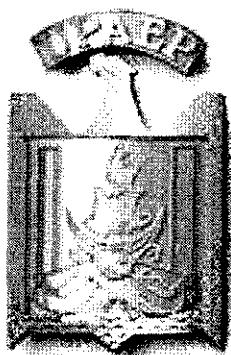


UNIVERSIDAD POPULAR AUTÓNOMA DEL ESTADO DE PUEBLA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL Y TECNOLOGÍAS DE INFORMACIÓN

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

TESIS



TEMA

"HERRAMIENTA PARA EL APOYO EN LA ADMINISTRACIÓN DE UN SERVIDOR UNIX."

**QUE PARA OBTENER EL TÍTULO DE LICENCIATURA EN
INGENIERIA EN COMPUTACIÓN**

PRESENTA:

ADALBERTO GONZÁLEZ AYALA

ASESOR

M.C. SANDRA ROCIO MURILLO CANO

PUEBLA, PUE., JULIO, 2005.



UPAEP – Secretaría General

Dirección General de Apoyos Académicos

Dirección del Centro de Recursos para el Aprendizaje y la Investigación.

Biblioteca Central - **Karol Wojtyla**

Tesis Digitales Restricciones de uso:

DERECHOS RESERVADOS ©

PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de textos, imágenes, gráficas, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente de donde la obtuvo mencionando el autor o autores involucrados en el documento.

Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

**A mis padres y
a mis hermanos.**

Índice

Índice de figuras	iv
Introducción.....	v
Capítulo 1. Seguridad informática	1
1.1 Conceptos básicos de seguridad informática.....	2
1.1.1 ¿Qué es una red?.....	2
1.1.2 Gestión de seguridad y gestión de red.....	2
1.1.3 Conceptos de seguridad informática.	3
1.1.4 Objetivo de la seguridad informática.	6
1.1.5 Características de la información.	7
1.1.6 Amenazas informáticas.	9
1.1.7 Iniciativas de seguridad.	11
1.2 Estadísticas.	13
1.2.1 CSI/FBI	13
1.2.2 US-CERT	16
1.3 Situación Actual.	18
1.4 Objetivo del Proyecto.....	19
Capítulo 2. Estado del arte de la seguridad informática en Linux	20
2.1 Sistemas Operativos (S.O.).	21
2.2 UNIX.....	21
2.2.1 LINUX.....	21
2.3 Seguridad en Linux	22
2.3.1 seguridad en el sistema de cuentas.	22
2.3.2 Control de los passwords.....	22
2.3.3 Control de las cuentas.....	24
2.3.4 Seguridad en el sistema de archivos.....	25
2.3.5 Llenado de disco.....	29
2.3.6 Permisos de un archivo.....	29

2.3.7 Métodos de acceso SUID, SGID y Sticky bit.	33
2.3.8 listas de control de acceso: ACLs.....	35
2.3.9 Almacenamiento seguro.	39
2.3.10 Informática forense.....	40
2.4 Seguridad en Redes	41
2.4.1 Escaneo de puertos.	42
2.4.2 Seguridad en la Web.....	45
2.4.3 Puertos.	47
2.5 Virus en Linux.....	48
2.6 Criptografía.	49
2.6.1 Algoritmos Simétricos.....	52
2.6.2 Algoritmos de llave pública.	52
2.6.3 PGP.....	55
2.7 Archivos Log	57
2.8 Herramientas de seguridad.	61
Capítulo 3. Diseño de la herramienta.....	65
3.1 Introducción.....	66
3.2 PSP	67
3.2.1 PSP (personal software process)	67
3.2.2 Niveles de PSP	68
3.3 Descripción General de HAASU	70
3.3.1 Perspectiva.....	70
3.3.1.1 Interfaz De Sistema	70
3.3.1.2 Interfaces De Hardware.....	71
3.3.1.3 Interfaces De Comunicación	72
3.4 Funcionalidades.....	72
3.4.1 Operaciones.....	73
3.5 Los Módulos de HAASU.	76
3.5.1 Puertos.	76

3.5.2 LOGs.	77
3.5.3 Hardware.	79
3.5.4 Reportes.....	80
3.5.5 Observaciones.	81
Capítulo 4. Pruebas y Resultados.....	82
Capítulo 5. Conclusiones	86
Capítulo 6. Trabajo a futuro	88
Bibliografía.....	90

Índice de figuras

Capítulo 1

Figura 1.1 elementos que comprometen a un sistema [BOR01].	9
Figura 1.2 Porcentaje del presupuesto de TI invertido en seguridad [COM04].	14
Figura 1.3 Uso No autorizado de sistemas de computadora [COM04].	15
Figura 1.4 Cantidad de dólares perdidos por tipo [COM04].	16
Figura 1.5 tabla de incidentes por tipo en el 2004 [CER05].	17
Figura 1.6 tabla de incidentes por tipo en el 2005 [CER05].	18

Capítulo 2

Figura 2.1 permisos de un archivo [VIL12].	30
Figura 2.2 diagrama de un algoritmo asimétrico.	53
Figura 2.3 Codificación de un mensaje PGP [LUC99].	57

Capítulo 3

Figura 3.1 Actividades de HAASU en el servidor.	67
Figura 3.2 Interfaz del sistema.	71
Figura 3.3 Estructura general de la aplicación.	72
Figura 3.4 Diagrama general de los módulos de HAASU.	73
Figura 3.4 pantalla principal de HAASU.	76
Figura 3.5 Informe de puertos abiertos vía ethereal.	77
Figura 3.6 Informe del archivo LOG: secure	78
Figura 3.7 Informe de hardware	79
Figura 3.8 Informe de llenado de particiones	80
Figura 3.9 Comentario agregado más la clave para su búsqueda.	81

Capítulo 4

Figura 4.1 Tabla de pruebas de integración..	83
Figura 4.1 Tabla de pruebas de integración (continuación).	84
Figura 4.2 Registro de productividad del desarrollador.	85

Introducción

Una de las finalidades de la administración de una red de computadoras es maximizar el desempeño de los nodos que la conforman. Esta actividad se basa en los principios de planeación, organización, monitoreo, seguridad, contabilidad y control de las actividades y recursos que tradicionalmente se manejan en las áreas administrativas.

Dentro de muchas organizaciones que precisen la administración de una red de computadoras, existe personal dedicado a la operación de dicha administración, siendo tarea en el mejor de los casos de una sola persona. Esta persona generalmente tiene como responsabilidad todos los servicios que se ofrezcan. Esto representa que dicha persona esté esclavizada de manera operativa a monitorear servicios en espera de que algo suceda.

El objetivo general de esta tesis es diseñar, desarrollar e implementar una herramienta informática para apoyar en la administración de los recursos de software y hardware de un servidor Unix.

Cabe recalcar que lo esencial de una red no es la infraestructura de interconexión por si misma, ya que ésta no procesa datos ni produce información. Lo importante se constituye por los nodos, sus recursos y desempeño.

En el capítulo 1 se presentan los conceptos básicos de seguridad informática, datos relevantes, además de una breve introducción a la situación actual de una estación de trabajo por la que se origina este proyecto.

En el capítulo 2 se presenta un análisis del estado del arte sobre la seguridad en el sistema operativo Linux. Se exponen brevemente algunas herramientas existentes.

En el capítulo 3 se presenta el diseño de HAASU como una propuesta a la problemática de la administración de un servidor Linux.

En el capítulo 4 se presentan las pruebas de integración así como la productividad registrada por el desarrollador de HASSU.

En el capítulo 5 se presentan las conclusiones de este proyecto.

Capítulo 1. Seguridad informática

Se presentan los conceptos básicos de seguridad informática, datos relevantes, además de una breve introducción a la situación actual de una estación de trabajo por la que se origina este proyecto

1.1 Conceptos básicos de seguridad informática.

Se presenta la información mas relevante de la seguridad informática, como conceptos, herramientas, problemas comunes y estadísticas generales.

1.1.1 ¿Qué es una red?

Una red es una serie de puntos o nodos interconectados a través de enlaces de comunicación [TAN03].

El objetivo básico de una red es permitir que varias computadoras se interconecten entre sí para compartir información.

Cabe recalcar que lo esencial de una red no es la infraestructura de interconexión por si misma, ya que ésta no procesa datos ni produce información. Lo importante se constituye por los nodos, sus recursos y desempeño.

De aquí en adelante al mencionar el término red se deberá entender el conjunto de nodos, recursos locales y remotos usados y no la infraestructura de interconexión.

1.1.2 Gestión de seguridad y gestión de red

La gestión se define como el conjunto de actividades que controlan o vigilan el uso de los recursos. Debe proporcionar la posibilidad de supervisar el estado, medir el rendimiento, reconocer actividades anormales y recuperar el servicio. Las funciones de red se suelen agrupar en dos categorías [STA94]:

Supervisión de red. Se considera una función de "lectura" y se encarga de observar y analizar el estado y el comportamiento de la configuración y componentes de la red.

Control de red. Se le considera como una función de "escritura" y se encarga de alterar los parámetros de los distintos componentes de la configuración de la red y hacer que lleven a cabo las acciones que se determinen.

1.1.3 Conceptos de seguridad informática.

A continuación se presentan algunos conceptos importantes de seguridad informática.

Biométrica: métodos automatizados usados para el reconocimiento de personas basados en comportamientos o características físicas. Se usan comúnmente para verificar identidades y restringir accesos a edificios, redes de computadoras otros sitios seguros.

Cracker: Persona que intenta acceder a un sistema informático sin autorización. Estas personas tienen a menudo malas intenciones, en contraste con los hackers, y pueden disponer de muchos medios para introducirse en un sistema.

Cyber: Prefijo utilizado ampliamente en la comunidad Internet para denominar conceptos relacionados con las redes.

Dato: Unidad mínima entre las que componen una información. Es una palabra latina que significa "lo que se da".

Denial of Service: DoS (Denegación de Servicio) En Internet, un DoS o ataque de denegación de servicio (no confundir con DOS, Disk Operating System, con O mayúscula) es un incidente en el cual un usuario o una organización se ven privados de un recurso que normalmente podrían usar. Habitualmente, la pérdida del servicio supone la indisponibilidad de un determinado servicio de red, como el correo electrónico, o la pérdida temporal de toda la conectividad y todos los servicios de red.

Encryption: El cifrado es el tratamiento de un conjunto de datos, contenidos o no en un paquete, a fin de impedir que nadie excepto el destinatario de los mismos pueda leerlos. Hay muchos tipos de cifrado de datos, que constituyen la base de la seguridad de la red.

Firewall: Dispositivo que se coloca entre una red local e Internet y cuyo objetivo es asegurar que todas las comunicaciones entre los usuarios de dicha red e Internet se realicen conforme a las normas de seguridad de la organización que lo instala.

Gusano: Programa informático que se autoduplica y autopropaga. En contraste con los virus, los gusanos suelen estar especialmente escritos para redes.

Hacker: Persona que goza alcanzando un conocimiento profundo sobre el funcionamiento interno de un sistema, de una computadora o de una red de computadoras.

Información: es una agregación de datos que tiene significado específico más allá de cada uno de estos, y tendrá un sentido particular según como y quien la procese.

Malicious Code: Cualquier programa cuyo objetivo sea causar daños a ordenadores, sistemas o redes y, por extensión, a sus usuarios.

Misuse of Resources: Cuando a los recursos se le s da un uso para el cual no fue diseñado.

PCC proof-carrying code: es una técnica que puede ser usada para la ejecución segura de código sospechoso. En un caso típico de PCC, el receptor de código establece medidas de seguridad para garantizar el buen comportamiento de programas, y el productor de código crea una prueba formal de seguridad que comprueba, si existe código sospechoso, en base a las reglas de seguridad. Entonces, el receptor es capaz de usar una prueba simple y rápida para comprobar, con la certeza, que la prueba es válida y de ahí el que el código es seguro para ejecutar.

Reconnaissance Activity: (actividad de reconocimiento) es el término militar para la recopilación de información activa sobre un enemigo, u otras condiciones, por medio de la observación física. Esto es la parte de la inteligencia de combate. Compárese con el *contraespionaje*.

Root Compromise: Es el termino utilizado cuando la computadora ha sufrido una intrusión por parte de un usuario malicioso que ha suplantado la identidad del *administrador o root*.

SSH: SSH codifica todo el tráfico (incluso contraseñas) para eliminar con eficiencia ataques a nivel de red en servicios como, ftp, telnet, rlogin y otros.

Troyano: Programa informático que lleva en su interior la lógica necesaria para que el creador del programa pueda acceder al interior del sistema en el que se introduce de manera subrepticia.

Web Site Defacement: una forma de código maligno en la cual un Sitio Web es "destrozado". A menudo el hacker malévolo sustituye el contenido normal del sitio por

un mensaje específico político o social o borrar el contenido del sitio completamente, confiando en vulnerabilidades de seguridad conocidas para el acceso al contenido del sitio.

Virus: Programa cuyo objetivo es causar daños en un sistema informático y que a tal fin se oculta o disfraz para no ser detectado. Estos programas son de muy diversos tipos y pueden causar problemas de diversa gravedad en los sistemas a los que infectan.

1.1.4 Objetivo de la seguridad informática.

No existe un sistema completamente seguro. Todo lo que puede hacer es aumentar la dificultad para que alguien pueda comprometer su sistema [BOR01].

El objetivo principal en seguridad informática es el aislamiento de los actos no deseables, y la prevención de aquellos que no se hayan considerado, de forma que si se producen hagan el menor daño posible. Entre las distintas actividades que se deben llevar a cabo se destacan las siguientes [AMO94]:

- **Identificación de los usuarios.** Existen varias técnicas entre las que se encuentran las contraseñas (passwords), o sistemas más sofisticados como reconocimiento del habla o a partir de huella dactilar o la retina del ojo.
- **Detección de intrusos en la red.** Se debe detectar y actuar sobre cualquier acceso no autorizado a un sistema. El objetivo es la detección de intrusos en tiempo real, antes de que el sistema haya sido dañado seriamente.
- **Análisis de riesgo.** Intenta cuantificar los beneficios obtenidos con la protección contra amenazas de seguridad. El riesgo es función de la frecuencia con la que se producen dichas amenazas, vulnerabilidad de la protección contra las mismas y las pérdidas potenciales que se produjesen en el caso de que se diese una.

- **Clasificación apropiada de los datos.** En la gestión de seguridad llegan gran cantidad de datos provenientes de los distintos programas de control generados a partir de las actuaciones que llevan a cabo los usuarios en el sistema. Es importante, para una buena supervisión de la seguridad, el clasificar los datos convenientemente, de tal forma que se ahorre tiempo en su análisis.
- **Control de las nuevas aplicaciones.** Cuando se instala una nueva aplicación se debe comprobar que no introduzca nuevas brechas de seguridad especialmente si se ejecuta con permisos de root.
- **Análisis de los accesos de los usuarios.** Es necesario tener un control para poder detectar intentos de acceso no autorizado.

Establecer el valor de la información es algo totalmente relativo, pues constituye un recurso que, en muchos casos, no se valora adecuadamente debido a su intangibilidad, cosa que no ocurre con los equipos, las aplicaciones y la documentación.

Para comenzar con el panorama de la seguridad informática se deberán conocer las características de lo que se pretende proteger: la información.

1.1.5 Características de la información.

Existe Información que **debe o puede ser pública**: puede ser visualizada por cualquier persona (por ejemplo índice de analfabetismo en un país); y aquella que **debe ser privada**: sólo puede ser visualizada por un grupo selecto de personas que trabaja con ella (por ejemplo antecedentes médicos). En esta última debemos maximizar nuestros esfuerzos para preservarla de ese modo reconociendo las siguientes características en la Información [BOR01]:

- **Es Crítica:** es indispensable para garantizar la continuidad operativa.
- **Es Valiosa:** es un activo con valor en sí misma.
- **Es Sensitiva:** debe ser conocida por las personas que la procesan y sólo por ellas.

Para hacer frente a las amenazas de seguridad del sistema, se definen una serie de servicios para proteger los sistemas de procesamiento de datos y transferencia de información de una organización. Estos servicios hacen uso de diversos mecanismos de seguridad. Una clasificación de dichos servicios es la siguiente [AMO94]:

- **La integridad** de la Información es la característica que hace que su contenido permanezca inalterado a menos por personal no autorizado.
- **La disponibilidad u operatividad** de la información es su capacidad de estar siempre disponible para ser procesada por las personas autorizadas.
- **La privacidad o confidencialidad** de la información es la necesidad de que la misma sólo sea conocida por personas autorizadas.
- **El control** sobre la información permite asegurar que sólo los usuarios autorizados pueden decidir cuando y como permitir el acceso a la misma.
- **La autenticidad** permite definir que la información requerida es válida y utilizable en tiempo, forma y distribución. También permite asegurar el origen de la información, validando el emisor de la misma, para evitar suplantación de identidades.
- **Protección a la Réplica:** se asegura que una solicitud sólo puede realizarse una vez, a menos que se especifique lo contrario

- **No Repudio:** mediante la cual se evita que cualquier entidad que envió o recibió información alegue, ante terceros, que no la envió o recibió.
- **Consistencia:** se debe poder asegurar que el sistema se comporte como se supone que debe hacerlo ante los usuarios que corresponda.
- **Aislamiento:** este aspecto esta íntimamente relacionado con la *confidencialidad*, permite regular el acceso al sistema, impidiendo que personas no autorizadas hagan uso del mismo.

1.1.6 Amenazas informáticas.

Es necesario definir amenaza en el entorno informático, como cualquier elemento que pueda comprometer al sistema.



Figura 1.1 elementos que comprometen a un sistema [BOR01].

Las amenazas pueden ser analizadas en tres momentos: antes del ataque, durante y después del mismo. Estos mecanismos nos revelaran un panorama con el cual decidiremos que metodología es mejor para garantizar la seguridad de nuestro sistema informático [BOR01].

- **La prevención (antes):** acciones que aumentan la seguridad (o fiabilidad) de un sistema durante su funcionamiento normal.
- **La detección (durante):** acciones orientadas a revelar violaciones a la seguridad. Generalmente son programas de monitoreo.
- **La recuperación (después):** acciones que se aplican cuando se ha detectado la intrusión del sistema, para regresar el sistema a su estado original.

Los problemas y amenazas de seguridad son comúnmente explotados por gusanos, virus y troyanos.

Ante un problema de seguridad, se deben plantear preguntas relacionadas con la solución de un problema dado [BOR01].

¿Cuánto tardará la amenaza en superar la solución planteada?

¿Cómo se hace para detectarla e identificarla a tiempo?

¿Cómo neutralizarla?

