes?

_____
_____
_____

16. ¿Se tienen procedimientos específicos que indiquen al operador que hacer cuando un programa interrumpe su ejecución u otras dificultades en proceso?

_____
_____
_____

17. ¿Puede el operador modificar los datos de entrada?

_____
_____
_____

18. ¿Se prohíbe a analistas y programadores la operación del sistema que programo o analizo?

_____
_____
_____

19. ¿Se prohíbe al operador modificar información de archivos o bibliotecas de programas?

_____
_____
_____

20. ¿El operador realiza funciones de mantenimiento diario en dispositivos que así lo requieran?

_____
_____
_____

21. ¿Las intervenciones de los operadores?:

| | | |
|---|---|---|
| Son muy numerosas | SI ( ) | NO ( ) |
| Se limitan los mensajes esenciales | SI ( ) | NO ( ) |

Otras (especifique) _____

22. ¿Se tiene un control adecuado sobre los sistemas y programas que están en operación?

SI ( ) NO ( )

23. ¿Cómo controlan los trabajos dentro del departamento de cómputo?

_____
_____
_____

24. ¿Se rota al personal de control de información con los operadores procurando un entrenamiento cruzado y evitando la manipulación fraudulenta de datos?

SI ( ) NO ( )

25. ¿Cuentan los operadores con una bitácora para mantener registros de cualquier evento y acción tomada por ellos?

- ☐ Si
- ☐ por máquina
- ☐ escrita manualmente
- ☐ NO

26. Verificar que exista un registro de funcionamiento que muestre el tiempo de paros y mantenimiento o instalaciones de software.

Observaciones_____
_____
_____

27. ¿Existen procedimientos para evitar las corridas de programas no autorizados?

SI ( ) NO ( )

28. ¿Existe un plan definido para el cambio de turno de operaciones que evite el descontrol y discontinuidad de la operación?

_____
_____
_____

29. Verificar que sea razonable el plan para coordinar el cambio de turno.

Observaciones_____
_____
_____

30. ¿Se hacen inspecciones periódicas de muestreo?

SI ( ) NO ( )

31. Enuncie los procedimientos mencionados en el inciso anterior:

_____
_____
_____

32. ¿Se permite a los operadores el acceso a los diagramas de flujo, programas fuente, etc. fuera del departamento de cómputo?

SI ( ) NO ( )

33. ¿Se controla estrictamente el acceso a la documentación de programas o de aplicaciones rutinarias?

SI ( ) NO ( )

¿Cómo?_____

34. Verifique que los privilegios del operador se restrinjan a aquellos que le son asignados a la clasificación de seguridad de operador.

Observaciones_____
_____
_____

35. ¿Existen procedimientos formales que se deban observar antes de que sean aceptados en operación, sistemas nuevos o modificaciones a los mismos?

SI ( ) NO ( )

36. ¿Estos procedimientos incluyen corridas en paralelo de los sistemas modificados con las versiones anteriores?

SI ( ) NO ( )

37. ¿Durante cuanto tiempo? _____

38. ¿Que precauciones se toman durante el periodo de implantación?

_____
_____
_____

39. ¿Quién da la aprobación formal cuando las corridas de prueba de un sistema modificado o nuevo están acordes con los instructivos de operación?

_____
_____
_____

40. ¿Se catalogan los programas liberados para producción rutinaria?

SI ( ) NO ( )

41. Mencione que instructivos se proporcionan a las personas que intervienen en la operación rutinaria de un sistema.

_____
_____
_____

42. Indique que tipo de controles tiene sobre los archivos magnéticos de los archivos de datos, que aseguren la utilización de los datos precisos en los procesos correspondientes.

_____
_____
_____

43. ¿Existe un lugar para archivar las bitácoras del sistema del equipo de cómputo?

SI ( ) NO ( )

44. Indique como está organizado este archivo de bitácora.

- ☐ Por fecha
- ☐ por fecha y hora
- ☐ por turno de operación
- ☐ Otros

45. ¿Cuál es la utilización sistemática de las bitácoras?

_____
_____
_____

46. ¿Además de las mencionadas anteriormente, que otras funciones o áreas se encuentran en el departamento de cómputo actualmente?

_____
_____
_____

47. Verifique que se lleve un registro de utilización del equipo diario, sistemas en línea y *batch*, de tal manera que se pueda medir la eficiencia del uso de equipo.

Observaciones_____
_____
_____

48. ¿Se tiene inventario actualizado de los equipos y terminales con su localización?

SI ( ) NO ( )

49. ¿Cómo se controlan los procesos en línea?

_____
_____
_____

50. ¿Se tienen seguros sobre todos los equipos?

SI ( ) NO ( )

51. ¿Conque compañía?

_____
_____
_____

Solicitar pólizas de seguros y verificar tipo de seguro y montos.

Observaciones_____
_____
_____

52. ¿Cómo se controlan las llaves de acceso (Password)?.

_____
_____
_____

**Controles de salida**

**CUESTIONARIO**

1. ¿Se tienen copias de los archivos en otros locales?

_____
_____
_____

2. ¿Dónde se encuentran esos locales?

_____
_____
_____

3. ¿Que seguridad física se tiene en esos locales?

_____
_____
_____

4. ¿Que confidencialidad se tiene en esos locales?

_____
_____
_____

5. ¿Quién entrega los documentos de salida?

   _____
   _____
   _____

6. ¿En que forma se entregan?

   _____
   _____
   _____

7. ¿Que documentos?

   _____
   _____
   _____

8. ¿Que controles se tienen?

   _____
   _____
   _____

9. ¿Se tiene un responsable (usuario) de la información de cada sistema? ¿Cómo se atienden solicitudes de información a otros usuarios del mismo sistema?

   _____
   _____
   _____

10. ¿Se destruye la información utilizada, o bien que se hace con ella?

