DETECCIÓN DE INTRUSOS EN HOST

TESIS

Que para obtener el grado de
MAESTRÍA EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta
Pedro Chávez Lugo

Dr. Juan José Flores Romero
Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Agosto 2005

Dedico éste trabajo a Lety, mi esposa y a mi hijo Alex.

Agradecimientos

Quiero agradecer al Posgrado de la Facultad de Ingeniería Eléctrica de la Universidad Michoacana de San Nicolás de Hidalgo, la oportunidad que me ofreció para realizar mis estudios de Maestría.

Agradezco al Dr. Juan José Flores Romero, su amistad, confianza depositada en mi persona y la oportunidad que me brindo de trabajar en equipo.

Al Dr. Juan Manuel García García le agradezco su amistad, su colaboración y disponibilidad.

Les agradezco al M.C. Luis R. Rusiles Zamora, Dr. Felix Calderón Solorio, Dr. Leonardo Romero Muñoz, su amistad brindada, así como también por sus comentarios acertados y oportunos sobre este trabajo.

Agradezco a todos y a cada uno de mis maestros que me guiaron por el sendero del conocimiento. Gracias maestros por compartir su amistad y conocimientos.

Resumen

El exponer información sensible a la Internet, implica contar con herramientas de seguridad que garanticen la detección de violaciones de seguridad. El presente trabajo ofrece una herramienta de seguridad que provee el registro y análisis de información en busca de intrusiones. Para ésto se construyen perfiles de usuario, los cuales representan la abstracción del comportamiento que exhiben los usuarios en base a ejecuciones de programas y comandos. Para cada usuario se tienen dos perfiles, uno se caracteriza en base a su historia de ejecuciones y otro se caracteriza dinámicamente. Con la medida estadística del coeficiente de correlación obtenemos una referencia de la semejanza entre ambos perfiles, para saber si el usuario es quien dice ser o es un intruso. En otras palabras, la diferencia en perfiles es un indicador de diferencia en comportamientos, lo cual nos alerta de la posibilidad de un usurpación de cuenta y derechos. Concluimos con la necesidad de incluir otros perfiles para caracterizar de una manera más completa a los usuarios. Dichos perfiles incluirán información adicional sobre las preferencias y necesidades de los usuarios con el sistema (red, frecuencia de sesiones, tiempo de sesiones, etc.). Se hicieron experimentos con 24 usuarios más el superusuario (root). Los resultados fueron buenos en términos de detecciones de intrusión con los usuarios.

Abstract

Exposing sensitive information to the Internet, arises the need security tools that guarantee the detection of security violations. The present work offer a security tool that provides recording and analysis of information in search for intrusions. With this in mind, we profile users, a profile includes the behavior that a users exhibits, in terms of executions of programs and commands. Two profiles we keep; one characterizes its historical behavior and another one characterizes it dynamically. Using correlation coefficients, we obtain a reference of how close both profiles are; this way we determine if the user is who he or she claims or if we are facing an intrusion. In other words, the difference in profiles is an indicator of a difference in behaviors, which in turns alert us of a possible steal of an account and its associated rights. We conclude that we need to include other profiles to characterize users more completely. Such profiles will include additional information about the user preferences and resource usage (network, frequency of sessions, length of sessions, etc.). We made experiments with 24 usuarios plus root. The results were acceptable in terms of intrusion detection in the users.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

r	Coeficiente de correlación.
n	Número de comandos y programas.
x	variable aleatoria x.
y	variable aleatoria y.
$\overline{x}$	Media aritmética para valores de X.
$\overline{y}$	Media aritmética para valores de Y.
rda	Registro de Auditoría.
$Móvil$	Perfil móvil.

Capítulo 1

Introducción

En los sistemas de cómputo se tiene un modelo estándar de seguridad [Red03a], **CID** (*Confidencialidad, Integridad, Disponibilidad*) o por sus siglas en Inglés **(CIA)**[1]. Este modelo de seguridad promueve que los recursos del sistema (datos, programas, parámetros para control de acceso, llaves criptográficas, memoria, tiempo de CPU, etc.) a proteger mantengan los siguientes atributos: Confidencialidad, Integridad y Disponibilidad.