Los riesgos deben prevenirse antes de que alguno de estos se vuelva una amenaza, por esto es necesario tomar en consideración algunos de estos puntos:

- Minimizar la posibilidad de su ocurrencia.

- Reducir al mínimo el perjuicio producido, si es que no se ha podido evitar que la catástrofe ocurra.
- Diseñar métodos para una inmediata recuperación de daños experimentados.
- Corrección de medidas de seguridad con base en antecedentes o en la experiencia ganada.

Ahora como consecuencia de la amenaza, o de la no acción, o de la acción defectuosa está el daño. El daño se produce cuando no se ha podido identificar un riesgo que lleve a una amenaza cuya consecuencia es el mismo daño.

La seguridad indicará el índice en que un sistema informático este libre de todo peligro, daño o riesgo. Esta característica, según los especialistas es imposible de conseguir en un 100% por lo que se habla de **Fiabilidad** y se define como la probabilidad de que un sistema se comporte tal y como se espera que lo haga, y se recomienda hablar de sistemas fiables en vez de sistemas seguros [HUE12].

Ahora para garantizar la fiabilidad de un sistema se deben seguir a fondo características como integridad, operatividad, privacidad, control y autenticidad. Se debe analizar que es lo que se desea proteger, y como lograrlo, para luego tener una estrategia adecuada de seguridad que reflejara disminución de riesgos.

1.1.7 Iniciativas de seguridad.

Para evitar que nuestros equipos de computo se vuelvan objetivos de amenazas, es necesario tomar en cuenta algunas iniciativas para prever danos o perdidas, en el peor de los casos irreparables. He aquí algunas de estas iniciativas:

- Proof-carrying code (PCC), proof-carrying authorization:
 - El código enviado por el servidor contiene una prueba segura de que dicho código sigue una política de seguridad determinada.
 - El cliente dispone de un sistema de validación de pruebas.
- Tarjetas inteligentes, biométrica para autenticación.
- Estándar de encriptación avanzado.
 - Algoritmo de clave simétrica de 128 a 256 bits.
- Firewalls con análisis criptográficos de programas.
 - ZoneAlarm es gratuito y eficaz.
- Detección de una intrusión:
 - Analizar el tráfico de red, examinar cada paquete del flujo de datos para identificar amenazas de usuarios no autorizados, hackers que hayan burlado los sistemas de control.
 - Los firewalls no son suficiente.
- AAA (autorización, autenticación y administración o gestión de cuentas) para servidores Web y de aplicaciones:
 - RADIUS (*Remote Authentication Dial-In User Service*): protocolo cliente/servidor que comunica los servidores de acceso remoto con el servidor central para autenticar las llamadas entrantes o los usuarios remotos y autorizar su acceso al servicio o sistema solicitado. Se empieza aplicar también en servidores de aplicaciones.
 - RADIUS es un estándar de facto de la industria, propuesto por el IETF.
- Recordar que, como siempre, las personas son el elemento clave:
 - Experiencias.
 - Trabajos "desde dentro".

1.2 Estadísticas.

A continuación se presentan algunas estadísticas sobre delitos informáticos.

1.2.1 CSI/FBI

Computer Crime and Security Survey es dirigida por CSI (Computer Security Institute) con la participación del FBI San Francisco división Intrusión en Computadoras. La revisión está ahora en su noveno año y creen, que al paso de este tiempo la fiabilidad de su información es cada vez más confiable en el campo de la seguridad.

Los resultados de la revisión del año 2004 están basados en las respuestas de 494 practicantes de seguridad en computación en corporaciones estadounidenses, agencias de gobierno, instituciones financieras, instituciones médicas y universidades.

Una pregunta surgió para determinar el tamaño del presupuesto de seguridad de información de una organización con relación al presupuesto total en TI de dicha organización.

Como se ve en la figura 1.3, el 46% de los encuestados indicó que su organización asignó entre el 1 por ciento y el 5% del total del presupuesto de TI a la seguridad.

Sólo el 16% de los encuestados indicó que la seguridad recibió menos del 1 por ciento del presupuesto de TI, el 23% de demandados indicó que la seguridad recibió más que el 5 por ciento del presupuesto, mientras el 14% de los demandados indicó que la parte era desconocida a ellos.

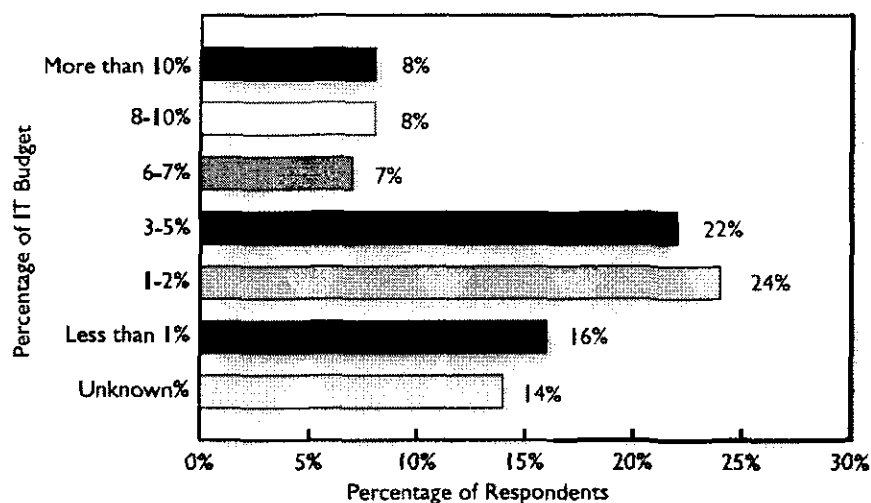


Figura 1.2 Porcentaje del presupuesto de TI invertido en seguridad [COM04].

Observando la figura 1.4, se puede percibir que la frecuencia total de ataques (acertados) contra sistemas computacionales disminuyó en el año 2004, tendencia que continua desde el 2001. Este año el porcentaje de encuestados experimentaron que en su organización el uso no autorizado de sistemas de computo durante los 12 meses pasados disminuyó al 53%, el porcentaje más pequeño a esta pregunta apareció en la edición 1999. Además, el porcentaje de encuestados que respondieron que no había ningún uso no autorizado de los sistemas de computo en su organización aumentó al 35%, el porcentaje de encuestados que no saben si ocurrió algún uso no autorizado se encuentra debajo del 11 por ciento.

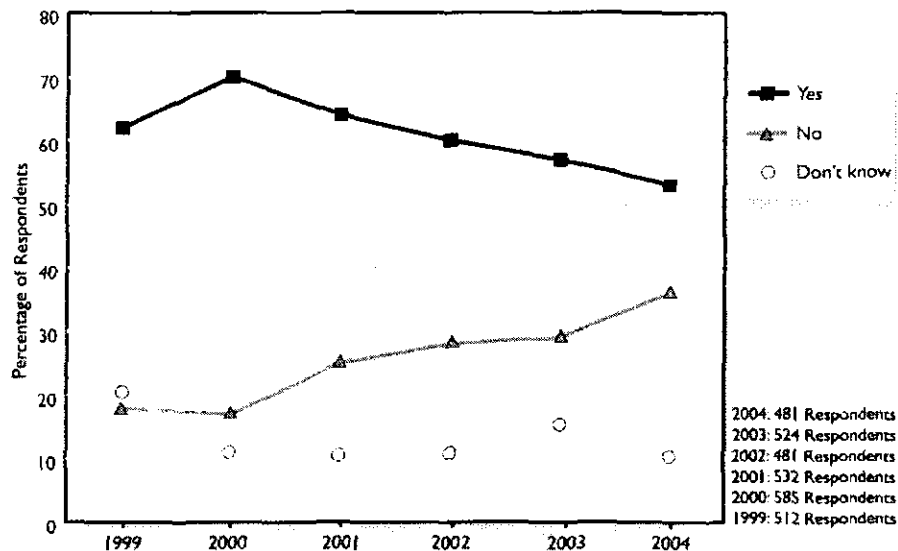


Figura 1.3 Uso No autorizado de sistemas de computadora [COM04].

Las estimaciones de las pérdidas causadas por diferentes tipos de incidentes de seguridad computacional son mostradas en la figura 1.5. Varios puntos importantes están relacionados con la figura 1.5, algunos de los cuales no son fácilmente accesibles de la inspección de la figura. Las pérdidas totales hasta el 2004 estaban en los 141,496,560 dólares, y en el 2003 por debajo de 201,797,340 dólares.

Como en el pasado, los demandados son generalmente incapaces o disuadidos para estimar las pérdidas en dólares. En la edición del 2004, solo 269 encuestados de un total de 494 proporcionaron estimaciones de las perdidas en dólares.

La categoría de virus surgió por primera vez como el tipo de incidente que genera las pérdidas más grandes (sustituyendo el robo de la información patentada, que había sido la categoría más cara en pérdidas durante cinco años consecutivos).

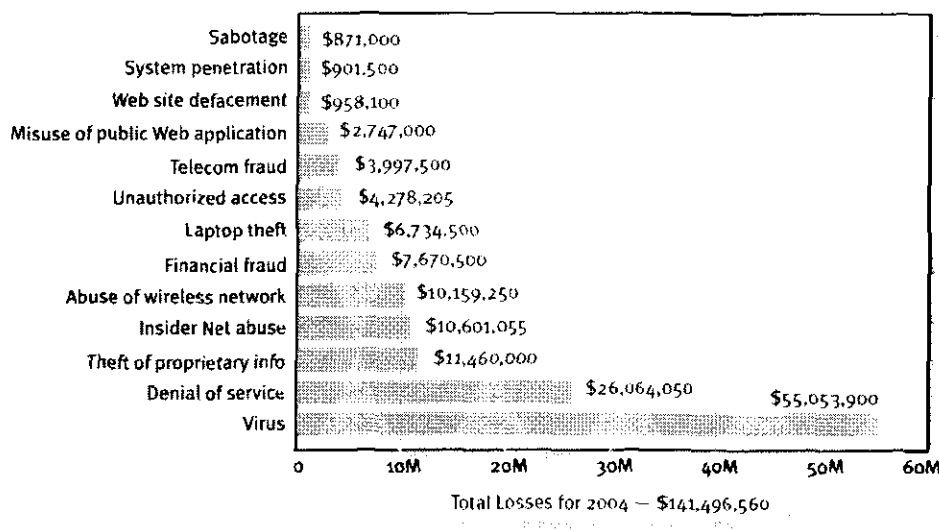


Figura 1.4 Cantidad de dólares perdidos por tipo [COM04].

1.2.2 US-CERT

United States Computer Emergency Readiness Team (US-CERT)

US-CERT es una sociedad encargada de mejorar la gestión de la seguridad informática y la respuesta a cyber-ataques en los Estados Unidos. US-CERT es responsable de:

- Analizar y reducir cyber amenazas y vulnerabilidades.
- Diseminar cyber amenazas como advertencias de información.
- Coordinar actividades de respuesta de incidentes.

US-CERT también proporciona una guía para ciudadanos, negocios, y otras instituciones para comunicar y coordinarse directamente con el gobierno de los Estados Unidos sobre la cyber seguridad.

La US-CERT ofrece una estadística sobre los reportes de incidentes que han registrado.

Estadísticas de incidentes en el 2004

Type	Jan	Feb	Mar	Apr	May	Jun
Root Compromiso	12	7	45	3	4	2
User Compromiso	4	7	18	144	6	4
Denial of Service	2	0	0	2	0	431
Malicious Code	3,645	71,970	666,343	642	1995	880,167
Web Site Defacement	14	0	1	3	4	6
Misuse of Resources	8	3	4	5	9	6
Other	697,934	326,128	1,166,141	378,113	1007	863789
Reconnaissance Activity	153,611	372,968	47,306	2,567	836	54,290,346
Monthly Totals	855,230	771,083	1,879,858	381,479	3861	56,034,751

Figura 1.5 tabla de incidentes por tipo en el 2004 [CER05].

Estadísticas de incidentes en el 2003

Type	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Yearly Totals
Root Compromise	11	10	25	30	7	3	6	17	6	6	11	5	137
User Compromise	3	8	3	7	6	2	4	530	10	1	2	11	587
Denial of Service	2	6	6	1	1	0	2	1	0	4	2	0	25
Malicious Code	24	161	16	7	12,006	206	24	14,993	687	38,280	29,170	95,732	191,306
Web Site Defacement	9	12	7	5	6	8	2	5	6	4	7	19	90
Misuse of Resources	1	0	0	1	0	0	2	0	0	0	1	21	26
Other	108	284	714	3	33	22	3	4	194	927	187,273	345,739	535,304
Reconnaissance	2,89	8,638	20,34	10,115	35,112	7,742	24,943	17,889	203,85	91,254	22,824	260,83	706,44

ce Activity	2		9						3			0	1
Monthly	3,05	9,119	21,12	10,169	47,171	7,983	24,986	33,439	204,75	130,47	239,29	702,35	1,433,
Totals	0		0						6	6	0	7	916

Figura 1.6 tabla de incidentes por tipo en el 2005 [CER05].

Esta información ha sido proporcionada para comunicar un panorama general sobre *variedad en las diversas actividades de intrusión que se ha reportado*. Las estadísticas de incidentes son presentadas sólo después de que las hemos analizado y clasificado. Dichas estadísticas no deben ser interpretadas como un registro cronológico de *actividades de intrusión*. Más de un tipo de actividad puede ser adjudicado a un incidente. Los incidentes complejos pueden implicar más actividades de intrusión de los que son explícitamente identificados. Los detalles de incidentes pueden no estar completos al 100 % o contienen el mismo nivel del detalle para cada incidente. El anfitrión y las cuentas de sitio son aproximados, y pueden no reflejar el número exacto de anfitriones implicados en el incidente [CER05].

1.3 Situación Actual.

Actualmente en la Universidad Popular Autónoma del Estado de Puebla (UPAEP) existe el laboratorio de servidores del departamento de Ingeniería Industrial y Tecnologías de Información (IITI). Este laboratorio cuenta con 2 servidores para uso de estudiantes y profesores del área de IITI, uno de ellos cuenta con el sistema operativo Linux el cual ofrece servicios entre los cuales se encuentran:

- Almacenamiento de trabajos prácticos en servidores (Investigación, tesis).
- Practicas en el sistema operativo Linux.
 - Apache
 - MySQL

- PHP
- Secure shell

Actualmente el acceso al servidor desde Internet no es respaldado por ninguna herramienta adicional y se puede tener acceso a el libremente desde cualquier lugar en Internet.

Es preciso que los usuarios del servidor tengan uso de los servicios las 24 horas del día los 365 días del año. Es por esto que es necesaria una propuesta integral de seguridad que salvaguarde la información que se encuentra en el servidor.

1.4 Objetivo del Proyecto.

Es necesario el contar con una herramienta que permita monitorear las actividades en el servidor y apoyar su administración. Así, se podrán ofrecer más servicios y se contará con información inmediata sobre actividades ya sean sospechosas o reglamentadas.

El objetivo principal de este proyecto consiste en diseñar, desarrollar e implantar una herramienta informática para apoyar en la administración de los recursos de software y hardware de un servidor Unix.

Alcances y limitaciones

El principal alcance del proyecto será el ofrecer una herramienta que mantenga el servidor del departamento de IITI libre de problemas que pudieran ser causados por actividades no permitidas.

La implantación en materia de seguridad informática estará limitada por los recursos e información que exista en la actualidad.

Capítulo 2. Estado del arte de la seguridad informática en Linux

Se presenta un análisis del estado del arte sobre la seguridad en el sistema operativo Linux. Se exponen brevemente algunas herramientas existentes.

2.1 Sistemas Operativos (S.O.).

El software responsable de asignar los recursos de un sistema, incluso memoria, tiempo de procesador, espacio de disco, y dispositivos periféricos. Todos los programas de aplicación usan el S.O. para obtener acceso a diferentes recursos del sistema cuando son necesarios. El SO es el primer programa cargado en la computadora cuando ésta inicializa, y permanece en la memoria en cualquier momento a partir de entonces. DOS, OS/2, Win9x*2000, Unix son algunos sistemas operativos[BAN01].

2.2 UNIX

Unix es diseñado para ser un poderoso sistema operativo multi-propósito que permita a múltiples usuarios usar una computadora al mismo tiempo, además de permitir a cada usuario el correr diferentes procesos. En el presente es implementado en una gran variedad de computadoras con diversas arquitecturas, incluyendo computadoras personales, estaciones de trabajo, minicomputadoras, mainframes, supercomputadoras y computadoras en paralelo [LEA94].

Unix se desarrolló de acuerdo a un extremadamente simple y elegante principio en su diseño: que dos tipos de objetos – archivos y procesos- compongan el sistema por completo [LEA94].

2.2.1 LINUX

Desarrollado por Linus Torvalds, Linux es un producto que imita la forma y la función de un sistema UNIX, pero no es sacado del código fuente autorizado. Más bien, fue

desarrollado independientemente; por un grupo de desarrolladores en una alianza informal sobre la red. Una ventaja principal es que el código fuente está disponible libremente (bajo el GNU copyleft), permitiendo técnicamente el cambiar y enmendar el sistema; esto también significa que hay muchas utilidades y drivers de especialistas disponibles en la red. Linux es una marca registrada de Linus Torvalds [BAN01].

2.3 Seguridad en Linux

A continuación se presentan las medidas de control, los posibles fallos si no se hace un manejo adecuado y algunas líneas a seguir para conservar una protección adecuada.

2.3.1 seguridad en el sistema de cuentas.

La forma más fácil para llevar a cabo un acceso no autorizado en un sistema es a través de la cuenta de otra persona, para evitar esto la medida de seguridad que presenta el sistema es el nombre de usuario y la contraseña. Es importante, para que la seguridad no se vea comprometida, controlar viejas cuentas, evitar contraseñas fácilmente "adivinables" y supervisar todo lo que pueda facilitar un acceso a una cuenta por parte de una persona que no sea su propietario [CUR90].

2.3.2 Control de los passwords.

El password es una de las medidas más importantes de seguridad en Linux. Si se consigue romper esta barrera el posible intruso adquiere la identidad del propietario. Esto es crucial en el caso de que se vea involucrada la identidad de root. Es fundamental un buen control de los passwords y de las posibles cuentas que no estén siendo usadas. Además es importante una buena política de elección.