    - ☐ Destruye
    - ☐ Vende
    - ☐ Tira
    - ☐ Otro _____

**Control de medios de almacenamiento masivo**

Los dispositivos de almacenamiento representan, para cualquier centro de cómputo, archivos extremadamente importantes cuya pérdida parcial o total podría tener repercusiones muy serias, no sólo en la unidad de informática, sino en la dependencia de la cual se presta servicio. Una dirección de informática bien administrada debe tener perfectamente protegidos estos dispositivos de almacenamiento, además de mantener registros sistemáticos de la utilización de estos archivos, de modo que servirán de base a registros sistemáticos de la utilización de estos archivos, de modo que sirvan de base a los programas de limpieza (borrado de información), principalmente en el caso de las cintas.

Además se deben tener perfectamente identificados los carretes para reducir la posibilidad de utilización errónea o destrucción de la información.

Un manejo adecuado de estos dispositivos permitirá una operación más eficiente y segura, mejorando además los tiempos de procesos.

El objetivo de este cuestionario es evaluar la forma como se administran los dispositivos de almacenamiento básico de la dirección.

## CUESTIONARIO

1. Los locales asignados a la cintoteca y discoteca tienen:
   - ☐ Aire acondicionado
   - ☐ Protección contra el fuego

   (señalar que tipo de protección) _____
   - ☐ Cerradura especial
   - ☐ Otra

2. ¿Tienen la cintoteca y discoteca protección automática contra el fuego?

   SI ( ) NO ( )

   (señalar de que tipo)_____

3. ¿Que información mínima contiene el inventario de la cintoteca y la discoteca?
   - ☐ Número de serie o carrete
   - ☐ Número o clave del usuario
   - ☐ Número del archivo lógico
   - ☐ Nombre del sistema que lo genera
   - ☐ Fecha de expiración del archivo
   - ☐ Fecha de expiración del archivo
   - ☐ Número de volumen

   Otros _____

4. ¿Se verifican con frecuencia la validez de los inventarios de los archivos magnéticos?

   SI ( ) NO ( )

5. ¿En caso de existir discrepancia entre las cintas o discos y su contenido, se resuelven y explican satisfactoriamente las discrepancias?

   SI ( ) NO ( )

6. ¿Que tan frecuentes son estas discrepancias?

_____

7. ¿Se tienen procedimientos que permitan la reconstrucción de un archivo en cinta a disco, el cual fue inadvertidamente destruido?

SI ( ) NO ( )

8. ¿Se tienen identificados los archivos con información confidencial y se cuenta con claves de acceso?

SI ( ) NO ( )

¿Cómo?_____

9. ¿Existe un control estricto de las copias de estos archivos?

SI ( ) NO ( )

10. ¿Que medio se utiliza para almacenarlos?

- ☐ Mueble con cerradura
- ☐ Bóveda

Otro (especifique) _____

11. Este almacén esta situado:

- ☐ En el mismo edificio del departamento
- ☐ En otro lugar

¿Cual?_____

12. ¿Se borran los archivos de los dispositivos de almacenamiento, cuando se desechan estos?

SI ( ) NO ( )

13. ¿Se certifica la destrucción o baja de los archivos defectuosos?

SI ( ) NO ( )

14. ¿Se registran como parte del inventario las nuevas cintas que recibe la biblioteca?

SI ( ) NO ( )

15 ¿Se tiene un responsable, por turno, de la cintoteca y discoteca?

SI ( ) NO ( )

16. ¿Se realizan auditorías periódicas a los medios de almacenamiento?

SI ( ) NO ( )

17. ¿Que medidas se toman en el caso de extravío de algún dispositivo de almacenamiento?

_____
_____
_____

18. ¿Se restringe el acceso a los lugares asignados para guardar los dispositivos de almacenamiento, al personal autorizado?

SI ( ) NO ( )

19. ¿Se tiene relación del personal autorizado para firmar la salida de archivos confidenciales?

SI ( ) NO ( )

20. ¿Existe un procedimiento para registrar los archivos que se prestan y la fecha en que se devolverán?

SI ( ) NO ( )

21. ¿Se lleva control sobre los archivos prestados por la instalación?

SI ( ) NO ( )

22. En caso de préstamo ¿Conque información se documentan?

Nombre de la institución a quién se hace el préstamo. _____

- ☐ fecha de recepción ( )
- ☐ fecha en que se debe devolver ( )
- ☐ archivos que contiene ( )
- ☐ formatos ( )
- ☐ cifras de control ( )
- ☐ código de grabación ( )
- ☐ nombre del responsable que los presto ( )
- ☐ otros

23. Indique qué procedimiento se sigue en el reemplazo de las cintas que contienen los archivos maestros:

_____
_____
_____

24. ¿Se conserva la cinta maestra anterior hasta después de la nueva cinta?

SI ( ) NO ( )

25. ¿El cintotecario controla la cinta maestra anterior previendo su uso incorrecto o su eliminación prematura?

SI ( ) NO ( )

26. ¿La operación de reemplazo es controlada por el cintotecario?

SI ( ) NO ( )

27. ¿Se utiliza la política de conservación de archivos hijo-padre-abuelo?

SI ( ) NO ( )

28. En los procesos que manejan archivos en línea, ¿Existen procedimientos para recuperar los archivos?

SI ( ) NO ( )

29. ¿Estos procedimientos los conocen los operadores?

SI ( ) NO ( )

30. ¿Con que periodicidad se revisan estos procedimientos?

- ☐ Mensual
- ☐ Anual
- ☐ Semestral
- ☐ Otra ¿Cuál?_____

31. ¿Existe un responsable en caso de falla?

SI ( ) NO ( )

32. ¿Explique que políticas se siguen para la obtención de archivos de respaldo?

_____
_____
_____

33. ¿Existe un procedimiento para el manejo de la información de la cintoteca?

SI ( ) NO ( )

34. ¿Lo conoce y lo sigue el cintotecario?

SI ( ) NO ( )

35. ¿Se distribuyen en forma periódica entre los jefes de sistemas y programación informes de archivos para que liberen los dispositivos de almacenamiento?

SI ( ) NO ( )

¿Con qué frecuencia? _____

## Control de mantenimiento

Como se sabe existen básicamente tres tipos de contrato de mantenimiento: El contrato de mantenimiento total que incluye el mantenimiento correctivo y preventivo, el cual a su vez puede dividirse en aquel que incluye las partes dentro del contrato y el que no incluye partes.