La confidencialidad implica que los recursos no sean leídos o adquiridos por personas no autorizadas. La integridad implica que los recursos no sean alterados por personas no autorizadas. La disponibilidad implica que los recursos deben estar disponibles a las personas autorizadas en el momento que se requiera.

La palabra *intrusión* [Webster94] se refiere a la acción de introducirse sin derecho a un sitio e *intruso* [Webster94] es la persona que realiza la intrusión. En los sistemas de cómputo ambos conceptos son aplicados en el contexto de la seguridad. La intrusión puede ser definida [Mukkamala02] como un conjunto de acciones que intentan comprometer la *confidencialidad*, *integridad* y la *disponibilidad* de los recursos del sistema.

Los intrusos se valen de ciertos tipos ataques[2] (ver Tabla 1.2) para ingresar a un sistema de cómputo específico. La Figura 1.1, muestra un ataque común de red [CERT03] utilizado para conseguir una intrusión.

[1] del Inglés *Confidentiality, Integrity, Availability.*
[2] Acción conducida por un adversario a una víctima.

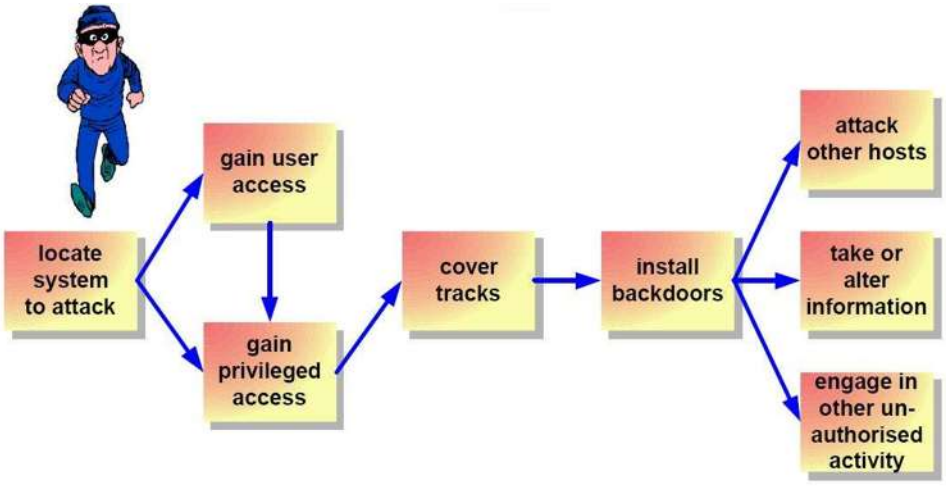


Figura 1.1: Ataque común de red

En la Figura 1.1, el atacante fija un sistema de cómputo como meta para una intrusión. El ataque de tipo sondeo (ver Sección 1.0.1) permite identificar vulnerabilidades[3] o huecos que tienen asociados los sistemas de cómputo. Una vez identificado el sistema de cómputo y las vulnerabilidades o huecos, el atacante puede hacer uso de explotaciones[4] (del Inglés *exploits*) de los ataques del tipo Usuario a Superusuario o Remoto a Local, para ingresar (intrusión) de una u otra forma al sistema. Una vez conseguida la intrusión y comprometida la cuenta del administrador, el intruso procede a eliminar toda evidencia originada de los ataques e instala puertas traseras (por ejemplo rootkit[5]), las cuales le permitirán ocultar la intrusión y el libre acceso al sistema. Una Intrusión puede tener varios objetivos: atacar a otros sistemas, robar o alterar información, realizar actividades no autorizadas.

1.0.1. Tipos de Ataques

En el año de 1998 la DARPA (del Inglés *Defense Advanced Research Projects Agency*) [ofDefense05]), lanzo una convocatoria para realizar evaluaciones en la detección de intrusión [Mukkamala02]. En esta convocatoria participaron varias empresas desarrolla-

[3]Una característica o falla en un sistema o programa que permite a un atacante pasar las medidas de seguridad.