Algunas de las líneas que los administradores de seguridad deben dar a los usuarios son las siguientes [CUR90]:

- No utilizar el nombre de la cuenta(login), tal cual, al revés, en mayúsculas.
- No utilizar nuestro nombre, ni apellidos.
- No utilizar nombres de familiares.
- No emplear información personal que pueda ser obtenida fácilmente.
- No emplear un password sólo con números o con la misma letra.
- No emplear palabras que se encuentren en los diccionarios.
- No utilizar palabras de menos de seis letras.
- Mezclar letras mayúsculas y minúsculas.
- Utilizar palabras que sean fáciles de recordar para que no haya que escribirlas.
- Emplear un password que pueda ser tecleado rápidamente sin necesidad de mirar al teclado.
- No escribirlos en ningún sitio, ni siquiera archivos.

El problema de no seguir estas reglas es que existen programas que van probando passwords hasta que encuentran el correcto, basados en un diccionario y combinando algunas palabras.

Una vez escogido un password es importante seguir una política adecuada, unas reglas generales pueden ser:

- Su único lugar debe ser la memoria de su dueño.
- No decirlos nunca a otra persona.
- Establecer un tiempo máximo de uso para cada uno.

2.3.3 Control de las cuentas

- **Cuentas con vida limitada.** Es posible que en algunos sitios, donde haya multitud de usuarios, existan viejas cuentas de personas que hayan abandonado la empresa. Estas cuentas suelen suponer un agujero de seguridad importante, no sólo porque alguien pueda entrar en ellas, sino porque si se produce un acceso no autorizado es muy difícil de detectar. Para evitar esto hay que poner fecha de expiración en todas las cuentas. La fecha puede almacenarse fácilmente en el archivo `/etc/shadow`. Este archivo está pensado para no dejar en el `/etc/passwd` los passwords encriptados de los usuarios. Puesto que UNIX requiere que sobre este archivo tengan derechos de lectura, los passwords se almacenan en el `/etc/shadow`. La línea completa que puede incluirse, para cada usuario, es la siguiente:

username:passwd:lastchg:min:max:warn:inactive:expire

lastchg indica el número de días desde el 1 de enero de 1970 hasta el último cambio de password.

min es el menor número de días entre cambios de password.

max máximo número de días entre cambios de password.

warn se da un aviso antes de que el password haya expirado.

inactive es el número de días que se permiten antes de que se bloquee la cuenta.

expire indica el momento en el que la cuenta va a expirar.

Mediante el comando `passwd`, se puede, cerrar una cuenta, bloquear un password, hacer que en el próximo acceso al sistema se cambie, o asignarles un tiempo de vida.

- **Cuentas multiusuario:** Hay que tener también un especial cuidado con las cuentas públicas [BAN01]. A ellas acceden usuarios que permanecen poco tiempo en el sistema. Lo mejor es crearlas para un fin concreto y borrarlas después. No deben tener palabras de acceso tan simples como "guest" o "visitor" y nunca deben permanecer en el archivo de passwords del sistema. También suelen plantear problemas de seguridad algunas cuentas que se instalan para ejecutar ciertos comandos sin acceder a la máquina. No tienen password, por tanto pueden ser usadas por cualquiera. Algunas de ellas tienen como identificador de usuario el cero. Estas cuentas prácticamente "invitan" a los crackers a entrar en ellas. La forma de acceder es sencilla, si un usuario puede acceder al sistema, reemplazando los comandos por unos suyos, podrá ejecutarlos con permisos de root.

2.3.4 Seguridad en el sistema de archivos

Cuando un sistema Unix arranca una de las tareas que obligatoriamente realiza, es incorporar diferentes sistemas de archivos (discos completos, una partición, una unidad de CD-ROM...) a la jerarquía de directorios Unix; este proceso se llama montaje, y para realizarlo generalmente se utiliza la orden mount. Es obligatorio montar al menos un sistema de archivos durante el arranque, el sistema raíz ("/), del que se desprenderán todos los demás[VIL12].

Montar un sistema de archivos significa asociar un determinado nombre de directorio, denominado punto de montaje (mount point) con el sistema en cuestión, de forma que al utilizar dicha ruta se trabajará sobre el sistema de archivos que se ha asociado a ella.

Para saber qué sistemas de archivos se deben de montar en el arranque de la maquina, y bajo qué nombre de directorio, linux utiliza el archivo `/etc/fstab`. Un ejemplo de este archivo es el siguiente:

```
root:~# cat /etc/fstab
/dev/hda3 /          ext2 defaults 1 1
/dev/hda4 /home      ext2 defaults 1 2
none /proc           proc defaults 1 1
root:~#
```

Cuando el sistema arranque, el archivo anterior indica que en `/dev/hda3` se encuentra el sistema de archivos raíz, de tipo `ext2` (habitual en Linux), y que se ha de montar con las opciones que se toman por defecto. La segunda línea nos dice que `/home` es un sistema diferente del anterior, pero del mismo tipo y que se montará con las mismas opciones; finalmente, la última entrada hace referencia al directorio `/proc/`, donde se encuentra un sistema de archivos especial que algunos Unix utilizan como interfaz entre estructuras de datos del núcleo y el espacio de usuario. Si cualquiera de las entradas anteriores fuera errónea, el sistema no arrancarían, o lo hará incorrectamente. Por lo que el archivo `/etc/fstab` debe ser modificable solo por el root [VIL12].

Lo anterior no suele representar ningún problema de seguridad; sin embargo diferentes problemas radican en una gestión incorrecta del montaje de sistemas de archivos. Por ejemplo, instalar ciertas utilidades que permitan gozar de los privilegios del root (por ejemplo, un *rootkit* o un simple *shell setuidado*); si guarda el archivo modificado en cualquier directorio del sistema, su localización será muy rápida: un comando como *find* alertará su presencia.

Pero puede suceder que se utilice un sistema de archivos oculto. Cuando se monta un sistema bajo un nombre de directorio, todo lo que había en ese directorio desaparece

de la vista, y es sustituido por el contenido del sistema montado; y dicho contenido no volverá a estar accesible hasta que se desmonte el sistema:

```
root:~# mount
/dev/hda3 on / type ext2 (rw)
/dev/hda4 on /home type ext2 (rw)
none on /proc type proc (rw)
root:~# ls /home/
ftp/ aglex/ lost+found/
root:~# umount /home
root:~# ls /home/
root:~#
```

Por otra parte, es posible desmontar una parte de la jerarquía de directorios, guardar ahí ciertos archivos, y volver a montar el sistema que había anteriormente. El comando `ncheck`, puede detectar estos archivos ocultos bajo un *mount point*; o desmontar manualmente los sistemas (a excepción del raíz) y comprobar que no hay nada oculto bajo ellos.

Otro problema de los sistemas de archivos viene de la necesidad de permitir a los usuarios (sin privilegios) montar y desmontar sistemas de archivos (discos exhibiles o CD-ROMs). Por ejemplo, en un laboratorio de máquinas Unix donde es deseable que todos los usuarios puedan acceder al floppy, para copiar prácticas realizadas en casa como para hacer una copia de las que se han hecho en el laboratorio. Unix permite dar una solución pero esta solución puede convertirse en una amenaza a la seguridad si no es implantada correctamente.

Al hablar de `/etc/fstab` hemos comentado el montaje con ciertas opciones tomadas por defecto; dichas opciones son[VIL12]:

- `rw` (se permite tanto la lectura como la escritura)
- `suid` (se permite la existencia de archivos *setuidados*).
- `dev` (se permite la existencia de dispositivos).
- `exec` (se permite la ejecución de binarios).
- `auto` (el sistema se monta automáticamente al arrancar o utilizar `mount -a`).
- `noauto` (el sistema no se monta automáticamente).
- `nouser` (sólo puede ser montado por el root).
- `async` (la entrada/salida sobre el dispositivo se realiza de forma asíncrona).

Es imprescindible utilizar ``nodev'`, de forma que si en el sistema montado existen archivos de tipo dispositivo (archivos que haga referencia a nuestros discos duros) ese archivo sea ignorado; en caso contrario, cualquiera podría acceder directamente a nuestro hardware.

También es importante especificar ``nosuid'`, de forma que se ignore el bit de `setuid` en cualquier archivo contenido en el sistema que el usuario monta: así evitamos que el usuario consiga privilegios de administrador en nuestro sistema.

Especificar ``noexec'`, permite que no se pueda ejecutar nada de lo que está en el dispositivo montado. De esta forma, si queremos que los usuarios puedan montar por ejemplo la unidad de floppy, una entrada correcta en `/etc/fstab` sería la siguiente [VIL12]:

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

2.3.5 Llenado de disco.

Un disco lleno implica muchos problemas para la máquina, desde correo electrónico que no se recibe, logs que no se registran, o una negación de servicio para el resto de usuarios, que no podrán almacenar nada. Aunque algunos Unix reservan una parte de cada disco o partición para escritura solo del root o de procesos que corran bajo el UID 0 (aproximadamente un 10% de su capacidad), no es posible confiar ciegamente mecanismos de este tipo para mantener segura la máquina; por esto es recomendable lo siguiente [VIL12]:

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

Se puede comprobar, si un usuario lanza un ftp en background desde su \$HOME durante la noche (proceso que llena gran cantidad de disco), en todo caso podría afectar al resto de usuarios, pero nunca al sistema en global (correo, logs, root. . .); este tipo de problemas no suelen ser ataques, sino mas bien descuidos de los usuarios que no tienen en cuenta el espacio disponible antes de descargar archivos de forma no interactiva. Si queremos que ni siquiera pueda afectar al resto de usuarios, podemos establecer un sistema de cuotas de disco en la máquina.

2.3.6 Permisos de un archivo.

La seguridad en el sistema de archivos es fundamental para la protección de los datos de acceso no autorizados. Las barreras de entrada que nos encontramos en el sistema de archivos son los derechos sobre los mismos[RIT79].

Cada archivo o directorio de nuestro sistema tiene asociado 3 conjuntos de bits asociados a los permisos: el primero del dueño del archivo (UID), el segundo para el grupo al que pertenece (GID) y el tercero para el resto de los usuarios. cada una de ellas indica la existencia o la ausencia de permiso para leer, escribir o ejecutar el archivo: una r indica un permiso de lectura, una w de escritura, una x de ejecución y un '-' indica que el permiso correspondiente no está activado. Así, si en una de las ternas se tienen los caracteres rwx, el usuario o usuarios afectados por esa terna tiene o tienen permisos para realizar cualquier operación sobre el archivo. En cada caso el significado en el primer bit (izquierda) se representa si hace referencia a un directorio con una 'd' o a un archivo con un '-'[VIL12][RIT79].

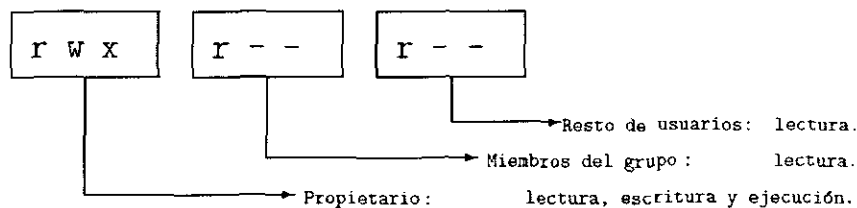


Figura 2.1 permisos de un archivo [VIL12].

Cuando un usuario intenta acceder en algún modo a un archivo, el sistema comprueba qué terna de permisos es la aplicable y se basa únicamente en ella para conceder o denegar el acceso. De esta forma es posible tener situaciones tan curiosas como la de un usuario que no tenga ningún permiso sobre uno de sus archivos, y en cambio que el resto de usuarios del

sistema pueda leerlo, ejecutarlo o incluso borrarlo; obviamente, esto no es lo habitual, y de suceder el propietario siempre podrá restaurar los permisos a un valor adecuado[VIL12].

El propietario y el grupo de un archivo se pueden modificar con los comandos `chown` y `chgrp` respectivamente; Ambas reciben como parámetros al menos el nombre de usuario o grupo (los nombres validos de usuario son los que poseen una entrada en `/etc/passwd` mientras que los grupos validos se leen de `/etc/group`) al que se puede otorgar la posesión del archivo, así como el nombre de archivo a modificar[FRI91]:

```
neo:~# ls -l /tmp/fichero
-rw-r--r-- 1 root other 799 Feb 8 19:47 /tmp/fichero
neo:~# chown toni /tmp/fichero
neo:~# ls -l /tmp/fichero
-rw-r--r-- 1 toni other 799 Feb 8 19:47 /tmp/fichero
neo:~# chgrp staff /tmp/fichero
neo:~# ls -l /tmp/fichero
-rw-r--r-- 1 toni staff 799 Feb 8 19:47 /tmp/fichero
neo:~#
```

Es posible cambiar a la vez el propietario y el grupo de un archivo mediante `chown`, separando ambos mediante un carácter especial, generalmente `:` o `.`[FRI91]:

```
neo:~# ls -l /tmp/fichero
-rw-r--r-- 1 root other 799 Feb 8 19:47 /tmp/fichero
neo:~# chown toni:staff /tmp/fichero
neo:~# ls -l /tmp/fichero
-rw-r--r-- 1 toni staff 799 Feb 8 19:47 /tmp/fichero
neo:~#
```

Ninguno de estos comandos altera el campo de permisos; para modificar los permisos de un archivo se utiliza la orden `chmod`. Este comando generalmente recibe como

parámetro el permiso en octal que queremos asignar a cierto archivo, así como el nombre del mismo:

```
neo:~# ls -l /tmp/fichero
-rw-r--r-- 1 root staff 799 Feb 8 19:47 /tmp/fichero
neo:~# chmod 755 /tmp/fichero
neo:~# ls -l /tmp/fichero
-rwxr-xr-x 1 root staff 799 Feb 8 19:47 /tmp/fichero
neo:~#
```

para usar el comando `chmod` es necesario hacer conversiones de sistema octal a binario[OLE05]. Una vez que se tiene la representación octal de los bits iniciales, y es necesario asignar esta terna a un archivo, se debe ejecutar el comando `chmod` indicándole este número y el nombre del archivo:

```
neo:~# chmod 643 /tmp/fichero
neo:~# ls -l /tmp/fichero
-rw-r--wx 1 root root 799 Feb 8 19:47 /tmp/fichero*
neo:~#
```

Una norma de seguridad en el sistema de archivos es dar los permisos adecuados a cada archivo y usuario. Normalmente los archivos se crean sin tener en cuenta los derechos que se les otorgan [BAN01].

Una forma sencilla para controlar esto es utilizar el comando `umask`, así pues:

Con `umask 022` los archivos tendrán los permisos: `- r w - r - - r - -`

Con `umask 077` crea los archivos con los permisos: `- r w - - - - - - -`

Hay que asegurar que estas líneas están incluidas en el `.cshrc`.

2.3.7 Métodos de acceso SUID, SGID y Sticky bit.

Los permisos de los archivos en Unix se relacionan con un número en octal que varia entre 000 y 777; sin embargo, existen algunos permisos especiales que hacen variar ese numero entre 0000 y 7777: se trata de los bits de permanencia (Sticky bit) (1000), sgid (2000) y suid (4000) [BAN01] [VIL12].

El bit de suid o setuid se activa sobre un archivo añadiéndole 4000 a la representación octal de los permisos del archivo y otorgándole además permiso de ejecución al propietario del mismo; al hacer esto, en lugar de la x en la primera terna de los permisos, aparecería una s o una S si no se ha otorgado el permiso de ejecución correspondiente (en este caso el bit no tiene efecto):

```
neo:~# chmod 4777 /tmp/file1
neo:~# chmod 4444 /tmp/file2
neo:~# ls -l /tmp/file1
-rwsrwxrwx 1 root other 0 Feb 9 17:51 /tmp/file1*
neo:~# ls -l /tmp/file2
-r-Sr--r-- 1 root other 0 Feb 9 17:51 /tmp/file2*
neo:~#
```

El bit suid activado sobre un archivo indica que todo aquel que ejecute el archivo va a tener durante la ejecución los mismos privilegios que quien lo creo.

Todo lo que anterior con respecto al bit setuid es aplicable al bit setgid pero a nivel de grupo del archivo en lugar de propietario.

Pero, ¿cómo afecta todo esto a la seguridad del sistema?: los bits de setuid y setgid dan a Unix una gran flexibilidad, pero constituyen al mismo tiempo la mayor fuente de ataques internos al sistema (aquellos realizados por un usuario autorizado o no con el objetivo de modificar sus privilegios). Cualquier sistema Unix tiene un cierto número de ejecutables. Cada uno de ellos, se ejecuta con los privilegios de quien lo creó lo que directamente implica que cualquier usuario tiene la capacidad de lanzar tareas que escapen total o parcialmente al control del sistema operativo; se ejecutan en modo privilegiado si es el administrador quien creó los ejecutables. Si por cualquier motivo un programa falla, se asume inmediatamente que presenta un problema de seguridad para la máquina, y se recomienda resetear el bit de setuid cuanto antes[VIL12].

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

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

Si el bit de permanencia de un archivo está activado, se está indicando al sistema operativo que se trata de un archivo muy utilizado, por lo que es conveniente que permanezca en memoria principal el mayor tiempo posible; esta opción se utilizaba en sistemas antiguos que disponían de muy poca RAM, pero hoy en día prácticamente no

se utiliza. Lo que si que sigue vigente es el efecto del sticky bit activado sobre un directorio: en este caso se indica al sistema operativo que, aunque los permisos “normales” digan que cualquier usuario pueda crear y eliminar archivos, sólo el propietario de cierto archivo y el administrador pueden borrar un archivo guardado en un directorio con estas características. Esto se utiliza principalmente en directorios del sistema de archivos en los que interesa que todos puedan escribir pero que no todos puedan borrar los datos escritos, como /tmp/ o /var/tmp/: si el equivalente octal de los permisos de estos directorios fuera 777 en lugar de 1777, cualquier usuario podría borrar los archivos del resto.

2.3.8 listas de control de acceso: ACLs

Las listas de control de acceso (ACLs, Access Control Lists) proveen de un nivel adicional de seguridad a los archivos extendiendo el clásico esquema de permisos en Unix[VIL12]: mientras que con estos últimos solo podemos especificar permisos para los tres grupos de usuarios habituales (propietario, grupo y resto), las ACLs van a permitir asignar permisos a usuarios o grupos concretos; por ejemplo, se pueden otorgar ciertos permisos a dos usuarios sobre algunos archivos sin necesidad de incluirlos en el mismo grupo. Este mecanismo está disponible en la mayoría de los sistemas Unix (Solaris, AIX, HP-UX..), desde hace mas de 10 años [COM88].