El contrato que incluye refacciones es propiamente como un seguro, ya que en caso de descompostura el proveedor debe proporcionar las partes sin costo alguno. Este tipo de contrato es normalmente más caro, pero se deja al proveedor la responsabilidad total del mantenimiento a excepción de daños por negligencia en la utilización del equipo. (Este tipo de mantenimiento normalmente se emplea en equipos grandes).

El segundo tipo de mantenimiento es “por llamada”, en el cual en caso de descompostura se le llama al proveedor y éste cobra de acuerdo a una tarifa y al tiempo que se requiera para componerlo (casi todos los proveedores incluyen, en la cotización de compostura, el tiempo de traslado de su oficina a donde se encuentre el equipo y viceversa). Este tipo de mantenimiento no incluye refacciones.

El tercer tipo de mantenimiento es el que se conoce como “en banco”, y es aquel en el cual el cliente lleva a las oficinas del proveedor el equipo, y este hace una cotización de acuerdo con el tiempo necesario para su compostura mas las refacciones (este tipo de mantenimiento puede ser el adecuado para computadoras personales).

Al evaluar el mantenimiento se debe primero analizar cual de los tres tipos es el que más nos conviene y en segundo lugar pedir los contratos y revisar con detalles que las cláusulas estén perfectamente definidas en las cuales se elimine toda la subjetividad y con penalización en caso de incumplimiento, para evitar contratos que sean parciales.

Para poder exigirle el cumplimiento del contrato de debe tener un estricto control sobre las fallas, frecuencia, y el tiempo de reparación.

Para evaluar el control que se tiene sobre el mantenimiento y las fallas se puede utilizar el siguiente cuestionario.

### CUESTIONARIO

1. Especifique el tipo de contrato de mantenimiento que se tiene (solicitar copia del contrato).
2. ¿Existe un programa de mantenimiento preventivo para cada dispositivo del sistema de computo?

   SI ( ) NO ( )
3. ¿Se lleva a cabo tal programa?

   SI ( ) NO ( )
4. ¿Existen tiempos de respuesta y de compostura estipulados en los contratos?

   SI ( ) NO ( )

5. Si los tiempos de reparación son superiores a los estipulados en el contrato, ¿Qué acciones correctivas se toman para ajustarlos a lo convenido?

SI ( ) NO ( )

6. Solicite el plan de mantenimiento preventivo que debe ser proporcionado por el proveedor.-

SI ( ) NO ( )

¿Cual? _____

8. ¿Cómo se notifican las fallas?

_____
_____
_____

10. ¿Cómo se les da seguimiento?

_____
_____
_____

**12. Orden en el Centro de Cómputo**

Una dirección de Sistemas de Información bien administrada debe tener y observar reglas relativas al orden y cuidado del departamento de cómputo.

Los dispositivos del sistema de cómputo, los archivos magnéticos, pueden ser dañados si se manejan en forma inadecuada y eso puede traducirse en pérdidas irreparables de información o en costos muy elevados en la reconstrucción de archivos.

Se deben revisar las disposiciones y reglamentos que coadyuven al mantenimiento del orden dentro del departamento de cómputo.

**CUESTIONARIO**

1. Indique la periodicidad con que se hace la limpieza del departamento de cómputo y de la cámara de aire que se encuentra abajo del piso falso si existe y los ductos de aire:

Semanalmente ( ) Quincenalmente ( )

Mensualmente ( ) Bimestralmente ( )

No hay programa ( ) Otra (especifique) ( )

2. ¿Existe un lugar asignado a las cintas y discos magnéticos?

SI ( ) NO ( )

3. ¿Se tiene asignado un lugar especifico para papelería y utensilios de trabajo?

SI ( ) NO ( )

4. ¿Son funcionales los muebles asignados para la cintoteca y discoteca?

SI ( ) NO ( )

5. ¿Se tienen disposiciones para que se acomoden en su lugar correspondiente, después de su uso, las cintas, los discos magnéticos, la papelería, etc.?

SI ( ) NO ( )

6. Indique la periodicidad con que se limpian las unidades de cinta:

Al cambio de turno ( ) cada semana ( )

cada día ( ) otra (especificar) ( )

7. ¿Existen prohibiciones para fumar, tomar alimentos y refrescos en el departamento de cómputo?

SI ( ) NO ( )

8. ¿Se cuenta con carteles en lugares visibles que recuerdan dicha prohibición?

SI ( ) NO ( )

9. ¿Se tiene restringida la operación del sistema de cómputo al personal especializado de la Dirección de Informática?

SI ( ) NO ( )

11. Mencione los casos en que personal ajeno al departamento de operación opera el sistema de cómputo:

_____
_____
_____

## 13. Evaluación de la Configuración en el Centro de Cómputo

Los objetivos son evaluar la configuración actual tomando en consideración las aplicaciones y el nivel de uso del sistema, evaluar el grado de eficiencia con el cual el sistema operativo satisface las necesidades de la instalación y revisar las políticas seguidas por la unidad de informática en la conservación de su programoteca.

Esta sección esta orientada a:

a) Evaluar posibles cambios en el hardware a fin de nivelar el sistema de cómputo con la carga de trabajo actual o de comparar la capacidad instalada con los planes de desarrollo a mediano y lago plazo.

b) Evaluar las posibilidades de modificar el equipo para reducir el costo o bien el tiempo de proceso.

c) Evaluar la utilización de los diferentes dispositivos periféricos.

## CUESTIONARIO

1. De acuerdo con los tiempos de utilización de cada dispositivo del sistema de cómputo, ¿existe equipo?

¿Con poco uso? SI ( ) NO ( )

¿Ocioso? SI ( ) NO ( )

¿Con capacidad superior a la necesaria? SI ( ) NO ( )

Describa cuales _____

2. ¿El equipo mencionado en el inciso anterior puede reemplazarse por otro mas lento y de menor costo?