[4]Proceso de usar una vulnerabilidad para violar una política de seguridad.

[5]Colección de herramientas que usa un intruso para ocultar la intrusión.

doras de software e instituciones de educación. Para esta evaluación fue necesario utilizar ataques que permitiesen realizar una intrusión. Los ataques empleados se clasificaron en las siguientes categorías:

- DOS: (del Inglés *Denial of Service*) Negación de Servicio.
  Estos ataques intentan mantener ocupado algún o algunos recursos del sistema de cómputo, para impedir el manejo de solicitudes válidas.

- U2R: (del Inglés *User to Root*) Usuario a Superusuario.
  Estos ataques intentan obtener el acceso a la cuenta del superusuario (administrador) desde una cuenta de un usuario normal.

- R2L: (del Inglés *Remote to Local*) Remoto a Local.
  Estos ataques tienen la finalidad de explotar una vulnerabilidad que permita obtener el acceso a una cuenta de usuario de una máquina local desde una máquina remota.

- Sondeo: Búsqueda de Vulnerabilidades o Huecos
  Los ataques de sondeo consisten en revisar una red de computadoras, en busca de vulnerabilidades y servicios disponibles.

La Tabla 1.2, muestra los tipos de ataques y sus correspondientes explotaciones conocidas.

Tabla 1.1: Lista de ataques difundidos por DARPA

Clase de ataque	Explotación	Clase de ataque	Explotación
	Apache2		Dictionary
	arppoison		ftpwrite
	Back		Guest
	Crashiis		Httptunnel
	dosnuke		Imap
	Land		Named
	Mailbomb		ncftp
Negación de	SYN Flood (Neptune)	Remoto a	netbus
servicio	Ping of Dead (POD)	local	netcat
	Process Table		phf

Tabla 1.2: Continuación de lista 1.2, Ataques Difundidos por DARPA

Clase de ataque	Explotación	Clase de ataque	Explotación
Negación de servicio	selfping Smurf sshprocesstable Syslogd tcpreset Teardrop Udpstorm	Remoto a local	ppmacro Sendmail sshtrojan Xlock Xsnoop
Usuario a Superusuario	anypw casesen Eject Ffbconfig Fdformat Loadmodule ntfsdos Perl Ps sechole Xterm yaga	Sondeo	insidesniffer Ipsweep Is_domain Mscan NTinfoscan Nmap queso resetscan Saint Satan

Sin duda el arma más efectiva para cualquier atacante o intruso, es la habilidad de encontrar defectos de un sistema, que puede que no sean evidentes para quienes lo diseñaron o para quienes lo utilizan a diario. Se conocen varias formas para lograr introducirse sin permiso a un sistema. Según Bob Toxen [Toxen00], los siete pecados mortales son:

- Contraseñas débiles

- Puertos de red abiertos

- Versiones no actualizadas de software

- Seguridad física pobre

- CGIs [NCSA98] (del Inglés *Common Gateway Interface*) inseguros

- Cuentas viejas e innecesarias

- Demora en actualizaciones por parte del administrador

La Figura 1.2 muestra los estudios de CERT (del Inglés *Computer Emergency Response Team*) que evidencian el crecimiento de las vulnerabilidades. Esto trae como consecuencia la necesidad de contar con las actualizaciones necesarias de software.