Si se desea saber si a un usuario se le permite cierto tipo de acceso sobre un archivo es necesario un listado largo:

```
neo:~# ls -l /usr/local/sbin/nessud
-rwxr-xr-x  1 root  root   1621158 ago 12 2004 usr/local/sbin/nessud
neo:~#
```

Viendo el resultado, se sabe que el archivo `nessusd` puede ser ejecutado, modificado y leído por el administrador, pero por nadie más; sin embargo, no conocemos el estado de la lista de control de acceso asociada al archivo. Para ver esta lista, se utiliza el comando `getfacl`:

```
neo:/# getfacl /usr/local/sbin/nessusd
# file: /usr/local/sbin/nessusd
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
neo:/#
```

En este listado se muestra el nombre del archivo, su propietario y su grupo, todos precedidos por ``#'`. Además la lista de control: los campos `user`, `group` y `other` son básicamente la interpretación que `getfacl` hace de los permisos del archivo. El campo `mask` es muy similar al `umask` clásico: define los permisos máximos que un usuario (a excepción del propietario) o grupo puede tener sobre el archivo. Finalmente, el campo `effective` nos dice, para cada usuario (excepto el propietario) o grupo el efecto que la máscara tiene sobre los permisos: es justamente el campo que tenemos que analizar si queremos ver quién puede acceder al archivo y de qué forma[VIL12].

Lo interesante de cara a la protección de archivos es extender los permisos clásicos del mismo, modificando su lista asociada con el comando `setfacl`:

```
neo:~# setfacl -m user:aglex:r-x /usr/local/sbin/nessusd
neo:~# getfacl /usr/local/sbin/nessusd
# file: /usr/local/sbin/nessusd
```

```
# owner: root
# group: bin
user::rwx
user:aglex:r-x #effective:---
group:--- #effective:---
mask:---
other:---
neo:~#
```

Como se aprecia se puede modificar la lista de control de acceso del archivo para asignarle a aglex permiso de ejecución y lectura sobre el mismo. El comando `setfacl` se utiliza principalmente de tres formas:

- Añadir entradas a la ACL, mediante la opción `-m` seguida de las entradas que se deseen añadir separadas por comas.
- Utilizar el parámetro `-s` para reemplazar la ACL completa (las entradas, separadas también por comas).
- Borrar las entradas de la lista con la opción `-d`.

Cada entrada de la ACL tiene el siguiente formato:

Tipo: UID|GID: permisos

El tipo indica a quien aplicar los permisos (por ejemplo, `user` para el propietario del archivo, o `mask` para la mascara), el `UID` indica el usuario al que queremos asociar la entrada (como hemos visto, se puede utilizar también el `login`, y el `GID` hace lo mismo con el grupo (de la misma forma, se puede especificar su nombre simbólico). Finalmente, el campo de permisos hace referencia a los permisos a asignar, y puede ser especificado mediante símbolos `rwx-` o de forma octal.

Se indico que el usuario aglex tenga permiso de lectura y ejecución en el archivo; sin embargo, si ahora este usuario intenta acceder al archivo en estos modos obtendrá un error:

```
neo:/usr/local/sbin$ id
uid=100(toni) gid=10(staff)
neo:/usr/local/sbin$ ./nessusd
bash: ./nessusd: Permission denied
neo:/usr/local/sbin$
```

lo que sucedió es que la máscara esta actuando sobre sus permisos. Para solucionar esto se ha de modificar el campo mask:

```
neo:~# setfacl -m mask:r-x /usr/local/sbin/nessusd
neo:~#
```

Ahora aglex podra acceder al archivo para leerlo o ejecutarlo.

```
neo:/usr/local/sbin$ id
uid=100(aglex) gid=10(staff)
neo:/usr/local/sbin$ ./nessusd
/etc/nessusd_config: No such file or directory
```

Aunque obtenga un error, este error ya no depende de la protección de los archivos sino de la configuración del programa. aglex no podrá utilizar recursos a los que no le está permitido el acceso, como puertos bien conocidos, otros archivos, o procesos que no le pertenezcan. Hay que recordar que aunque un usuario ejecute un archivo perteneciente al root, los privilegios del usuario no cambian.

2.3.9 Almacenamiento seguro.

Cuando la integridad de una computadora ha sido puesta en evidencia, una alternativa es evitar que la información sea utilizada aplicando instrucciones de encriptación.

La orden `crypt` permite cifrar y descifrar archivos; si no recibe parámetros lee los datos de la entrada estándar y los escribe en la salida estándar[BAN01]. Un ejemplo simple de su uso puede ser el siguiente:

```
$ crypt <fichero.txt >fichero.crypt
```

Enter key:

\$

En el anterior ejemplo hemos cifrado utilizando `crypt` el archivo `fichero.txt` y guardado el resultado en `fichero.crypt`; el original en texto mantiene en su directorio.

Para descifrar un archivo cifrado mediante `crypt` se utiliza la misma clave:

```
$ crypt <fichero.crypt>salida.txt
```

Enter key:

\$

El anterior comando ha descifrado `fichero.crypt` con la clave tecleada y guardado el resultado en el archivo `salida.txt`, que coincidiría en contenido con el anterior `fichero.txt`.

`crypt` no se debe utilizar nunca para cifrar información confidencial; la seguridad del algoritmo de cifra utilizado por esta orden es mínima, ya que `crypt` se basa en una

máquina con un rotor de 256 elementos similar en muchos aspectos a la alemana Enigma, con unos métodos de ataque rápidos y conocidos por todos ([RW84]).

2.3.10 Informática forense.

“Best way to remove e-mail from a hard drive is to hit with a sledge hammer and throw it into a furnace.” John Patzakis, President & Chief Legal Officer Guidance Software

La informática forense es descrita como una autopsia al disco duro de una computadora debido a la especialización de las herramientas de software, y el uso de técnicas aplicadas para analizar los diferentes niveles de información recopilada después del acontecimiento[BAR 05].

La informática forense es un campo que día a día toma importancia; de la misma forma que la medicina forense es capaz de extraer información valiosa de un cadáver, la informática forense pretende extraer información de un 'cadáver' computarizado (por ejemplo, un sistema en el que un intruso ha borrado sus huellas), también incluso mucho después de haber 'muerto' (esto es, haber sido borrado)[VIL12]. Aunque las técnicas de recuperación de datos en Unix se aplican habitualmente para potenciar la seguridad de un equipo, éstas mismas técnicas utilizadas por un atacante pueden llegar a comprometer gravemente la seguridad del sistema: un intruso que haya conseguido cierto nivel de privilegio puede recuperar, por ejemplo, el texto plano de un documento que un usuario haya cifrado y posteriormente borrado. Aunque esto no es tan trivial como en otros sistemas menos seguros (en los que incluso se facilitan herramientas tipo undelete), parece claro que este tipo de actividades constituyen una amenaza a la seguridad de cualquier sistema operativo.

Existen diferentes tipos de información que se puede recuperar de una computadora, he aquí algunos ejemplos:

- Recuperar cientos de e-mails del disco duro.
- Recuperar información años después de que un empleado dejó la empresa.
- Obtener información después de que el disco duro ha sido formateado.
- Recuperar archivos incluso si el disco duro ha tenido otro uso hace algunos meses.

El análisis forense, especialmente la recuperación de datos, es sumamente importante a la hora de analizar los alcances de una intrusión a un equipo. En estas situaciones, es muy posible que el atacante modifique archivos log (cuando no los borra completamente), troyanize programas o ejecute procesos determinados: es aquí donde la persona encargada de retornar al sistema a la normalidad debe desconfiar de cuanto la maquina diga, y recurrir al análisis forense para determinar el impacto real del ataque y devolver el sistema a un correcto funcionamiento.

2.4 Seguridad en Redes

*La rápida expansión y popularización de Internet ha convertido a la seguridad en redes en uno de los tópicos más importantes dentro de la informática moderna. Con tal nivel de interconexión, los virus y los *hackers* acampan a sus anchas, aprovechando las deficientes medidas de seguridad tomadas por administradores y usuarios a los que esta nueva revolución ha tomado por sorpresa[BAN01].*

Las ventajas de las redes en informática son evidentes, pero muchas veces se menosprecian ciertos riesgos, circunstancias que a menudo ponen en peligro la seguridad de los sistemas. En unos pocos años la inmensa mayoría de las empresas

operarían a través de la red, y esto solo sería posible si los profesionales de la informática saben aportar soluciones que garanticen la seguridad de la información.

Cuando se conecta una máquina a una red, al mismo tiempo que se abre el sistema a innumerables fuentes de información y servicios [BAN01], queda también expuesta a todos los posibles ataques desde el exterior. El control de la red es uno de los aspectos fundamentales de seguridad, puesto que la mayoría de los intentos de ataque se llevan a cabo a partir de posibles agujeros existentes en los sistemas y servicios de red.

2.4.1 Escaneo de puertos.

Una de las primeras actividades que un potencial (o no tan potencial) atacante realizaría contra su objetivo sería sin duda un escaneo de puertos, un portscan; esto le permitiría obtener en primer lugar información básica acerca de qué servicios ofrecen nuestras máquinas y, adicionalmente, otros detalles del entorno como qué sistema operativo están instalados en cada host o ciertas características de la arquitectura de la red. Analizando qué puertos están abiertos en un sistema, el atacante puede buscar agujeros en cada uno de los servicios ofrecidos[LUC99].

Telnet

Comprobar el estado de un determinado puerto es a priori una tarea sencilla; incluso es posible llevarla a cabo desde la línea de comandos, usando una herramienta tan genérica como telnet. Por ejemplo, si se desea conocer el estado del puerto 5000 en la máquina cuya dirección IP es 192.168.0.10; si el telnet ofrece una respuesta a la petición realizada a dicho puerto, entonces está abierto y escuchando peticiones:

```
neo:~$ telnet 192.168.0.10 5000
```

```
Trying 192.168.0.10...
Connected to 192.168.0.10.
Escape character is '^]'.
^D
Connection closed by foreign host.
neo:~$
```

Si por el contrario el puerto está abierto pero en él no hay ningún demonio atendiendo peticiones, la respuesta sería una negativa de conexión. Por último, si el puerto está protegido por un firewall, lo más probable es que no obtengamos respuesta alguna; el telnet lanzado se quedaría intentando la conexión hasta que se produzca un timeout o hasta que lo paremos manualmente.

Open

El escaneo open se basan en el establecimiento de una conexión tcp completa [TAN03], por lo que son muy sencillos de detectar y detener. Utilizan la llamada `connect()`, siendo lo más similar al ejemplo del telnet: el escaneador intenta establecer una conexión con un puerto concreto del host atacado, y en función de la respuesta obtenida conoce su estado: *una técnica rápida, sencilla, fiable y que no necesita de ningún privilegio especial en la máquina atacante.*

Existen diferentes técnicas de escaneo entre las cuales e encuentran:

- half-open.
- dumb scanning
- stealth scanning. [KLA95]

Actualmente el significado del termino ha cambiado, ya que lo que Klaus [KLA95] presentó se denomina hoy en día half-open scanning, y por stealth scanning se conoce a una familia de técnicas de escaneo que cumplen alguna de las siguientes condiciones:

- Eludir firewalls o listas de control de acceso.
- No ser registradas por sistemas de detección de intrusos, ni orientados a red ni en el propio host escaneado.
- Simular tráfico normal y real para no levantar sospechas ante un analizador de red.

Spoofing

Por spoofing se conoce a la creación de tramas tcp/ip utilizando una dirección IP falsa; la idea de este ataque es muy sencilla: desde su equipo, un pirata simula la identidad de otra máquina de la red para conseguir acceso a recursos de un tercer sistema que ha establecido algún tipo de confianza basada en el nombre o la dirección IP del host suplantado. Y como los anillos de confianza basados en estas características tan fácilmente falsificables son aún demasiado abundantes, el spoofing sigue siendo en la actualidad un ataque no trivial, pero factible contra cualquier tipo de organización [HEB96].

Existen otros ataques de impostura relacionados en mayor o menor medida con este, entre los que destacan el DNS Spoofing, el ARP Spoofing y el Web Spoofing [RIS01]. Para finalizar este punto, se comentara brevemente sobre los mismos:

- **DNS Spoofing:** Este ataque hace referencia al falseamiento de una dirección IP ante una consulta de resolución de nombre (esto es, resolver con una dirección falsa un cierto nombre DNS), o viceversa (resolver con un nombre falso

una cierta dirección IP). Esto se puede conseguir de diferentes formas, desde modificar las entradas del servidor encargado de resolver una cierta petición para falsear las relaciones dirección-nombre, hasta comprometer un servidor que infecte la cache de otro (lo que se conoce como DNS Poisoning).

- ARP Spoofing: Hace referencia a la construcción de tramas de solicitud y respuesta arp falseadas, de forma que en una red local se puede forzar a una determinada maquina a que envíe los paquetes a un host atacante en lugar de hacerlo a su destino legítimo[TAN03].
- Web Spoofing: Este ataque permite a un pirata visualizar y modificar cualquier página web que su víctima solicite a través de un navegador, incluyendo las conexiones seguras vía ssl. Para ello, mediante código malicioso un atacante crea una ventana del navegador correspondiente, de apariencia inofensiva, en la máquina de su víctima; a partir de ahí, enruta todas las páginas dirigidas al equipo atacado a través de su propia máquina, donde son modificadas para que cualquier evento generado por el cliente sea registrado [FEL96].

2.4.2 Seguridad en la Web.

Durante los últimos años los servidores Web se han convertido en una excelente fuente de diversión para piratas. Hoy en día le basta con teclear el nombre de su objetivo en un navegador y añadir '.com' detrás del mismo para contactar con al menos una de sus máquinas: su servidor Web[VIL12].

Los ataques a las páginas Web de una organización son casi siempre los más 'vistosos' que la misma puede sufrir: en cuestión de minutos piratas de todo el mundo se enteran de cualquier problema en la página Web de una empresa más o menos grande pueda estar sufriendo, y si se trata de una modificación de la misma incluso existen recopilatorios de páginas 'hackeadas'.

La mayor parte de estos ataques tiene éxito gracias a una configuración incorrecta del servidor o a errores de diseño del mismo: Servidores Web bastante complejos (alta disponibilidad, balanceo de carga, sistemas propietarios de actualización de contenidos) y difíciles de administrar correctamente, mientras que si la empresa es pequeña es muy posible que haya elegido un servidor Web simple en su instalación y administración pero en el cual es casi imposible garantizar una mínima seguridad. Sea por el motivo que sea, cada día es más sencillo para un pirata ejecutar ordenes de forma remota en una máquina, o al menos modificar contenidos de forma no autorizada, gracias a los servidores Web que un sistema pueda albergar.

Cualquier analizador de vulnerabilidades que se pueda ejecutar contra los sistemas (nessus, ISS Security Scanner, NAI CyberCop Scanner. . .) es capaz de revelar información que va a resultar útil a la hora de reforzar la seguridad de los servidores Web. incluso existen analizadores que están diseñados para auditar únicamente este servicio, como whisker. Ejecutando este comando contra una máquina es posible obtener resultados similares a los siguientes:

```
neo:~/security/whisker$ ./whisker.pl -h luisa
-- whisker / v1.4.0 / rain forest puppy / www.wiretrip.net --
= Host: neo
= Server: Apache/1.3.19 (Unix) PHP/4.0.4pl1 mod_ssl/2.8.2 OpenSSL/0.9.5a
+ 200 OK: HEAD /docs/
+ 200 OK: HEAD /cgi-bin/Count.cgi
+ 200 OK: HEAD /cgi-bin/textcounter.pl
+ 200 OK: HEAD /ftp/
+ 200 OK: HEAD /guestbook/
+ 200 OK: HEAD /usage/
neo:~/security/whisker$
```

Es posible observar que el servidor nos proporciona excesiva información sobre su configuración (versión, módulos, etc. . .), y que la herramienta ha obtenido algunos archivos y directorios que pueden resultar interesantes para un atacante: en el caso de los CGI no se tiene más que acercarse a alguna base de datos de vulnerabilidades para obtener información sobre los posibles problemas de seguridad que pueda presentar[VIL12].

Para eliminar los problemas de seguridad, una medida elemental es eliminar del servidor cualquier directorio o CGI de ejemplo que se instale por defecto. Muchos servidores incorporan programas que no son necesarios para el correcto funcionamiento del software, y que en ciertos casos abren enormes agujeros de seguridad.

Otra medida de seguridad básica es deshabilitar el Directory Indexing que por defecto muchos servidores incorporan: la capacidad de obtener el listado de un directorio cuando no existe un archivo `index.html` o similar en el mismo.

2.4.3 Puertos.

Cuando maneja información que pasa de un sistema de computadoras a otro, esta información se transmite desde una aplicación servidor a una aplicación cliente. Cada programa servidor tendrá un puerto predeterminado al que se acopla (por el que escucha) esperando peticiones que vengan de un cliente. El cliente a su vez está diseñado para enviar peticiones al puerto predeterminado del servidor.

Así trabajan los puertos. La autoridad de asignación de números de Internet, ha reservado determinados puertos a los que ciertos servicios deben acoplarse. Algunos de estos puertos son [BAN01]:

PuertoServicio

21	ftp
23	telnet
25	mail (SMTP)
80	http (WWW)
110	pop3

2.5 Virus en Linux.

Un virus es una secuencia de código que se inserta en un archivo ejecutable denominado host, de forma que al ejecutar el programa también se ejecuta el virus; generalmente esta ejecución implica la copia del código viral (o una modificación del mismo) en otros programas. El virus necesita obligatoriamente un programa donde insertarse para poderse ejecutar, por lo que no se puede considerar un programa o proceso independiente.

Durante años, un debate típico entre la comunidad de la seguridad informática es la existencia de virus en Unix [Rad95]. ¿Existen virus en este entorno, o por el contrario son un producto de otros sistemas en los que el concepto de seguridad se pierde? Realmente existen virus sobre plataformas Unix capaces de reproducirse e infectar archivos, tanto ELF como *shellscripts*: ya en 1983 Fred Cohen diseñó un virus que se ejecutaba con éxito sobre Unix en una VAX 11-750 [COH84]; incluso puede encontrar el código necesario para la infección [DUF89].

Parece claro que la existencia de virus en Unix es algo sobradamente comprobado; entonces, ¿dónde está el debate? La discusión se centra en hasta qué punto un virus para Unix puede comprometer la seguridad del sistema; generalmente, la existencia de estos virus y sus efectos no suelen ser muy perjudiciales en los sistemas Unix de hoy en día. Se suele tratar de código escrito únicamente como curiosidad científica, ya que cualquier acción que realice un virus es en general más fácilmente realizable por otros medios como un simple exploit; de hecho, uno de los primeros virus para Unix (en términos puristas se podría considerar un troyano más que un virus) fué creado por uno de los propios diseñadores del sistema operativo, Ken Thompson ([THO84]), con el fin no de dañar al sistema, sino de mostrar hasta qué punto se puede confiar en el software de una máquina.