SI ( ) NO ( )

3. Si la respuesta al inciso anterior es negativa, ¿el equipo puede ser cancelado?

SI ( ) NO ( )

4. De ser negativa la respuesta al inciso anterior, explique las causas por las que no puede ser cancelado o cambiado.

_____
_____
_____

5. ¿El sistema de cómputo tiene capacidad de teleproceso?

SI ( ) NO ( )

6. ¿Se utiliza la capacidad de teleproceso?

SI ( ) NO ( )

7. ¿En caso negativo, exponga los motivos por los cuales no utiliza el teleproceso?

SI ( ) NO ( )

8. ¿Cuantas terminales se tienen conectadas al sistema de cómputo?

9. ¿Se ha investigado si ese tiempo de respuesta satisface a los usuarios?

SI ( ) NO ( )

10. ¿La capacidad de memoria y de almacenamiento máximo del sistema de cómputo es suficiente para atender el proceso por lotes y el proceso remoto?

SI ( ) NO ( )

## 14. Seguridad Lógica y Confidencial

La computadora es un instrumento que estructura gran cantidad de información, la cual puede ser confidencial para individuos, empresas o instituciones, y puede ser mal utilizada o divulgada a personas que hagan mal uso de esta.

También pueden ocurrir robos, fraudes o sabotajes que provoquen la destrucción total o parcial de la actividad computacional.

Esta información puede ser de suma importancia, y el no tenerla en el momento preciso puede provocar retrasos sumamente costosos. Antes esta situación, en el transcurso del siglo XX, el mundo ha sido testigo de la transformación de algunos aspectos de seguridad y de derecho.

En la actualidad y principalmente en las computadoras personales, se ha dado otro factor que hay que considerar el llamado ""virus" de las computadoras, el cual aunque tiene diferentes intenciones se encuentra principalmente para paquetes que son copiados sin autorización ("piratas") y borra toda la información que se tiene en un disco.

Al auditar los sistemas se debe tener cuidado que no se tengan copias "piratas" o bien que, al conectarnos en red con otras computadoras, no exista la posibilidad de transmisión del virus.

El uso inadecuado de la computadora comienza desde la utilización de tiempo de máquina para usos ajenos de la organización, la copia de programas para fines de comercialización sin reportar los derechos de autor hasta el acceso por vía telefónica a bases de datos a fin de modificar la información con propósitos fraudulentos.

Un método eficaz para proteger sistemas de computación es el software de control de acceso. Dicho simplemente, los paquetes de control de acceso protegen contra el acceso no autorizado, pues piden del usuario una contraseña antes de permitirle el acceso a información confidencial. Dichos paquetes han sido populares desde hace muchos años en el mundo de las computadoras grandes, y los principales proveedores ponen a disposición de clientes algunos de estos paquetes.

☑ El sistema integral de seguridad debe comprender:

- ✧ Elementos administrativos
- ✧ Definición de una política de seguridad
- ✧ Organización y división de responsabilidades
- ✧ Seguridad física y contra catástrofes (incendio, terremotos, etc.)
- ✧ Prácticas de seguridad del personal
- ✧ Elementos técnicos y procedimientos
- ✧ Sistemas de seguridad (de equipos y de sistemas, incluyendo todos los elementos, tanto redes como terminales.
- ✧ Aplicación de los sistemas de seguridad, incluyendo datos y archivos
- ✧ El papel de los auditores, tanto internos como externos
- ✧ Planeación de programas de desastre y su prueba.

☑ Se debe evaluar el nivel de riesgo que puede tener la información para poder hacer un adecuado estudio costo/beneficio entre el costo por pérdida de información y el costo de un sistema de seguridad, para lo cual se debe considerar lo siguiente:

- ✧ Clasificar la instalación en términos de riesgo (alto, mediano, pequeño).
- ✧ Identificar aquellas aplicaciones que tengan un alto riesgo.
- ✧ Cuantificar el impacto en el caso de suspensión del servicio en aquellas aplicaciones con un alto riesgo.
- ✧ Formular las medidas de seguridad necesarias dependiendo del nivel de seguridad que se requiera.
- ✧ La justificación del costo de implantar las medidas de seguridad para poder clasificar el riesgo e identificar las aplicaciones de alto riesgo, se debe preguntar lo siguiente:
  - ✓ ¿Qué sucedería si no se puede usar el sistema?
  - ✓ Si la contestación es que no se podría seguir trabajando, esto nos sitúa en un sistema de alto riego.

  La siguiente pregunta es:
  - ✓ ¿Que implicaciones tiene el que no se obtenga el sistema y cuanto tiempo podríamos estar sin utilizarlo?
  - ✓ ¿Existe un procedimiento alterno y que problemas nos ocasionaría?
  - ✓ ¿Que se ha hecho para un caso de emergencia?

Una vez que se ha definido, el grado de riesgo, hay que elaborar una lista de los sistemas con las medias preventivas que se deben tomar, así como las correctivas en caso de desastre señalándole a cada uno su prioridad.

Hay que tener mucho cuidado con la información que sale de la oficina, su utilización y que sea borrada al momento de dejar la instalación que está dando respaldo.

☑ Para clasificar la instalación en términos de riesgo se debe:

- ✧ Clasificar los datos, información y programas que contienen información confidencial que tenga un alto valor dentro del mercado de competencia de una organización, e información que sea de difícil recuperación.
- ✧ Identificar aquella información que tenga un gran costo financiero en caso de pérdida o bien puede provocar un gran impacto en la toma de decisiones.
- ✧ Determinar la información que tenga una gran pérdida en la organización y, consecuentemente, puedan provocar hasta la posibilidad de que no pueda sobrevivir sin esa información.

Para cuantificar el riesgo es necesario que se efectúen entrevistas con los altos niveles administrativos que sean directamente afectados por la suspensión en el procesamiento y que cuantifiquen el impacto que les puede causar este tipo de situaciones.

☑ Para evaluar las medidas de seguridad se debe:

- ✧ Especificar la aplicación, los programas y archivos.
- ✧ Las medidas en caso de desastre, pérdida total, abuso y los planes necesarios.
- ✧ Las prioridades que se deben tomar en cuanto a las acciones a corto y largo plazo.
- ✧ En cuanto a la división del trabajo se debe evaluar que se tomen las siguientes precauciones, las cuales dependerán del riesgo que tenga la información y del tipo y tamaño de la organización.
    - ✓ El personal que prepara la información no debe tener acceso a la operación.
    - ✓ Los análisis y programadores no deben tener acceso al área de operaciones y viceversa.
    - ✓ Los operadores no debe tener acceso irrestringido a las bibliotecas ni a los lugares donde se tengan los archivos almacenados; es importante separar las funciones de biblioteca y de operación.
    - ✓ Los operadores no deben ser los únicos que tengan el control sobre los trabajos procesados y no deben hacer las correcciones a los errores detectados.

Al implantar sistemas de seguridad puede, reducirse la flexibilidad en el trabajo, pero no debe reducir la eficiencia.

## 15. Seguridad Física

El objetivo es establecer políticas, procedimientos y prácticas para evitar las interrupciones prolongadas del servicio de procesamiento de datos, información debido a contingencias como incendio, inundaciones, huelgas, disturbios, sabotaje, etc. y continuar en medio de emergencia hasta que sea restaurado el servicio completo.

☑ Entre las precauciones que se deben revisar están:

- ✧ Los ductos del aire acondicionado deben estar limpios, ya que son una de las principales causas del polvo y se habrá de contar con detectores de humo que indiquen la posible presencia de fuego.
- ✧ En las instalaciones de alto riesgo se debe tener equipo de fuente no interrumpible, tanto en la computadora como en la red y los equipos de teleproceso.
- ✧ En cuanto a los extintores, se debe revisar en número de estos, su capacidad, fácil acceso, peso y tipo de producto que utilizan. Es muy frecuente que se tengan los extintores, pero puede suceder que no se encuentren recargados o bien que sean de difícil acceso de un peso tal que sea difícil utilizarlos.
- ✧ Esto es común en lugares donde se encuentran trabajando hombres y mujeres y los extintores están a tal altura o con un peso tan grande que una mujer no puede utilizarlos.
- ✧ Otro de los problemas es la utilización de extintores inadecuados que pueden provocar mayor perjuicio a las máquinas (extintores líquidos) o que producen gases tóxicos.
- ✧ También se debe ver si el personal sabe usar los equipos contra incendio y si ha habido prácticas en cuanto a su uso.
- ✧ Se debe verificar que existan suficientes salidas de emergencia y que estén debidamente controladas para evitar robos por medio de estas salidas.
- ✧ Los materiales más peligrosos son las cintas magnéticas que al quemarse, producen gases tóxicos y el papel carbón que es altamente inflamable.

Tomando en cuenta lo anterior se elaboro el siguiente cuestionario:

### CUESTIONARIO

1. ¿Se han adoptado medidas de seguridad en el departamento de sistemas de información?

SI ( ) NO ( )

2. ¿Existen una persona responsable de la seguridad?

SI ( ) NO ( )

3. ¿Se ha dividido la responsabilidad para tener un mejor control de la seguridad?

SI ( ) NO ( )

4. ¿Existe personal de vigilancia en la institución?

SI ( ) NO ( )

5. ¿La vigilancia se contrata?

- ☐ Directamente
- ☐ Por medio de empresas que venden ese servicio

6. ¿Existe una clara definición de funciones entre los puestos clave?

SI ( ) NO ( )

7. ¿Se investiga a los vigilantes cuando son contratados directamente?

SI ( ) NO ( )

8. ¿Se controla el trabajo fuera de horario?

SI ( ) NO ( )

9. ¿Se registran las acciones de los operadores para evitar que realicen algunas pruebas que puedan dañar los sistemas?.

SI ( ) NO ( )

10. ¿Existe vigilancia en el departamento de cómputo las 24 horas?

SI ( ) NO ( )

11. ¿Existe vigilancia a la entrada del departamento de cómputo las 24 horas?

- ☐ Vigilante
- ☐ Recepcionista
- ☐ Tarjeta de control de acceso
- ☐ Nadie

12. ¿Se permite el acceso a los archivos y programas a los programadores, analistas y operadores?

SI ( ) NO ( )

13. Se ha instruido a estas personas sobre que medidas tomar en caso de que alguien pretenda entrar sin autorización?

SI ( ) NO ( )

14. El edificio donde se encuentra la computadora esta situado a salvo de:

- ☐ Inundación
- ☐ Terremoto

- ☐ Fuego
- ☐ Sabotaje

15. El centro de cómputo tiene salida al exterior al exterior?

SI ( ) NO ( )

16. Describa brevemente la construcción del centro de cómputo, de preferencia proporcionando planos y material con que construido y equipo (muebles, sillas etc.) dentro del centro.

_____
_____
_____

17. ¿Existe control en el acceso a este cuarto?

- ☐ Por identificación personal
- ☐ Por tarjeta magnética
- ☐ Por claves verbales
- ☐ Otras: _____

18. ¿Son controladas las visitas y demostraciones en el centro de cómputo?

SI ( ) NO ( )

19. ¿Se registra el acceso al departamento de cómputo de personas ajenas a la dirección de informática?

SI ( ) NO ( )

20. ¿Se vigilan la moral y comportamiento del personal de la dirección de informática con el fin de mantener una buena imagen y evitar un posible fraude?

SI ( ) NO ( )

21. ¿Existe alarma para

- ☐ Detectar fuego(calor o humo) en forma automática
- ☐ Avisar en forma manual la presencia del fuego
- ☐ Detectar una fuga de agua
- ☐ Detectar magnéticos
- ☐ No existe

22. ¿Estas alarmas están

- ☐ En el departamento de cómputo
- ☐ En la cintoteca y discoteca

23. ¿Existe alarma para detectar condiciones anormales del ambiente

- ☐ En el departamento de cómputo
- ☐ En la cíntoteca y discoteca
- ☐ En otros lados

24. ¿La alarma es perfectamente audible?