2.6 Criptografía.

Según el Diccionario de la Real Academia, la palabra Criptografía proviene del griego *criptos* que significa oculto, y *grafía*, que significa escritura, y su definición es: *Arte de escribir con clave secreta o de un modo enigmático*". Obviamente la criptografía hace años que dejó de ser un arte para convertirse en una técnica, o más bien un conglomerado de técnicas, que tratan sobre la protección (ocultamiento frente a observadores no autorizados) de la información. Entre las disciplinas que engloba cabe destacar la teoría de la información, la teoría de números, y la complejidad algorítmica [LUC99].

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

que se ha determinado la presencia de información cifrada en un dispositivo de almacenamiento. Esto, que puede parecer algo exagerado, no lo es en absoluto: todos los expertos en criptografía coinciden en afirmar que los métodos de ataque más efectivos contra un sistema encriptado no son los efectuados contra el algoritmo, sino contra las personas (chantaje, amenazas, presiones judiciales. . .) [VIL12]. Intentando dar solución a este problema, prestigiosos investigadores de la talla de Roger Needham, Ross Anderson o Adi Shamir ([AND98]) han establecido las bases de sistemas seguros basados en modelos esteganográficos, con desarrollos especialmente importantes sobre plataformas Linux [MCD99], [VAN98]. La disponibilidad del código fuente completo de este clon de Unix unida a su política de desarrollo ha propiciado enormemente estos avances, hasta el punto de que existen en la actualidad sistemas de archivos basados en esteganografía que se insertan en el kernel igual que lo hace un sistema normal como *ufs* o *nfs*, o que se añaden a *ext2* proporcionando funciones de cifrado.

La idea es sencilla: si por ejemplo tenemos cinco archivos cifrados con una aplicación como *pgp*, cualquier atacante con acceso al dispositivo y que haga algunas operaciones sobre archivos puede determinar que tenemos exactamente esos archivos cifrados; con esta información, su labor para obtener la información esta muy clara: se ha de limitar a obtener las cinco claves privadas usadas para cifrar los archivos. Conocer el número exacto es de una ayuda incalculable para el atacante. Con los sistemas esteganográficos, a pesar de que es imposible ocultar la existencia de cierta información cifrada, alguien que la inspeccione no va a poder determinar si la clave de descifrado que el propietario le ha proporcionado otorga acceso a toda la información o sólo a una parte de la misma.

Un atacante que no posea todas las claves no va a poder descifrar todos los archivos, y lo más importante: no va a poder saber ni siquiera si otros archivos aparte de aquellos a los que ha accedido existen o no, aunque posea un acceso total al software y al soporte

físico. Para conseguir esto se utiliza una propiedad de ciertos mecanismos de seguridad denominada *plausible deniability*, algo que se vendría a traducir como 'negación creíble'; dicha propiedad permitiría a un usuario negar de forma creíble que en un dispositivo exista más información cifrada de la que ya se ha podido descubrir, o que cierta transacción se haya llevado a cabo. Volviendo al ejemplo de pgp, el usuario podría revelar la clave de cifrado de sólo uno o dos de los archivos, aquellos que no considere vitales, ocultando las claves y la existencia del resto sin que el atacante sea capaz de determinar que la información accedida no es toda la existente.

Criptografía clásica.

El ser humano siempre ha tenido secretos de muy diversa índole, y ha buscado mecanismos para mantenerlos fuera del alcance de miradas indiscretas.

En esta sección se hace un breve repaso de los mecanismos criptográficos considerados *clásicos*. Podemos llamar así a todos los sistemas de cifrado anteriores al nacimiento de las computadoras. Estas técnicas tienen en común que pueden ser empleadas usando simplemente lápiz y papel, y que pueden ser criptoanalizadas casi de la misma forma. De hecho, con la ayuda de las computadoras, los mensajes cifrados empleando estos códigos son fácilmente descifrables, por lo que cayeron rápidamente en desuso.

La transición desde la criptografía clásica a la moderna se da precisamente durante la II Guerra Mundial, cuando el Servicio de Inteligencia aliado *rompe* la maquina de cifrado del ejercito alemán, llamada ENIGMA.

Todos los algoritmos criptográficos clásicos son simétricos, ya que hasta mediados de los años setenta no nació la criptografía asimétrica, y por esa razón se hace referencia a los algoritmos de llave privada.

Existen diversas clasificaciones para los algoritmos de encriptación, a continuación se presentan algunos y sus características mas generales [GAR95]:

2.6.1 Algoritmos Simétricos.

Un sistema de cifrado simétrico es un tipo de cifrado que usa una misma clave para cifrar y para descifrar. Las dos partes que se comunican mediante el cifrado simétrico deben estar de acuerdo en la clave a usar de antemano. Una vez de acuerdo, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra usando la misma clave. Algunos ejemplos de algoritmos simétricos son:

DES (estándar de encriptación de datos)

El algoritmo DES codifica bloques de 64 bits empleando claves de 56 bits. Es una red de Feistel de 16 rondas, mas dos permutaciones, una que se aplica al principio (P_i) y otra que se aplica al final (P_f), tales que $P_i = P_f^{-1}$.

Sin embargo existen problemas de las claves simétricas:

Debe haber un intercambio previo, por medio de un método seguro, La comunicación en múltiples direcciones no resulta factible: Si muchos usuarios se deben comunicar con el servidor, comprometer a cualquiera de ellos puede implicar comprometerlos a todos

2.6.2 Algoritmos de llave pública.

Los sistemas de cifrado de clave pública se inventaron con el fin de evitar por completo el problema del intercambio de claves. Un sistema de cifrado de clave pública usa un

par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona a la que se ha enviado el mensaje. Una clave es *pública* y se puede entregar a cualquier persona. La otra clave es *privada* y el propietario debe guardarla para que nadie tenga acceso a ella. El remitente usa la clave pública del destinatario para cifrar el mensaje, y una vez cifrado, sólo la clave privada del destinatario podrá descifrar este mensaje.

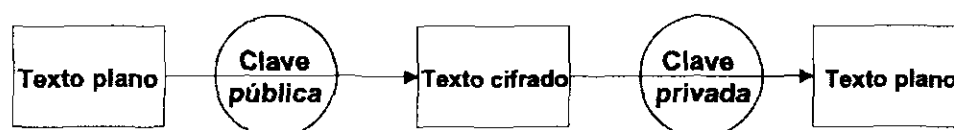


Figura 2.2 diagrama de un algoritmo asimétrico.

RSA

Si un usuario A desea enviar información cifrada, en primer lugar tiene que calcular un par de claves (*pública* y *privada*), para lo que ha de elegir aleatoriamente dos números primos grandes (del orden de cien dígitos), p y q , números que se han de mantener en secreto; si llamamos N (N se conoce como *modulo*) al producto $p \cdot q$, el usuario ha de determinar otro entero, d , llamado *exponente privado*, que cumpla

$$\text{mcd}(d; (p - 1) \cdot (q - 1)) = 1; d < N$$

es decir, d y el producto $(p - 1) \times (q - 1)$, que llamaremos función de Euler y denotaremos $\phi(N)$, han de ser primos. Con estos datos, ya tenemos la clave privada del cifrado: el par $(N; d)$; para obtener la clave pública, hallamos el inverso multiplicativo del número d respecto de $\phi(N)$, de la forma $e \times d = 1 \bmod \phi(N)$. Calculado este entero e , llamado *exponente publico*, la clave pública será el par $(N; e)$.

Una vez el emisor A dispone de sus claves pública y privada, podría enviar un mensaje cifrado, que llamaremos m , a un posible receptor, mediante la operación

$$c = m^e \bmod N$$

aplicada a cada elemento del mensaje.

Cuando el receptor del criptograma desee descifrar el mensaje recibido, ha de realizar la operación.

$$m = c^d \bmod N$$

para obtener el texto en claro del mensaje que acaba de recibir. El sistema RSA ha permanecido invulnerable hasta hoy, a pesar de los numerosos ataques de criptoanalistas; teóricamente es posible despejar d para obtener la clave privada, a partir de la función de descifrado, resultando

$$d = \log_c m(\bmod(p - 1))$$

Sin embargo, el calculo de logaritmos discretos es un problema de una complejidad desbordante, por lo que este tipo de ataque se vuelve impracticable: la resolución de congruencias del tipo $a^x = b(n)$, necesarias para descifrar el mensaje, es algorítmicamente inviable sin ninguna información adicional, debido al elevado tiempo de ejecución del algoritmo.

Los problemas mas representativos de los algoritmos de clave pública:

Velocidad: el RSA es 1000 veces más lento que los algoritmos simétricos. El problema se evita utilizando RSA para intercambiar una “clave de sesión” simétrica y luego usar un método simétrico para el resto de la sesión.

2.6.3 PGP

Con el paso de los años, PGP *pretty good privacy* (privacidad bastante buena) se ha convertido en uno de los mecanismos más populares y fiables para mantener la seguridad y privacidad en las comunicaciones, especialmente a través del correo electrónico, tanto para pequeños usuarios como para grandes empresas. Hasta la fecha la política de distribución de PGP ha consistido en permitir su uso gratuito para usos no comerciales y en publicar el código fuente en su integridad, con el objetivo de satisfacer a los desconfiados y a los curiosos.

Actualmente PGP se ha convertido en un estándar internacional [IET04], lo cual está dando lugar a la aparición de múltiples productos PGP, que permiten desde cifrar correo electrónico hasta a codificar particiones enteras del disco duro (PGPDisk), pasando por la codificación automática y transparente de todo el tráfico TCP/IP (PGPnet).

Codificación de Mensajes

Los algoritmos asimétricos emplean generalmente longitudes de clave mucho mayores que los simétricos. Por ejemplo, mientras que para algoritmos simétricos se considera segura una clave de 128 bits, para algoritmos asimétricos se recomiendan claves de al menos 1024 bits. Además, la complejidad de calculo que comportan estos últimos los hace considerablemente más lentos que los algoritmos de cifrado por bloques. En la práctica los métodos asimétricos se emplean únicamente para codificar la *clave de sesión* (simétrica) de cada mensaje[LUC99].

los algoritmos simétricos de cifrado son considerablemente mas rápidos que los asimétricos [GAR95].

Por esta razón PGP cifra primero el mensaje empleando un algoritmo simétrico (ver figura 2.2) con una clave generada aleatoriamente (*clave de sesión*) y posteriormente codifica la clave haciendo uso de la llave pública del destinatario. Dicha clave es extraída convenientemente del anillo de claves públicas a partir del identificador suministrado por el usuario, todo ello de forma transparente, por lo que únicamente debemos preocuparnos de indicar el mensaje a codificar y la lista de identificadores de los destinatarios.

Cuando se trata de decodificar el mensaje, PGP simplemente busca en la cabecera las claves públicas con las que está codificado y nos pide una contraseña. La contraseña serviría para que PGP abra nuestro anillo de claves privadas y compruebe si tenemos una clave que permita decodificar el mensaje. En caso afirmativo, PGP descifrara el mensaje. Nótese que siempre que queramos hacer uso de una clave privada, habremos de suministrar a PGP la contraseña correspondiente, por lo que si el anillo de claves privadas quedara comprometido, un atacante aún tendría que averiguar la contraseña para descifrar los mensajes. No obstante, si el anillo de claves privadas quedara comprometido, lo mejor sería *revocar* todas las claves que tuviera almacenadas y generar otras nuevas.

Como puede comprenderse, gran parte de la seguridad de PGP reside en la calidad del generador aleatorio que se emplea para generar las claves de sesión, puesto que si alguien logra predecir la secuencia de claves que se esta usando, podría descifrar todos los mensajes independientemente de los destinatarios a los que vayan dirigidos. Afortunadamente, PGP utiliza un método de generación de números pseudo-aleatorios muy seguro, y protege criptográficamente la semilla aleatoria que necesita. No obstante, se consideraría *sensible* al archivo que contiene dicha semilla, y por lo tanto se ha de evitar que quede expuesto.

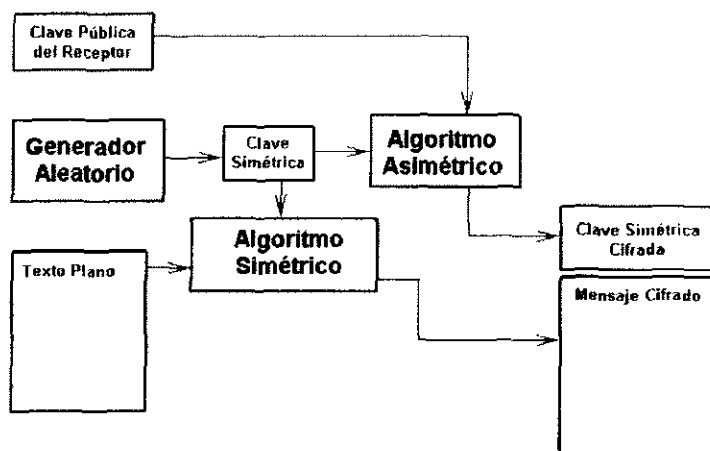


Figura 2.3 Codificación de un mensaje PGP [LUC99].

2.7 Archivos Log.

En función de la configuración del sistema de auditoría de cada equipo Unix los eventos que sucedan en la máquina se registraran en determinados archivos; aunque podemos loggear en cualquier archivo, existen ciertos archivos de registro 'habituales' en los que se almacena información. A continuación vamos a comentar los más comunes y la información que registra cada uno de ellos.

syslog

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

```
neo:/# cat /var/log/syslog
Feb 6 03:31:37 neo rpcbind: connect from 127.0.0.1 to getport(R )
Feb 6 11:07:04 neo in.telnetd[14847]: connect from aplex
Feb 6 11:40:43 neo in.telnetd[14964]: connect from localhost
neo:/#
```

messages

En este archivo de texto se almacenan datos 'informativos' de ciertos programas, mensajes de baja o media prioridad destinados más a informar que a avisar de sucesos importantes, como información relativa al arranque de la máquina; no obstante, como sucedía con el archivo syslog, en función de /etc/syslog.conf podremos guardar todo tipo de datos. Para visualizar su contenido es suficiente una orden como cat o similares:

```
neo:/# head -70 /var/adm/messages
Jan 24 18:09:54 neo unix: SunOS Release 5.7 Version Generic
[UNIX(R) System V Release 4.0]
Jan 24 18:09:54 neo unix: Copyright (c) 1983-1998, Sun Microsystems, Inc.
Jan 24 18:09:54 neo unix: mem = 65152K (0x3fa0000)
Jan 24 18:09:54 neo unix: avail mem = 51167232
Jan 24 18:09:54 neo unix: root nexus = i86pc
Jan 24 18:09:54 neo unix: isa0 at root
Jan 24 18:09:54 neo unix: pci0 at root: space 0 offset 0
Jan 24 18:09:54 neo unix: IDE device at targ 0, lun 0 lastlun 0x0
Jan 24 18:09:54 neo unix: model WDC WD84AA, stat 50, err 0
Jan 24 18:09:54 neo unix: cfg 0x427a, cyl 16383, hd 16, sec/trk 63
Jan 24 18:09:54 neo unix: mult1 0x8010, mult2 0x110, dwcap 0x0,
neo:/#
```

wtmp

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

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

Los registros guardados en este archivo (y también en el archivo `utmp`) tienen el formato de la estructura `utmp`, que contiene información como el nombre de usuario, la línea por la que accede, el lugar desde donde lo hace y la hora de acceso; se puede consultar la página de manual de funciones como `getutent()` para ver la estructura concreta en el clon de Unix en el que trabajemos. Algunas variantes de Unix (como Solaris o IRIX) utilizan un archivo `wtmp` extendido denominado `wtmpx`, con campos adicionales que proporcionan más información sobre cada conexión.

Lastlog

El archivo `lastlog` es un archivo binario guardado generalmente en `/var/adm/`, y que contiene un registro para cada usuario con la fecha y hora de su última conexión;

podemos visualizar estos datos para un usuario dado mediante `_orden` como `who` o `finger`:

```
neo:/# finger aglex
```

```
Login: aglex                      Name: Adalberto Gonzalez
```

```
Directory: /home/aglex           Shell: /bin/bash
```

```
Last login Thu Jan 27 10:29 (CST) on pts/1 from customer-148-235-151-214.uninet-ide.com.mx
```

```
No mail.
```

```
neo:/#
```

Faillog

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

```
andercheran login: toni
```

```
Password:
```

```
Linux 2.0.33.
```

```
1 failure since last login. Last was 14:39:41 on tty9.
```

```
Last login: Wed May 13 14:37:46 on tty9 from pleione.cc.upv.es.
```

Sulog

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

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

El registro de las invocaciones a la orden su puede estar deshabilitado por defecto en algunos sistemas Unix: En algunas distribuciones de Linux es necesario modificar el archivo `/etc/login.defs` para habilitar el registro de la actividad.

2.8 Herramientas de seguridad.

A continuación se presentan algunas herramientas de seguridad existente, algunas de dominio publico mundial y otras no es posible adquirirlas fuera de los Estados Unidos.

CFS(Cryptographic File System)

CFS es un sistema de archivos de cifrado que opera a nivel de usuario. Actúa como interfaz de cualquier sistema de archivos estándar de Unix, incluyendo NFS(los datos no viajan o se mantienen en el disco en texto claro en ningún momento).

COPS (Computer Oracle and Password System)

COPS es una colección de herramientas de seguridad para Unix que sirve para automatizar tareas que normalmente el administrador realiza de forma manual (passwords, restricciones, sistemas de archivos, etc). Las posibles debilidades del

sistema son guardadas en un archivo resultado o enviadas por correo, pero NUNCA corregidas.

DTK

Detection toolKit es una herramienta que escucha peticiones al sistema de Unix y responde con información falsa, haciendo que el potencial intruso piense que nuestro sistema está lleno de bugs. De esta forma, y gracias al monitoreo y sistema de logs de DTK, se puede obtener mucha información sobre los atacantes, así como ganar tiempo si interesa seguir el ataque.

Isotf (list open files)

Isotf es un programa que examina descriptores de archivo abiertos por proceso; de esta forma, se puede localizar sockets a la escucha, procesos que describen en un archivo, archivos abiertos por un proceso (importante a la hora de detectar sniffers), etc.