SI ( ) NO ( )

25.¿Esta alarma también está conectada

- ☐ Al puesto de guardias
- ☐ A la estación de Bomberos
- ☐ A ningún otro lado

Otro_____

26. Existen extintores de fuego

- ☐ Manuales
- ☐ Automáticos
- ☐ No existen

27. ¿Se ha adiestrado el personal en el manejo de los extintores?

SI ( ) NO ( )

28. ¿Los extintores, manuales o automáticos a base de

Agua SI ( ) NO ( )

Gas SI ( ) NO ( )

Otros SI ( ) NO ( )

29. ¿Se revisa de acuerdo con el proveedor el funcionamiento de los extintores?

SI ( ) NO ( )

30. ¿Si es que existen extintores automáticos son activador por detectores automáticos de fuego?

SI ( ) NO ( )

31. ¿Si los extintores automáticos son a base de agua ¿Se han tomado medidas para evitar que el agua cause mas daño que el fuego?

SI ( ) NO ( )

32. ¿Si los extintores automáticos son a base de gas, ¿Se ha tomado medidas para evitar que el gas cause mas daño que el fuego?

SI ( ) NO ( )

33. ¿Existe un lapso de tiempo suficiente, antes de que funcionen los extintores automáticos para que el personal

| | |
|---|---|
| Corte la acción de los extintores por tratarse de falsas alarmas? | SI ( ) NO ( ) |
| Pueda cortar la energía Eléctrica | SI ( ) NO ( ) |
| Pueda abandonar el local sin peligro de intoxicación | SI ( ) NO ( ) |
| Es inmediata su acción? | SI ( ) NO ( ) |

34. ¿Los interruptores de energía están debidamente protegidos, etiquetados y sin obstáculos para alcanzarlos?

SI ( ) NO ( )

35. ¿Saben que hacer los operadores del departamento de cómputo, en caso de que ocurra una emergencia ocasionado por fuego?

SI ( ) NO ( )

36. ¿El personal ajeno a operación sabe que hacer en el caso de una emergencia (incendio)?

SI ( ) NO ( )

37. ¿Existe salida de emergencia?

SI ( ) NO ( )

38. ¿Esta puerta solo es posible abrirla?

- ☐ Desde el interior
- ☐ Desde el exterior
- ☐ Ambos Lados

39. ¿Se revisa frecuentemente que no esté abierta o descompuesta la cerradura de esta puerta y de las ventanas, si es que existen?

SI ( ) NO ( )

40. ¿Se ha adiestrado a todo el personal en la forma en que se deben desalojar las instalaciones en caso de emergencia?

SI ( ) NO ( )

41. ¿Se ha tomado medidas para minimizar la posibilidad de fuego?

- ☐ Evitando artículos inflamables en el departamento de cómputo
- ☐ Prohibiendo fumar a los operadores en el interior
- ☐ Vigilando y manteniendo el sistema eléctrico
- ☐ No se ha previsto

42. ¿Se ha prohibido a los operadores el consumo de alimentos y bebidas en el interior del departamento de cómputo para evitar daños al equipo?

SI ( ) NO ( )

43. ¿Se limpia con frecuencia el polvo acumulado debajo del piso falso si existe?

SI ( ) NO ( )

44. ¿Se controla el acceso y préstamo en la?

- ☐ Discoteca
- ☐ Cintoteca
- ☐ Programoteca

45. Explique la forma como se ha clasificado la información vital, esencial, no esencial etc.

_____
_____
_____

46. ¿Se cuenta con copias de los archivos en lugar distinto al de la computadora?

SI ( ) NO ( )

47. Explique la forma en que están protegidas físicamente estas copias (bóveda, cajas de seguridad etc.) que garantice su integridad en caso de incendio, inundación, terremotos, etc.

48. ¿Se tienen establecidos procedimientos de actualización a estas copias?

SI ( ) NO ( )

49. Indique el número de copias que se mantienen, de acuerdo con la forma en que se clasifique la información:

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3

50. ¿Existe departamento de auditoría interna en la institución?

SI ( ) NO ( )

51. ¿Este departamento de auditoría interna conoce todos los aspectos de los sistemas?

SI ( ) NO ( )

52. ¿Que tipos de controles ha propuesto?

_____
_____
_____

53. ¿Se cumplen?

SI ( ) NO ( )

54. ¿Se auditan los sistemas en operación?

SI ( ) NO ( )

55.¿Con que frecuencia?

- ☐ Cada seis meses
- ☐ Cada año
- ☐ Otra: _____

56.¿Cuándo se efectúan modificaciones a los programas, a iniciativa de quién es?

- ☐ Usuario
- ☐ Director de informática
- ☐ Jefe de análisis y programación
- ☐ Programador
- ☐ Otras _____

57.¿La solicitud de modificaciones a los programas se hacen en forma?

- ☐ Oral
- ☐ Escrita

En caso de ser escrita solicite formatos

Observaciones_____
_____
_____

58.Una vez efectuadas las modificaciones, ¿se presentan las pruebas a los interesados?

SI ( ) NO ( )

59.¿Existe control estricto en las modificaciones?

SI ( ) NO ( )

60.¿Se revisa que tengan la fecha de las modificaciones cuando se hayan efectuado?

SI ( ) NO ( )

61.¿Si se tienen terminales conectadas, ¿se ha establecido procedimientos de operación?

SI ( ) NO ( )

62.Se verifica identificación:

- ☐ De la terminal
- ☐ Del Usuario
- ☐ No se pide identificación

63.¿Se ha establecido que información puede ser acezada y por qué persona?

SI ( ) NO ( )

64.¿Se ha establecido un número máximo de violaciones en sucesión para que la computadora cierre esa terminal y se de aviso al responsable de ella?

SI ( ) NO ( )

65.¿Se registra cada violación a los procedimientos con el fin de llevar estadísticas y frenar las tendencias mayores?

SI ( ) NO ( )

66.¿Existen controles y medidas de seguridad sobre las siguientes operaciones?

¿Cuáles son?