Npasswd

Npasswd es una herramienta que sustituye al comando passwd de cualquier sistema operativo Unix habitual. Es una recomendable herramienta para cualquier sistema, ya que obliga a los usuarios a escoger passwords aceptables, rechazando palabras de diccionario, joes (mismo login y password), claves demasiado cortas, etc. ... Funciona con NIS o seguridad C2 (Shadow Password).

Portmapper

Portmapper es una herramienta que permite monitorear y filtrar peticiones a diferentes servicios atendidos con portmapper (NFS o Yellow Pages). Su uso es mucho menos común ya que ha quedado desplazado por los maleadores de puertos estándar que

vienen con muchos sistemas Unix, y que incorporan en la actualidad mas medidas de seguridad que este paquete.

Sniff & TOD

Sniff es una herramienta de monitoreo y sniffer que trabaja sobre diferentes plataformas Unix. Proporciona información detallada sobre todo el trafico que circula por su sistema, y también le ofrece la posibilidad de neutralizarlo si se utiliza TOD (touch of dead), es decir, cerrar las conexiones uqe pasan por su maquina.

Strobe

Este programa es un supervisor de puertos que funciona sobre multitud de sistemas Unix. Puede resultar útil para comprobar que servicios de red se tienen aceptando conexiones en el sistema, y así poder reducir su numero al mínimo necesario.

SWatch

Este programa es útil para monitorear los archivos de log de un sistema y emprender acciones (como notificar al administrador) cuando se detectan situaciones sospechosas.

TCFS(transparent Cryptographic File System)

TCFS es un sistema de archivos que incorpora cifrado de datos. Siguiendo un mecanismo parecido a NFS, ofrece la posibilidad de mantener archivos encriptados de unidad, y también incrementa la seguridad en la comunicación por red entre cliente y servidor de discos: todos los datos viajan encriptados, lo que hace a TCFS bastante recomendable para sistemas distribuidos. TCFS opera anivel de kernel, por lo que a priori es mas rápido y seguro que CFS, de Matt Blaze.

TCPdump

Es una herramienta para analizar el trafico de la red. Imprime las cabeceras de paquetes que circulan por una interfaz de red y cumplen ciertas características(protocolo, puerto,direccion).

Titan

Esta formado por una serie de shellscrips que solucionan pequeños problemas de seguridad genéricos, como la existencia por defecto accesos con un shell valido.

Tripwire

Tripwire es un verificador de integridad para rchivos y directorios, muy util para evitar la onsercion de troyanos entre los ejecuales de nuestro sistema. Utiliza para ello un algoritmo de firma digital.

Capítulo 3. Diseño de la herramienta

Se presenta una propuesta a la problemática de la administración de un servidor Linux.

3.1 Introducción

HAASU es una propuesta para resolver la problemática y el exceso de trabajo que presenta la administración manual de servidores Linux. Se pretende que las actividades propias de administrar un servidor se realicen de forma automática, de manera que la ejecución de instrucciones y el monitoreo de actividades sean cada vez mas rápidas y eficientes, reduciendo de esta manera el uso de capital humano y el tiempo que normalmente se invierte para obtener información útil para la toma de decisiones. Es por lo anterior que HAASU propone:

- Administrar a los usuarios , a los grupos que pertenecen y sus actividades dentro del servidor.
- Ofrecer información confiable para que el administrador realice una interpretación adecuada para la toma de decisiones de la situación del servidor.
- Capacidad de ejecutar respuestas automatizadas a eventos que perturben la seguridad del servidor
- El uso de aplicaciones conjuntas desde una sola herramienta.
- Verificar la funcionalidad del hardware del servidor.

Todas esas funciones tienen como fin el hacer de la administración de un servidor una tarea sencilla, eliminando obstáculos, y perdida de tiempo identificando actividades en el servidor de forma manual, concentrando el análisis de las actividades del servidor en un solo punto, en este caso HAASU.

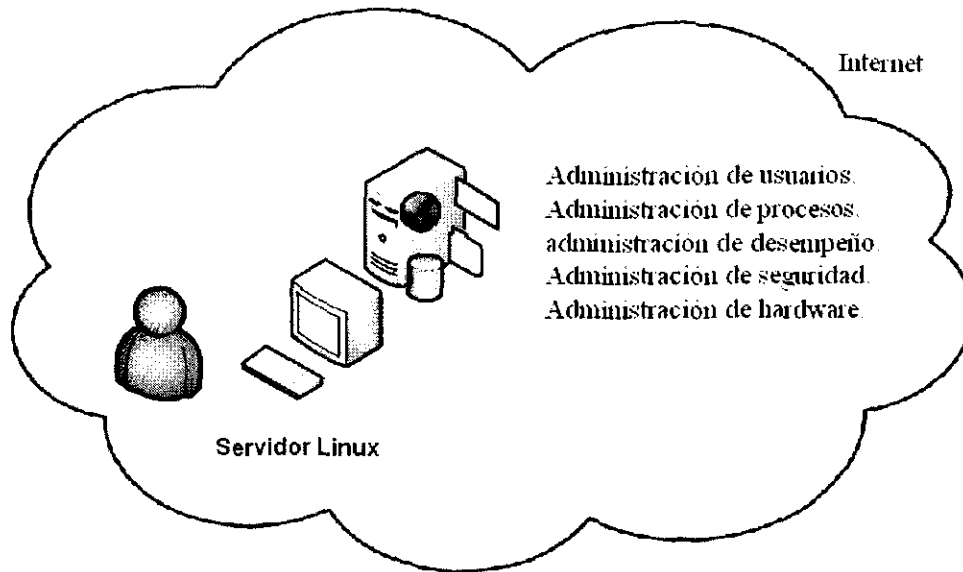


Figura 3.1 Actividades de HAASU en el servidor.

Una característica importante para el desarrollo de HAASU es el uso de PSP (Personal Software Process) con el fin de contar con el respaldo de una metodología para el desarrollo de software.

3.2 PSP

A continuación se explica brevemente qué es PSP, su propósito y el por qué se ha adoptado PSP para el desarrollo de HAASU.

3.2.1 PSP (*personal software process*)

PSP (Personal Software Process) muestra a los ingenieros como[HUM02]:

- Administrar la calidad de sus proyectos.
- Hacer compromisos que puedan lograr.
- Mejorar sus estimaciones y la planeación.

- Reducir defectos en sus productos.

Como los gastos de personal constituyen el 70 por ciento del costo del desarrollo de software, las habilidades y los hábitos de trabajo de los ingenieros, determinan en gran parte los resultados del proceso de desarrollo de software. Basado en prácticas encontradas en el Capability Maturity Model (CMM), el PSP puede ser usado por ingenieros como una guía a un acercamiento disciplinado y estructurado al software en vías de desarrollo.

PSP puede ser aplicado a muchas partes del procesos del desarrollo de software, incluyendo:

- Desarrollo de pequeños programas.
- Definición de requerimientos.
- Redacción de documentos.
- Pruebas de sistema.
- Mantenimiento de sistemas.
- Realce de grandes sistemas de software.

3.2.2 Niveles de PSP

En PSP todas las tareas y actividades que el ingeniero de software debe realizar durante el proceso de desarrollo de un producto de software, están puntualmente definidas en un conjunto de documentos conocidos como scripts. Los scripts son el punto medular de PSP, por lo que se hace mucho énfasis en que deben ser seguidos en forma disciplinada, ya que de ello dependerá el éxito de la mejora que se busca. Gran parte de las tareas y actividades definidas en los scripts generará en su realización un conjunto de datos, fundamentalmente de carácter estadístico. La

aplicación de PSP en varios procesos de desarrollo y el análisis de la información estadística generada en cada uno de éstos permitirán al ingeniero de software identificar, tanto sus fortalezas como sus debilidades y crecer a través de un proceso de autoaprendizaje y auto mejora[VEL03].

Los scripts se organizan en cuatro niveles, identificados del 0 al 3, atendiéndose en cada nivel un conjunto de aspectos a mejorar del proceso de desarrollo de software.

- PSP 0. MEDICIÓN PERSONAL
- PSP 1. PLANEACIÓN PERSONAL
- PSP 2. CALIDAD PERSONAL
- PSP 3. CÍCLICO PERSONAL

Al primer nivel (nivel 0) o de medición personal, al segundo(nivel 1) o de planeación personal, al tercero (nivel 2) o de calidad personal, y al cuarto (nivel 3) o cíclico personal. Cada uno de estos niveles, con excepción del 3, tiene una versión que los extiende, introduciendo tareas y actividades para un mejor manejo de los aspectos de interés en nivel. Cada uno de los niveles extiende los aspectos considerados en el nivel inmediato anterior. Una de las razones de esta clasificación puede ser el que PSP es una metodología de mejora basada en datos estadísticos, los cuales deben ser cuidadosamente recabados por el ingeniero de software; el aumento gradual de la cantidad de datos que debe recolectar el ingeniero introduce, por consiguiente, el cambio en su manera de trabajo de una manera paulatina[VEL03].

Es por lo anterior que se utilizara PSP 1.1 para el desarrollo de HAASU, teniendo como justificación las ventajas que presenta el usar una metodología de este tipo. Considerando las limitantes del conocimiento a fondo en los niveles de PSP, las cuales solo permiten que se desarrolle bajo el nivel 1.1 de PSP.

3.3 Descripción General de HAASU

3.3.1 *Perspectiva*

HAASU es una Herramienta para el apoyo en la administración de un servidor Unix. Donde HAASU tendrá como objetivos:

- Proporcionar una herramienta confiable que ofrezca un sistema de respuesta inmediata ante situaciones de riesgo para la integridad del servidor.
- Que la herramienta proporcione información concreta que sirva de base para la toma de decisiones y que esta misma información permita conocer vulnerabilidades y prever mejoras en la administración del servidor.
- Que la herramienta tenga una interfaz amigable para su uso.

3.3.1.1 Interfaz De Sistema

Existen dos interfaces principales:

- Una interfaz responsable de recibir las peticiones hechas por el administrador, para que sean realizadas por la aplicación. Está también encargada de presentarle los resultados obtenidos.
- La segunda interfaz, ya dentro del sistema, se encargará de transformar todas las peticiones en instrucciones que sean entendibles para la herramienta cargada en el servidor.

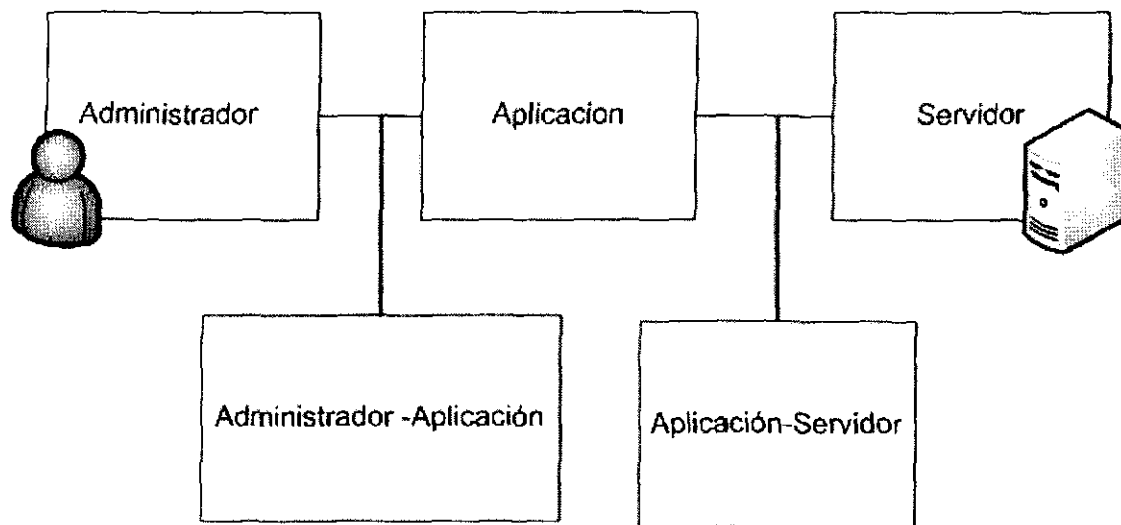


Figura 3.2 Interfaz del sistema.

3.3.1.2 Interfaces De Hardware

HAASU obtiene información de los siguientes medios:

- Ingreso de instrucciones del usuario a través del teclado, con la intención de que el sistema reciba estas peticiones y sean ejecutadas por la herramienta.
- Información almacenada en el servidor (medio lógico de almacenamiento). La herramienta hace uso de esta información para su ejecución.

Las salidas de información del sistema de la herramienta son las siguientes:

- Salida mediante monitor. Generalmente al usuario se le presentará la información solicitada a través del monitor. Ésta será la salida predeterminada de información.

3.3.1.3 Interfaces De Comunicación

HAASU podrá ejecutarse en un modelo cliente servidor a través de Internet, haciendo uso de SSH(Secure Shell) como herramienta de conexión.

De esta manera el usuario (administrador) contara con la libertad de administrar el servidor sin importar que no este en forma física con el servidor.

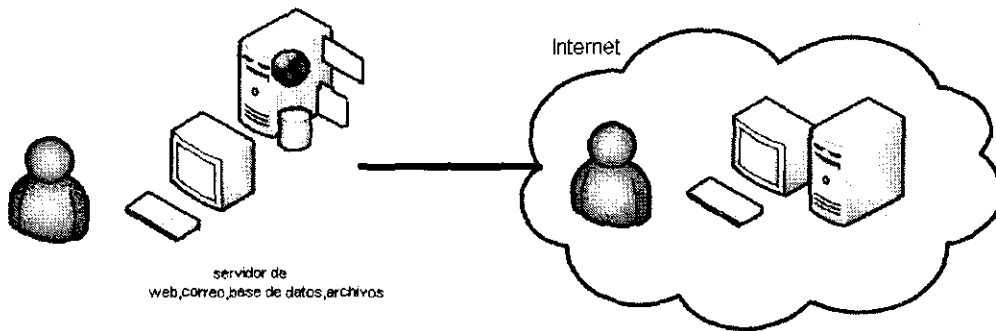


Figura 3.3 Estructura general de la aplicación.

3.4 Funcionalidades

Dentro de la administración de un servidor unix, existen diferentes actividades las cuales dividimos en 4 perfiles que serán cubiertos por HAASU una vez implementada en el servidor:

- Administración de usuarios.
- Administración de procesos.
- administración de desempeño.

- Administración de seguridad.
- Administración de hardware.

La gestión de estos módulos será responsabilidad de un programa principal que hará uso de estos módulos y los ejecutara de manera simultanea.

3.4.1 Operaciones

HAASU podrá realizar las siguientes operaciones en sus 5 diferentes módulos:

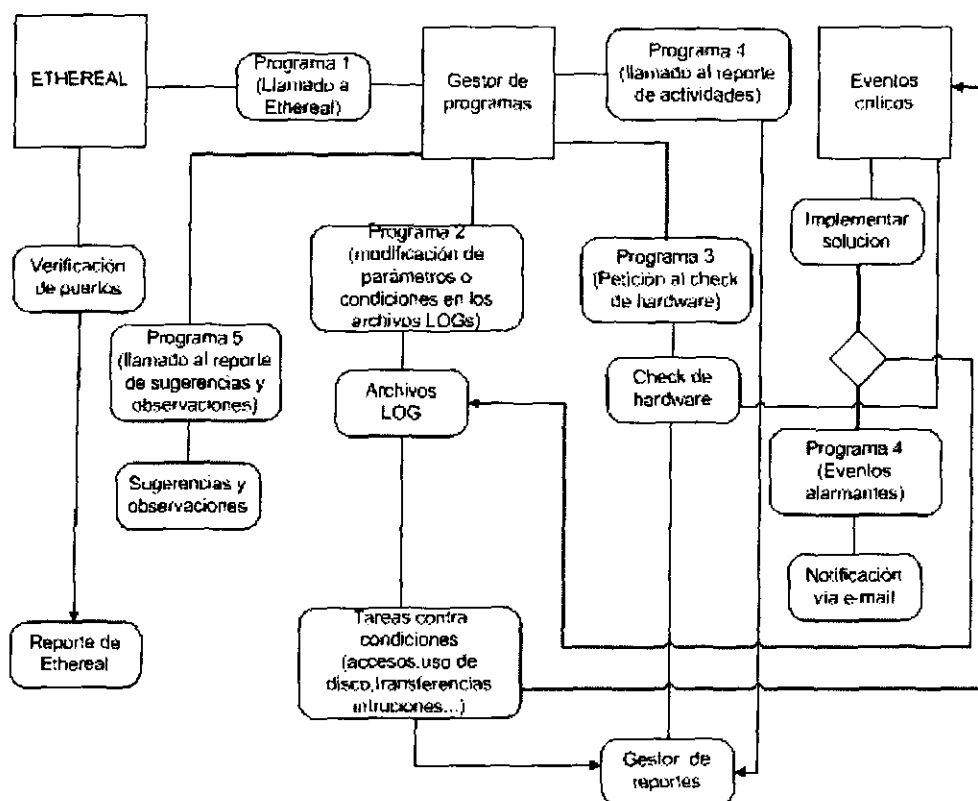


Figura 3.4 Diagrama general de los módulos de HAASU.

- Gestor de Módulos.
 - Mostrará el encabezado y será el encargado de presentar la gestión de las actividades. Incluyendo un menú con los 5 módulos principales.
 - Tendrá acceso a cinco opciones:
 - Puertos.
 - LOGs.
 - Hardware.
 - Reportes.
 - Observaciones.
- Puertos.
 - Invocará a Ethereal cuya función es la de analizar el trafico en el servidor.
 - Ofrecerá un reporte con los puertos más usados.
 - Identificara los puertos que estén abiertos.
- LOGs.
 - Permitirá visualizar las siguientes condiciones de operación del servidor:
 - Arranque.
 - Cron.

- Messages.
 - Secure.
 - Maillog.
 - Spooler.
-
- Permitirá hacer modificaciones de rango de eventos críticos (el porcentaje de uso de disco, tiempo de conexión, etc.) las cuales determinan cuando una actividad debe ser sujeta al a un análisis para determinar si dicha actividad no está permitida, o es riesgosa.
-
- Hardware.
 - Permitirá hacer un chequeo general del estado del hardware, con base en la respuesta será posible determinar el funcionamiento del hardware en el servidor.
-
- Reportes.
 - Desplegara la información recabada del análisis de la herramienta en el servidor.
 - Dicha información proveerá los elementos necesarios para el apoyo en la toma de decisiones.
-
- Observaciones.
 - Permitirá que el administrador asiente sus observaciones según la información que ha podido visualizar en los reportes de actividades.

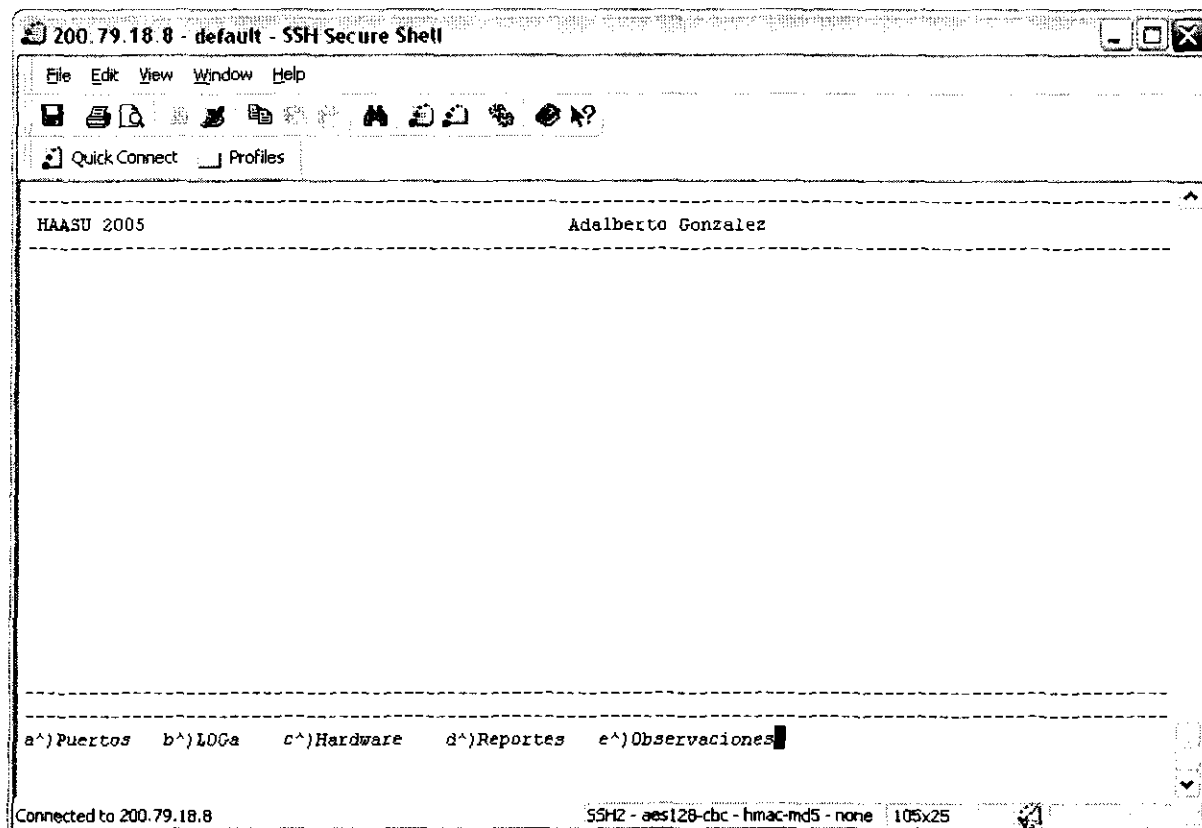


Figura 3.4 pantalla principal de HAASU.

3.5 Los Módulos de HAASU.

El usuario tendrá acceso a los módulos de HAASU a través de la interfaz creada en C. El uso de C es necesario para no afectar el rendimiento del servidor con procesos extras o uso de memoria excesivo. La idea es que la herramienta pase desapercibida por el servidor y de esta manera no afecte el las actividades del mismo.

3.5.1 Puertos.

Se utilizará Ethereal. Éste es un potente analizador libre de protocolos de redes, para computadoras con el sistema operativo Unix (o Windows). Permite capturar los datos

directamente de una red u obtener la información a partir de una captura en disco (puede leer más de 20 tipos de formato distintos). Destaca también por su impresionante soporte de más de 300 protocolos, gracias a la Licencia Pública General y sus colaboradores de todo el mundo. Pero sin duda alguna, la característica más útil para el estudio de ingeniería ha sido la posibilidad de ver la reconstrucción de un flujo de bits de una sesión TCP.

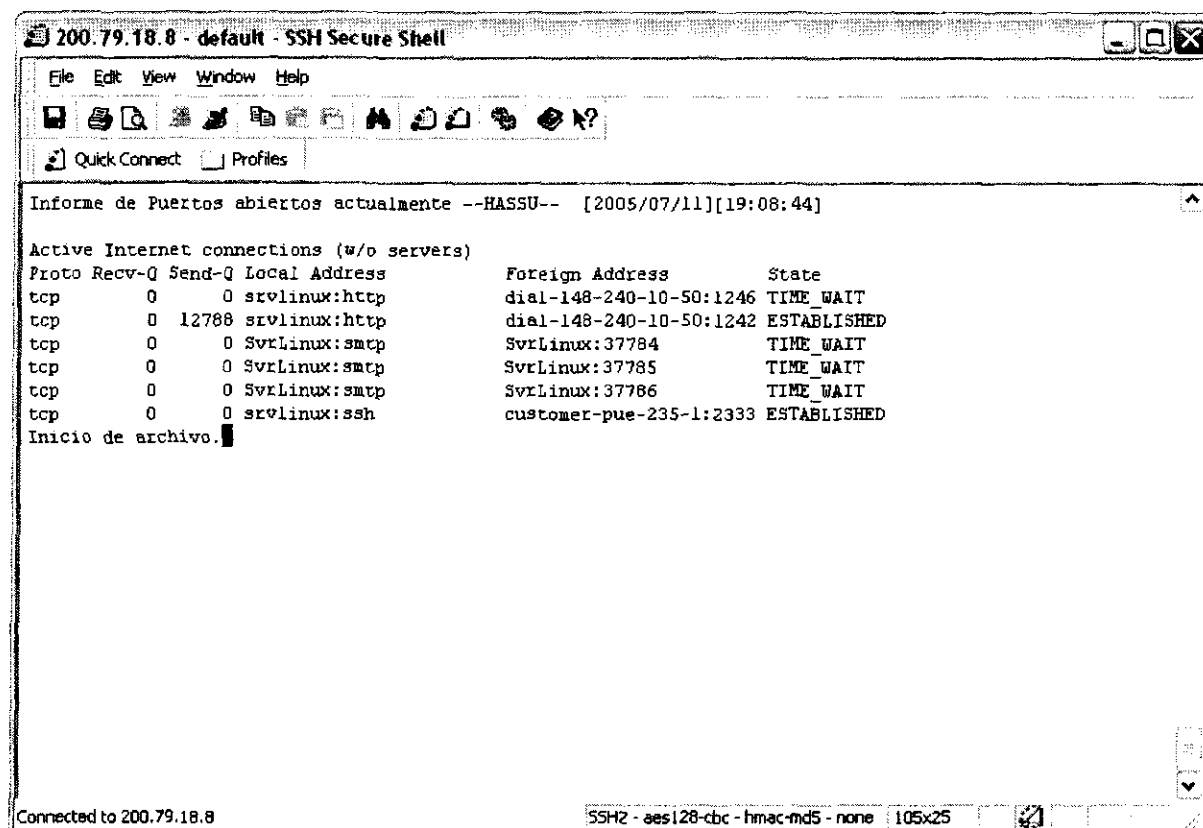


Figura 3.5 Informe de puertos abiertos vía ethereal

3.5.2 LOGs.

Visualiza los archivos almacenados en var /logs.

HAASU genera una copia temporal de cada LOG que se solicite en el momento que se desee visualizar algún archivo, de esta manera el archivo original esta disponible para recibir la información que le sea reportada.

Se hace uso de 5 LOG's principales:

- Boot.log
- Cron
- Messages
- Secure
- Maillog
- Spooler

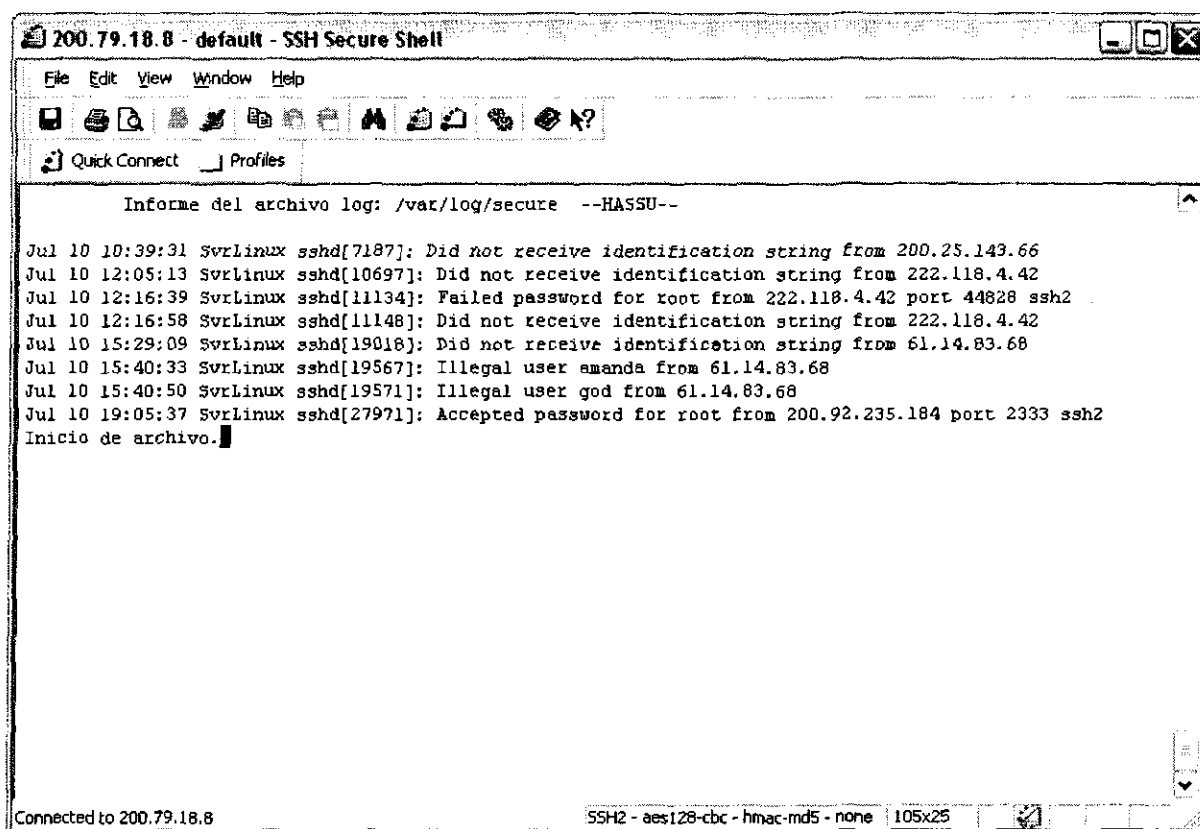
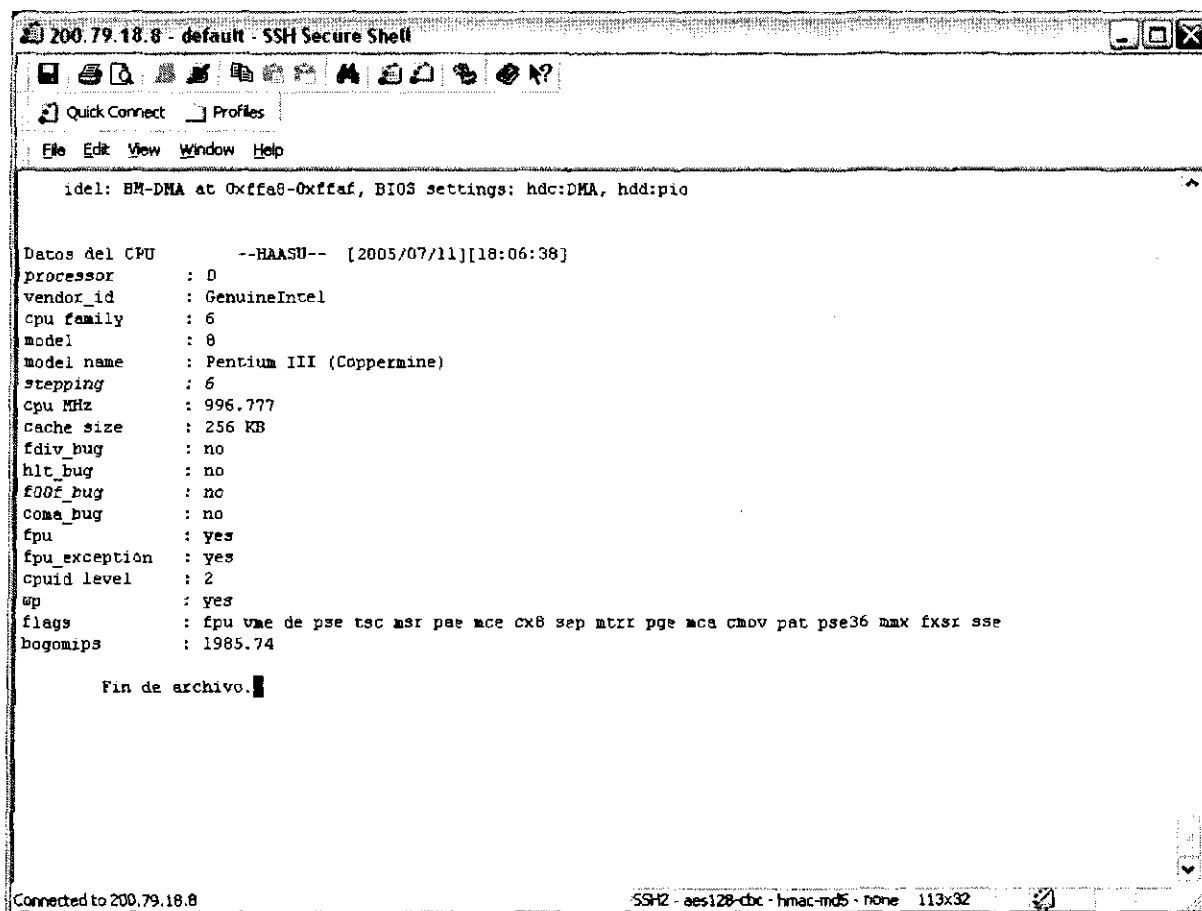


Figura 3.6 Informe del archivo LOG: secure

3.5.3 Hardware.

HAASU analiza la configuración de memoria, discos duro IDE, dispositivos SCSI y CPU.

Toma la configuración de arranque como la inicial y las analiza cada determinado tiempo por posibles variaciones en los subsecuentes chequeos, si existe un cambio en la configuración envía una notificación por correo al administrador.



```

200.79.18.8 - default - SSH Secure Shell

Quick Connect Profiles
File Edit View Window Help

idel: BM-DMA at 0xffa8-0xffaf, BIOS settings: hdc:DMA, hdd:pio

Datos del CPU      --HAASU-- [2005/07/11][18:06:38]
processor          : 0
vendor_id         : GenuineIntel
cpu_family        : 6
model             : 8
model name        : Pentium III (Coppermine)
stepping          : 6
cpu MHz           : 996.777
cache size        : 256 KB
fdiv_bug          : no
hlt_bug           : no
f00f_bug          : no
coma_bug          : no
cpu               : yes
fpu_exception     : yes
cpuid level       : 2
wp                : yes
flags              : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov pat pse36 mmx fxsr sse
bogomips          : 1985.74

Fin de archivo.

Connected to 200.79.18.8
SSH2 - aes128-cbc - hmac-md5 - none 113x32

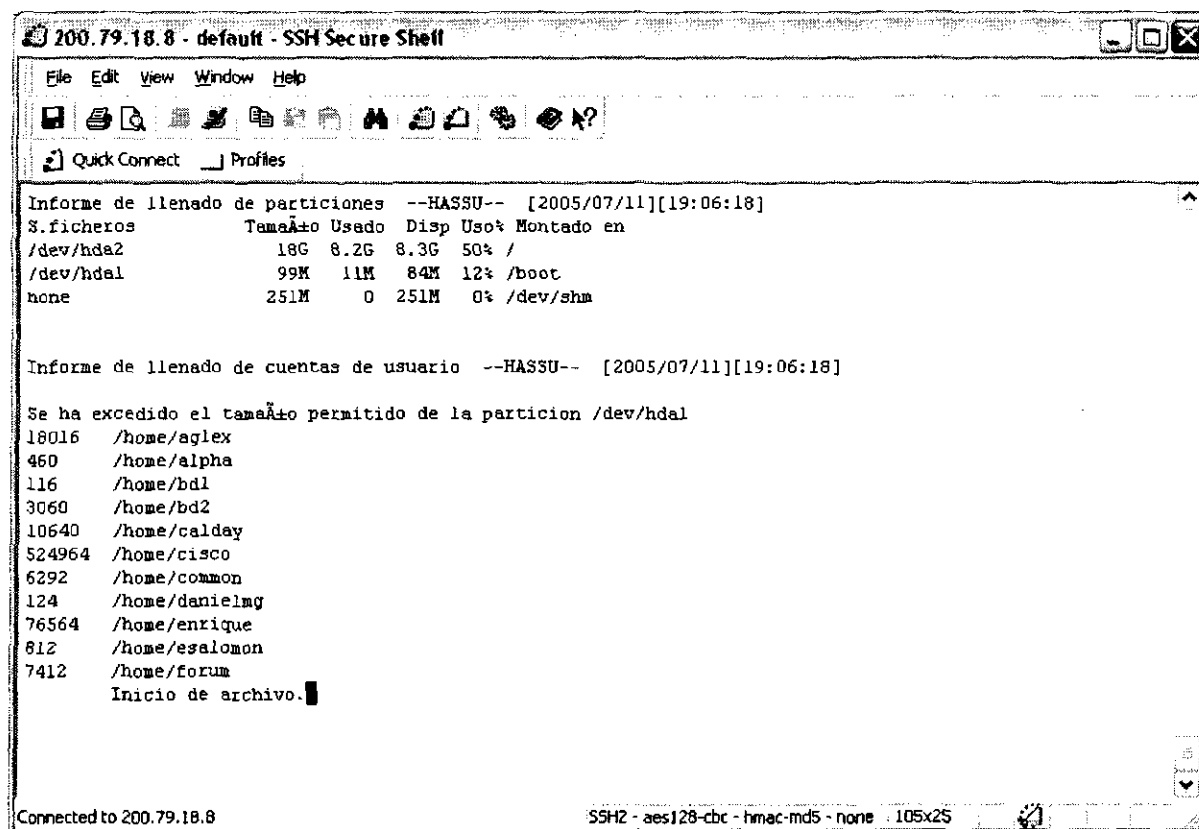
```

Figura 3.7 Informe de hardware

3.5.4 Reportes.

Visualiza todos los reportes de puertos y trafico de red, archivos LOG mas el de espacio en disco duro. Verifica que las particiones no sobrepasen cierto limite, notificando al administrador vía correo en caso de que esto suceda.