- ☐ Recepción de documentos _____
- ☐ Información Confidencial _____
- ☐ Captación de documentos _____
- ☐ Cómputo Electrónico _____
- ☐ Programas _____
- ☐ Discotecas y Cintotecas _____
- ☐ Documentos de Salida _____
- ☐ Archivos Magnéticos _____
- ☐ Operación del equipo de computación _____
- ☐ En cuanto al acceso de personal _____
- ☐ Identificación del personal _____
- ☐ Policía _____
- ☐ Seguros contra robo e incendio _____
- ☐ Cajas de seguridad _____
- ☐ Otras (especifique _____

## 16. Seguridad en la utilización del Equipo

En la actualidad los programas y los equipos son altamente sofisticados y sólo algunas personas dentro del centro de cómputo conocen al detalle el diseño, lo que puede provocar que puedan producir algún deterioro a los sistemas si no se toman las siguientes medidas:

1) Se debe restringir el acceso a los programas y a los archivos.
2) Los operadores deben trabajar con poca supervisión y sin la participación de los programadores, y no deben modificar los programas ni los archivos.
3) Se debe asegurar en todo momento que los datos y archivos usados sean los adecuados, procurando no usar respaldos inadecuados.
4) No debe permitirse la entrada a la red a personas no autorizadas, ni a usar las terminales.
5) Se deben realizar periódicamente una verificación física del uso de terminales y de los reportes obtenidos.
6) Se deben monitorear periódicamente el uso que se le está dando a las terminales.
7) Se deben hacer auditorías periódicas sobre el área de operación y la utilización de las terminales.
8) El usuario es el responsable de los datos, por lo que debe asegurarse que los datos recolectados sean procesados completamente. Esto sólo se logrará por medio de los controles adecuados, los cuales deben ser definidos desde el momento del diseño general del sistema.
9) Deben existir registros que reflejen la transformación entre las diferentes funciones de un sistema.
10) Debe controlarse la distribución de las salidas (reportes, cintas, etc.).
11) Se debe guardar copias de los archivos y programas en lugares ajenos al centro de cómputo y en las instalaciones de alta seguridad; por ejemplo: los bancos.
12) Se debe tener un estricto control sobre el acceso físico a los archivos.
13) En el caso de programas, se debe asignar a cada uno de ellos, una clave que identifique el sistema, subsistema, programa y versión.

También evitará que el programador ponga nombres que nos signifiquen nada y que sean difíciles de identificar, lo que evitará que el programador utilice la computadora para trabajos personales. Otro de los puntos en los que hay que tener seguridad es en el manejo de información. Para controlar este tipo de información se debe:

1) Cuidar que no se obtengan fotocopias de información confidencial sin la debida autorización.
2) Sólo el personal autorizado debe tener acceso a la información confidencial.

3) Controlar los listados tanto de los procesos correctos como aquellos procesos con terminación incorrecta.
4) Controlar el número de copias y la destrucción de la información y del papel carbón de los reportes muy confidenciales.

El factor más importante de la eliminación de riesgos en la programación es que todos los programas y archivos estén debidamente documentados.

☑ El siguiente factor en importancia es contar con los respaldos, y duplicados de los sistemas, programas, archivos y documentación necesarios para que pueda funcionar el plan de emergencia.

- ✧ Equipo, programas y archivos
- ✧ Control de aplicaciones por terminal
- ✧ Definir una estrategia de seguridad de la red y de respaldos
- ✧ Requerimientos físicos.
- ✧ Estándar de archivos.
- ✧ Auditoría interna en el momento del diseño del sistema, su implantación y puntos de verificación y control.

**17. Seguridad al Restaurar el Equipo**

En un mundo que depende cada día mas de los servicios proporcionados por las computadoras, es vital definir procedimientos en caso de una posible falta o siniestro. Cuando ocurra una contingencia, es esencial que se conozca al detalle el motivo que la originó y el daño causado, lo que permitirá recuperar en el menor tiempo posible el proceso perdido. También se debe analizar el impacto futuro en el funcionamiento de la organización y prevenir cualquier implicación negativa.

En todas las actividades relacionadas con las ciencias de la computación, existe un riesgo aceptable, y es necesario analizar y entender estos factores para establecer los procedimientos que permitan analizarlos al máximo y en caso que ocurran, poder reparar el daño y reanudar la operación lo más rápidamente posible.

En una situación ideal, se deberían elaborar planes para manejar cualquier contingencia que se presente.

Analizando cada aplicación se deben definir planes de recuperación y reanudación, para asegurarse que los usuarios se vean afectados lo menos posible en caso de falla o siniestro.

☑ Las acciones de recuperación disponibles a nivel operativo pueden ser algunas de las siguientes:

- ✧ En algunos casos es conveniente no realizar ninguna acción y reanudar el proceso.

- ✧ Mediante copias periódicas de los archivos se puede reanudar un proceso a partir de una fecha determinada.
- ✧ El procesamiento anterior complementado con un registro de las transacciones que afectaron a los archivos permitirá retroceder en los movimientos realizados a un archivo al punto de tener la seguridad del contenido del mismo a partir de él reanudar el proceso.
- ✧ Analizar el flujo de datos y procedimientos y cambiar el proceso normal por un proceso alterno de emergencia.
- ✧ Reconfigurar los recursos disponibles, tanto de equipo y sistemas como de comunicaciones.

Cualquier procedimiento que se determine que es el adecuado para un caso de emergencia deberá ser planeado y probado previamente.

Este grupo de emergencia deberá tener un conocimiento de los posibles procedimientos que puede utilizar, además de un conocimiento de las características de las aplicaciones, tanto desde el punto técnico como de su prioridad, el nivel de servicio planeado y su influjo en la operación de la organización.

Además de los procedimientos de recuperación y reinicio de la información, se deben contemplar los procedimientos operativos de los recursos físicos como hardware y comunicaciones, planeando la utilización de equipos que permitan seguir operando en caso de falta de la corriente eléctrica, caminos alternos de comunicación y utilización de instalaciones de cómputo similares. Estas y otras medidas de recuperación y reinicio deberán ser planeadas y probadas previamente como en el caso de la información.

El objetivo del siguiente cuestionario es evaluar los procedimientos de restauración y repetición de procesos en el sistema de cómputo.