Además ofrece un reporte general en el cual se archivan todas las irregularidades que ocurran en el servidor ofreciendo así información concreta sobre el estado del servidor.



The screenshot shows an SSH terminal window titled "200.79.18.8 - default - SSH Secure Shell". The window contains two reports generated by the HASSU tool on 2005/07/11 at 19:06:18.

The first report is "Informe de llenado de particiones" (Disk Space Usage Report). It shows a table with columns: S.ficheros, Tamaño, Usado, Disp, Uso%, and Montado en.

S.ficheros	Tamaño	Usado	Disp	Uso%	Montado en
/dev/hda2	18G	8.2G	8.3G	50%	/
/dev/hda1	99M	11M	84M	12%	/boot
none	251M	0	251M	0%	/dev/shm

The second report is "Informe de llenado de cuentas de usuario" (User Account Usage Report). It states: "Se ha excedido el tamaño permitido de la particion /dev/hda1" (The size of the /dev/hda1 partition has exceeded the allowed size). It then lists a series of user accounts and their home directories, including /home/aglex, /home/alpha, /home/bdl, /home/bd2, /home/calday, /home/cisco, /home/common, /home/danielmg, /home/enrique, /home/esalomon, and /home/forum. The report ends with "Inicio de archivo."

The terminal window also shows a status bar at the bottom indicating "Connected to 200.79.18.8" and "SSH2 - aes128-cbc - hmac-md5 - none 105x25".

Figura 3.8 Informe de llenado de particiones

3.5.5 Observaciones.

Permite agregar observaciones extensas guardando fecha y hora en la que se ingresa el comentario, además permite agregar clave de búsqueda al comentario, de ésta manera es posible realizar búsquedas de los comentarios previamente guardados con sus claves logrando así búsquedas mas rápidas.

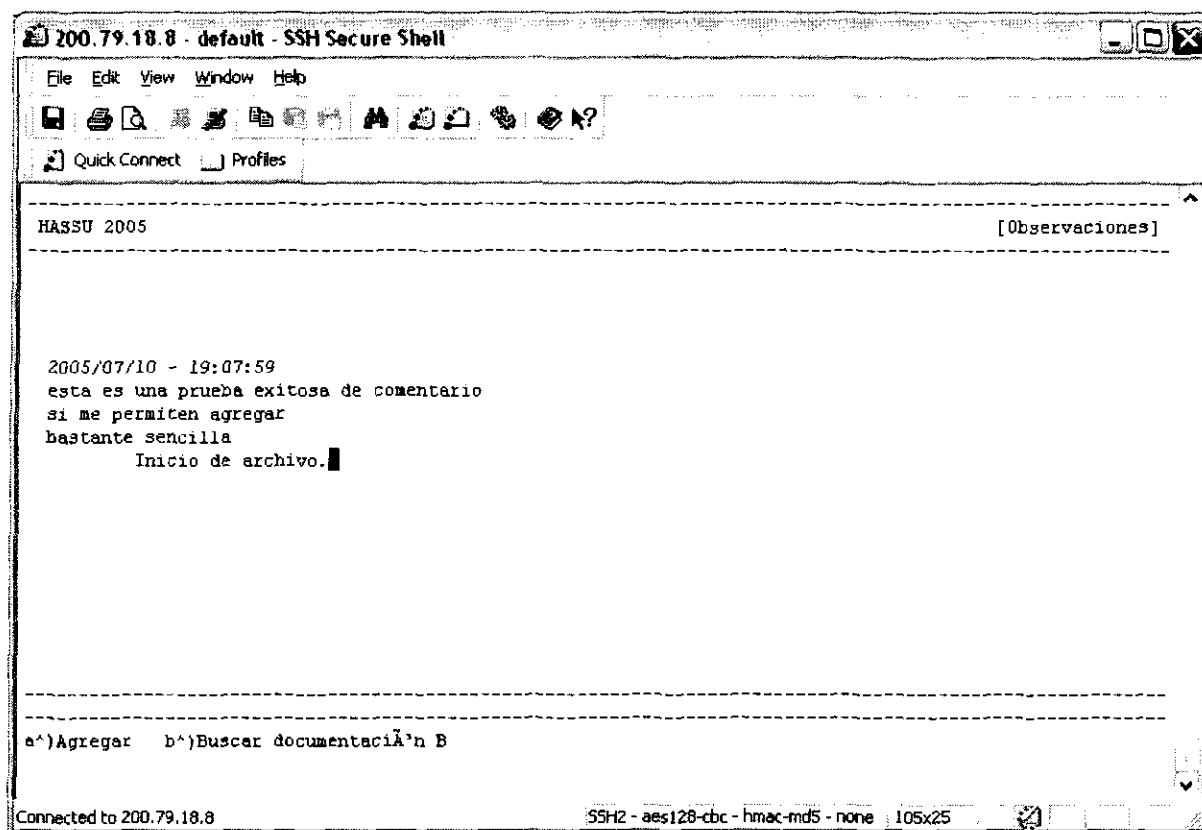


Figura 3.9 Comentario agregado más la clave para su búsqueda.

Capítulo 4. Pruebas y Resultados

En este capítulo se presentan las pruebas de integración así como la productividad registrada por el desarrollador de HASSU.

Las pruebas realizadas a HAASU se realizaron en un servidor Linux (svrlinux) que presta servicios a la población estudiantil de I.I.T.I. así como al personal docente en la Universidad Popular Autónoma del Estado de Puebla.

La siguiente tabla muestra los resultados obtenidos en las pruebas de integración realizadas a HAASU. Las pruebas de integración permiten verificar que todos los componentes de HAASU interactúen de manera optima en el servidor.

PRUEBAS DE INTEGRACIÓN			
Orden	Nombre Prueba	Tipología prueba	Salida Esperada
1	Gestor de Programas		
1.1	Interfaz	Caja Negra	muestra el nombre de la herramienta
1.2	Interfaz	Caja Negra	muestra la opción puertos
1.3	Interfaz	Caja Negra	muestra la opción logs
1.4	Interfaz	Caja Negra	muestra la opción hardware
1.5	Interfaz	Caja Negra	muestra la opción reportes
1.6	Interfaz	Caja Negra	muestra la opción observaciones
2	Puertos		
2.1	Ethereal	Caja Negra	muestra el informe de puertos abiertos
3	LOG's		
3.1	LOG's	Caja Negra	muestra el menu de opciones LOG's
3.1.1	Boot.log	Caja Negra	muestra el informe de Boot.log
3.1.2	Cron	Caja Negra	muestra el informe de Cron
3.1.3	Messages	Caja Negra	muestra el informe de Messages
3.1.4	Secure	Caja Negra	muestra el informe de Secure
3.1.5	Maillog	Caja Negra	muestra el informe de Maillog
3.1.6	Spooler	Caja Negra	muestra el informe de Spooler
4	Hardware		
4.1	Hardware	Caja Negra	muestra el informe de hardware
5	Reportes		
5.1	Reportes	Caja Negra	muestra el menú de opciones Reportes
5.2	HD y particiones	Caja Negra	muestra el reporte de HD y particiones
5.3	Hardware y cpu	Caja Negra	muestra el reporte de hardware y cpu
5.4	Puertos y trafico	Caja Negra	muestra el reporte de puertos y trafico
6	Observaciones		
6.1	Agregar	Caja Negra	permite agregar una observación y una clave
6.2	Buscar por clave	Caja Negra	despliega el resultado de una búsqueda por clave
6.3	Buscar por fecha	Caja Negra	despliega el resultado de una búsqueda por fecha
7	Consulta remota		
7.1	HASSU vía SSH	Caja Negra	Ejecución de HASSU usando SSH como terminal

Figura 4.1 Tabla de pruebas de integración..

PRUEBAS DE INTEGRACIÓN			
Clave	Entrada Real	Salida Real	Resultado
1			
1.1	# haasu	HASSU 2005	Satisfactorio
1.2	# haasu	a)Puertos	Satisfactorio
1.3	# haasu	b)LOG's	Satisfactorio
1.4	# haasu	c)Hardware	Satisfactorio
1.5	# haasu	d)Reportes	Satisfactorio
1.6	# haasu	e)Observaciones	Satisfactorio
2			
2.1	'a'	salida esperada	Satisfactorio
3			
3.1	'b'	salida esperada	Satisfactorio
3.1.1	'a' (en LOG's)	salida esperada	Satisfactorio
3.1.2	'b' (en LOG's)	salida esperada	Satisfactorio
3.1.3	'c' (en LOG's)	salida esperada	Satisfactorio
3.1.4	'd' (en LOG's)	salida esperada	Satisfactorio
3.1.5	'e' (en LOG's)	salida esperada	Satisfactorio
3.1.6	'f' (en LOG's)	salida esperada	Satisfactorio
4			
4.1	'c'	salida esperada	Satisfactorio
5			
5.1	'd'	salida esperada	Satisfactorio
5.2	'a' (en reportes)	salida esperada	Satisfactorio
5.3	'b' (en reportes)	salida esperada	Satisfactorio
5.4	'c' (en reportes)	salida esperada	Satisfactorio
6			
6.1	'a' (en observaciones)	salida esperada	Satisfactorio
6.2	'b' (en observaciones)	salida esperada	Satisfactorio
6.3	'c' (en observaciones)	salida esperada	Satisfactorio
7			
7.1	Conexión al ip 200.79.18.8 y ejecución de HAASU	Ejecución de HAASU	Satisfactorio

Figura 4.1 Tabla de pruebas de integración (continuación).

Para el caso de HAASU, además de estar instalada y operando en el servidor Linux de I.I.T.I se probó desde 3 diferentes PC's vía SSH obteniendo resultados satisfactorios en la ejecución de las pruebas.

El uso de PSP para el desarrollo de HASSU permite obtener resultados de la productividad y los defectos inyectados por el desarrollador durante la elaboración de este proyecto.

La cifra registrada en la productividad del desarrollador (figura 4.2) dada en LOC/Hr (líneas de código /Hora) podría considerarse de buen nivel, debido a que para el conteo de LOC se evita el registrar líneas vacías, declaraciones, llaves y todo aquello que no represente código productivo.

La densidad de defectos dada en Defects/KLOC (defectos / mil líneas de código) es considerada baja, y se le atribuye al tiempo invertido al diseño de HAASU, el cual devuelve beneficios en tiempo, pues la inversión de tiempo es mínima cuando se trata de corregir errores.

Desarrollador	Numero de LOC Productivas	Productividad LOC/Hr	Densidad de defectos Defects/KLOC
Adalberto González Ayala	1417	27.9	84.2

Figura 4.2 Registro de productividad del desarrollador.

Capítulo 5. Conclusiones

En este capítulo se presentan las conclusiones de este proyecto.

Al concluir este trabajo se reafirma la necesidad de contar con una herramienta capaz de proveer información fundamental e inmediata para la toma de decisiones en la administración de servidores.

El uso de HAASU reduciría el uso de licencias de software en caso de que se optara por las herramientas comerciales, minimizaría el uso de personal que interviene en la administración y mantenimiento del servidor y se vislumbran ahorros en recursos de software, hardware, personal y tiempo.

HAASU se volverá un aliado de confianza en la administración de servidores debido a que presenta la posibilidad de generar reportes de eventualidades y enviarlos al administrador vía e-mail cuando estos pudieran ocurrir en el servidor.

En esencia HAASU puede ayudar a mejorar el desempeño de la administración de un servidor de T.I. respondiendo con los recursos adecuados en el momento y lugar adecuado.

Capítulo 6. Trabajo a futuro

En este capítulo se presentan algunas propuestas de trabajos a futuro sobre este proyecto.

La funcionalidad de HAASU permite visualizar cierto potencial en el campo de la administración de servidores, aquí se presentan algunas propuestas de trabajos a futuro.

Probar el supuesto de que HAASU genera ahorros en recursos de software , hardware, tiempo y personal que se involucre en la administración y mantenimiento de un servidor. Esto a partir de hacer pruebas exhaustivas en diferentes organizaciones las cuales dependan de un servidor para su operación.

Ocupar una granja de servidores los cuales contarían con HAASU para su operación, además de un servidor principal el cual pudiera servir de gestor entre todos estos servidores. Esta propuesta podría generar información importante en el mundo de las comunicaciones.

Exportar HASSU de Linux a otros sistemas operativos, lo cual permitiría abarcar un mayor mercado en el apoyo de servidores.

Bibliografía

Bibliografía

AMO94	Amoroso, E. , <i>Fundamentals of Computer Security Technology</i> , Upper Saddle River, NJ, USA, 1994,ISBN:0-13-108929-3
AND98	Anderson R. J., Needham R. M., and Shamir A.. <i>The Steganographic File System</i> ,Lecture Notes in computer science,Vol. 1525,pages73-82, 1998, ISBN:3-540-65386-4.
BAN01	Bandel David, Napier Robert, <i>Linux Edición Especial</i> , sexta edición, Madrid , España, 2001, ISBN: 8420533017.
BOR01	Borghello F. C., <i>Seguridad Informática: Sus Implicancias e Implementación</i> , Tesis maestría, Universidad de Buenos Aires, Buenos Aires, Argentina 2001, .
CER05	US-CERT, <i>Statistics on Federal Incident Reports</i> , [en línea], USA, 2005, Disponible: http://www.us-cert.gov/federal/statistics/
COH84	Fred Cohen, <i>Computer viruses: theory and experiments</i> , volume 6, issue1, pages:22-35, 1984, ISSN:0167-4048.
COR03	Corner , D. E y Stevens, D. <i>interworking with TCP/IP Vol. III</i> , Englewood Cliffs, NJ , USA 2003, ISBN:0-13-216987-8..
COM04	CSI Computer Security Institute, <i>CSI/FBI Computer Crime and Security Survey</i> ,[en linea], Sn Francisco, Cal., USA, 2004, Disponible: http://www.gocsi.com/
COM88	Apollo Computer. <i>Managing SysV System Software. Technical Report 010851-A00</i> , Apollo Computer, Inc. (Hewlett Packard), 1988.

CUR 90	Curry David A.. <i>Improving the security of your UNIX system. Technical report</i> , Information and Telecommunication Sciences and Technology Division, SRI International. 1990, ITSTD-721-FR-90-21.
DUF89	Tom Duff. <i>Experience with viruses on UNIX systems. Computing systems</i> , volume 2, pages: 155-171. USENIX Asociation, 1989.
FEL96	Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. <i>Web Spoofing: an Internet Con Game</i> . 20th National Information Systems Security Conference, Princeton University Department of Computer Science, Baltimore, Maryland, USA, 1997, Technical Report 540-96
FRI91	Frisch Aeleen, <i>Essential System Administration</i> . O'Reilly Associates Inc, edition 2, 1991, ISBN: 1-56592-127-5.
GAR95	Garfinkel Simson, <i>PGP Pretty Good Privacy</i> ,. O'Reilly & Associates, Inc, 1995, ISBN1-56592-098-8
HEB96	Heberlein L. Todd and Bishop Matt. <i>Attack class: Address spoofing</i> . In Proceddings of the 19th Nacional Infomation Systems Security Conference, pp. 371-377, 1996
HUM02	Humprey, Watts S. <i>A Discipline for Software Engineering</i> , Carnegie Mellon University, USA, ISBN:0-201-54610-8, 2002
IET04	The Internet Engineering Task Force, <i>IEFT Open PGP</i> , [en linea] [consulta: promavera 2005] Disponible: http://web.mit.edu/network/ietf/sa/
KLA95	Klaus Christopher. <i>Stealth Scanning – Bypassing Firewalls and satan Detectors</i> , Internet Security Systems, Inc, Miller Court

	West, Norcross, GA, 1995.
LUC99	Lucena López, Manuel José, <i>Criptografía y Seguridad en Computadores</i> , 2da edición, Departamento de Informática, Escuela Politécnica Superior Universidad de Jaen, 1999.
MCD99	McDonald Andrew D. and G. Kuhn Markus. <i>StegFS: A Steganographic File System for Linux</i> , University of Cambridge, 1999, CB2-3QG
NEC97	Necula, George C., <i>Compiling with Proofs</i> , School of Computer Science, Carnegie Mellon University. Pittsburgh, PA 15213, 1998, CMU-CS-98-154,
LEA94	Leach Ronald J, <i>Advanced topics in UNIX: processes, files and systems</i> , 1994, ISBN:0-471-03663-3
OLE05	O'Leary, T. J. and O'Leary, L. I., <i>Computing Essentials Brief</i> , 12 edition, New York, USA, 2005, ISBN: 0072256478
RAD95	Peter V. Radatti. <i>Computer viruses in Unix networks</i> . Technical report, Photonics East 1995 October 25, 1995 (SPIE, International Society for Optical Pennsylvania Convention Center Engineering) Philadelphia, PA Cybersoft, Inc., 1995.
RIS01	Riser Neil B. <i>An overview of some the current spoofing threats</i> , Version 1.2d, The SANS institute 2001.
RIT79	Dennis Ritchie. 1979. <i>On the security of UNIX</i> . Technical report, AT&T Bell Laboratories.
STA94	Stallings William, <i>SNMP, SNMPv2 and CMPI : The practical Guide to Network-Management Standards</i> , 1994, ISBN: 0201633310

TAN03	Tanenbaum Andrew S, <i>Redes de Computadoras</i> , 4ta edición, USA, 2003, ISBN: 9702601622
THO84	Ken Thompson. <i>Reflections on trusting trust</i> . Communications of the ACM, 27(8), 1984.
VAN98	van Schaik Carl and Smeddle Paul. <i>A Steganographic File System Implementation for Linux</i> . Technical report, University of Cape Town (South Africa), 1998.
VEL03	Velasco Elizondo, Perla Inés. <i>PSP: Una Alternativa Para Mejorar Los Procesos Del Software</i> , Newsletters ,Año 12, Vol. 39 y 40,LANIA(Laboratorio Nacional De Informática Avanzada A.C.), Primavera Verano 2003
VIC94	Vickers Bob. <i>Guide to safe X</i> , (Systems Group, ULCC) University Of London Computer Center, 1994
VIL12	Villalón Huerta, Antonio, <i>Seguridad en Unix y Redes</i> , Tesis maestría, Versión 1.2 ,GNU free documentación license, 2002.