## CUESTIONARIO

1) ¿Existen procedimientos relativos a la restauración y repetición de procesos en el sistema de cómputo?

SI ( ) NO ( )

2) Enuncie los procedimientos mencionados en el inciso anterior

_____
_____
_____

3) ¿Cuentan los operadores con alguna documentación en donde se guarden las instrucciones actualizadas para el manejo de restauraciones?

SI ( ) NO ( )

☑ En el momento que se hacen cambios o correcciones a los programas y/o archivos se deben tener las siguientes precauciones:

- ✧ Las correcciones de programas deben ser debidamente autorizadas y probadas. Con esto se busca evitar que se cambien por nueva versión que antes no ha sido perfectamente probada y actualizada.
- ✧ Los nuevos sistemas deben estar adecuadamente documentos y probados.
- ✧ Los errores corregidos deben estar adecuadamente documentados y las correcciones autorizadas y verificadas.

☑ Los archivos de nuevos registros o correcciones ya existentes deben estar documentados y verificados antes de obtener reportes.

**18. Procedimientos de Respaldo en el Caso de Desastre**

Se debe establecer en cada dirección de informática un plan de emergencia el cual ha de ser aprobado por la dirección de informática y contener tanto procedimiento como información para ayudar a la recuperación de interrupciones en la operación del sistema de cómputo.
El sistema debe ser probado y utilizado en condiciones anormales, para que en casó de usarse en situaciones de emergencia, se tenga la seguridad que funcionará.

La prueba del plan de emergencia debe hacerse sobre la base de que la emergencia existe y se ha de utilizar respaldos.

Se deben evitar suposiciones que, en un momento de emergencia, hagan inoperante el respaldo, en efecto, aunque el equipo de cómputo sea aparentemente el mismo, puede haber diferencias en la configuración, el sistema operativo, en disco etc.

El plan de emergencia una vez aprobado, se distribuye entre personal responsable de su operación, por precaución es conveniente tener una copia fuera de la dirección de informática.

En virtud de la información que contiene el plan de emergencia, se considerará como confidencial o de acceso restringido.

La elaboración del plan y de los componentes puede hacerse en forma independiente de acuerdo con los requerimientos de emergencia, La estructura del plan debe ser tal que facilite su actualización.

Para la preparación del plan se seleccionará el personal que realice las actividades claves del plan. El grupo de recuperación en caso de emergencia debe estar integrado por personal de administración de la dirección de informática, debe tener tareas específicas como la operación del equipo de respaldo, la interfaz administrativa.

☑ Los desastres que pueden suceder podemos clasificar así:

- ✧ Completa destrucción del centro de cómputo,
- ✧ Destrucción parcial del centro de cómputo,
- ✧ Destrucción o mal funcionamiento de los equipos auxiliares del centro de cómputo (electricidad, aire, acondicionado, etc.)
- ✧ Destrucción parcial o total de los equipos descentralizados
- ✧ Pérdida total o parcial de información, manuales o documentación
- ✧ Pérdida del personal clave
- ✧ Huelga o problemas laborales.

☑ El plan en caso de desastre debe incluir:

- ✧ La documentación de programación y de operación.
- ✧ Los equipos:
  - ✓ El equipo completo
  - ✓ El ambiente de los equipos
  - ✓ Datos y archivos
  - ✓ Papelería y equipo accesorio
  - ✓ Sistemas (sistemas operativos, bases de datos, programas).

El plan en caso de desastre debe considerar todos los puntos por separado y en forma integral como sistema. La documentación estará en todo momento tan actualizada como sea posible, ya que en muchas ocasiones no se tienen actualizadas las últimas modificaciones y eso provoca que el plan de emergencia no pueda ser utilizado.

☑ Cuando el plan sea requerido debido a una emergencia, el grupo deberá:

- ✧ Asegurarse de que todos los miembros sean notificados,
- ✧ informar al director de informática,
- ✧ Cuantificar el daño o pérdida del equipo, archivos y documentos para definir que parte del plan debe ser activada.
- ✧ Determinar el estado de todos los sistemas en proceso,
- ✧ Notificar a los proveedores del equipo cual fue el daño,
- ✧ Establecer la estrategia para llevar a cabo las operaciones de emergencias tomando en cuenta:
  - ✓ Elaboración de una lista con los métodos disponibles para realizar la recuperación.

- ✓ Señalamiento de la posibilidad de alternar los procedimientos de operación (por ejemplo, cambios en los dispositivos, sustituciones de procesos en línea por procesos en lote).
- ✓ Señalamiento de las necesidades para armar y transportar al lugar de respaldo todos los archivos, programas, etc., que se requieren.
- ✓ Estimación de las necesidades de tiempo de las computadoras para un periodo largo.

☑ Cuando ocurra la emergencia, se deberá reducir la carga de procesos, analizando alternativas como:

- ✧ Posponer las aplicaciones de prioridad más baja,
- ✧ Cambiar la frecuencia del proceso de trabajos.
- ✧ Suspender las aplicaciones en desarrollo.

Por otro lado, se debe establecer una coordinación estrecha con el personal de seguridad a fin de proteger la información.

Respecto a la configuración del equipo hay que tener toda la información correspondiente al hardware y software del equipo propio y del respaldo.

☑ Deberán tenerse todas las especificaciones de los servicios auxiliares tales como energía eléctrica, aire acondicionado, etc. a fin de contar con servicios de respaldo adecuados y reducir al mínimo las restricciones de procesos, se deberán tomar en cuenta las siguientes consideraciones:

- ✧ Mínimo de memoria principal requerida y el equipo periférico que permita procesar las aplicaciones esenciales.
- ✧ Se debe tener documentados los cambios de software.
- ✧ En caso de respaldo en otras instituciones, previamente se deberá conocer el tiempo de computadora disponible.

☑ Es conveniente incluir en el acuerdo de soporte recíproco los siguientes puntos:

- ✧ Configuración de equipos.
- ✧ Configuración de equipos de captación de datos.
- ✧ Sistemas operativos.
- ✧ Configuración de equipos periféricos.

**“La seguridad máxima tendría un costo infinito”**
Anónimo