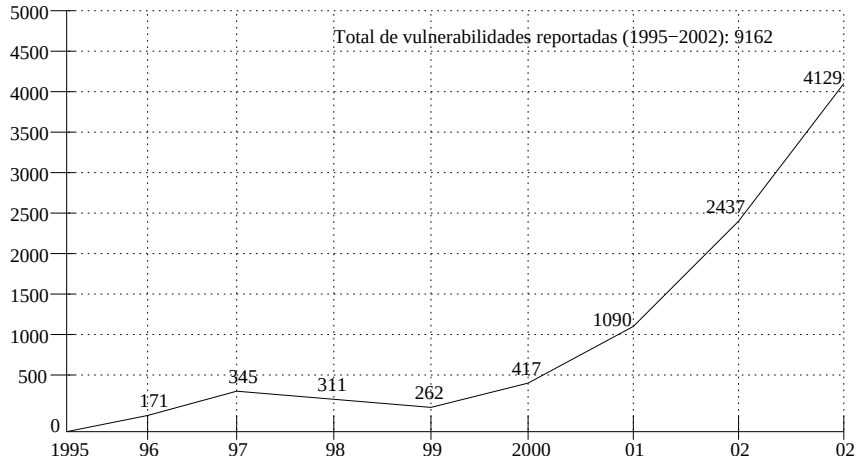


Figura 1.2: Vulnerabilidades reportadas por CERT desde 1995-2002

Diariamente se conocen nuevas vulnerabilidades. Para una referencia actualizada, se recomienda consultar la base de datos de ataques y explotaciones ofrecida por DARPA ([Lincold Laboratory99], CERT [CERT05]).

1.1. Hechos Relevantes para la Seguridad en Cómputo

Algunos eventos han contribuido al nacimiento y crecimiento de la seguridad de cómputo. A continuación se listan algunos de los más importantes desde los años sesentas hasta los noventas, tomados de [Red03a].

1.1.1. Los Años 60's

Estudiantes del MIT (del Inglés *Massachusetts Instute of Technology*) miembros TMRC (del Inglés *Tech Model Railroad Club*), inventaron el termino "hacker" bajo el contexto que hoy es conocido e iniciaron con la exploración y programación de un sistema de cómputo PDP-1[6].

A mediados de la década de 1960, el DoD (del Inglés *Department of Defense*) de Estados Unidos quería una red de comando y control que pudiera sobrevivir a una guerra nuclear. Mediante ARPA (del Inglés *Advanced Research Agency*) lanzo una convocatoria y de esta surguio la ARPANet. Esto dió pausa para la creación de la red conocida en la actualidad como Internet.

El Sistema Operativo Unix fue diseñado a finales de los años sesenta y principios de los años setenta por Ken Thompson de AT&T. El núcleo de Unix fué escrito originalmente en lenguaje ensamblador y reescrito en 1973 en el lenguaje de programación C.

1.1.2. Los Años 70's

Bolt, Beranek y Newman, investigadores y desarrolladores contratados por el gobierno y la industria de Estados Unidos, desarrollaron el programa telnet para una extensión pública de ARPANet. Esto abrió las puertas al usó público de las redes de datos que estaba restringida al gobierno e investigadores académicos. Telnet, es el programa más inseguro para redes públicas, por no contar con la opción de cifrar información.

Steve Jobs y Steve Wozniak fundaron la compañia *Apple Computer* e iniciaron la venta de Computadoras Personales (PC).

Jim Ellis y Tom Truscott crearon USENET, un boletín informativo electrónico para la comunicación entre usuarios distantes. USENET pasó a ser uno de los foros más populares para el intercambio de ideas en computación, redes y por supuesto cracking.

En 1978 nació el lenguaje C, con la publicación de *The C Programming Languaje* por Brian Kernighan y Dennis Ritchie. Desde su nacimiento C, ha mostrado eficacia y potencia, ya que este lenguaje no está practicamente asociado a ningún sistema operativo, ni a ninguna máquina. Es por esta razón fundamental, que C es conocido como el lenguaje de programación de sistemas por excelencia.

[6]Computadora marca Digital.

1.1.3. Los Años 80's

La IBM desarrolló y vendió PCs basadas en el microprocesador 8086 de Intel, la cual era una arquitectura barata para el cómputo de oficina y casero.

El Protocolo de Control en la Transmisión (TCP), fue desarrollado por Vint Cerf, el cual fue dividido en dos partes. Una parte fue nombrada como el protocolo de Internet. De la combinación de ambas partes se obtuvo el protocolo TCP/IP, el cual es en la actualidad un estándar para toda la comunidad en Internet.

La Legión de *Doom* y *Chaos Computer Club* son dos grupos pioneros de hackers que iniciaron con la explotación de vulnerabilidades en computadoras y redes.

En 1986 el acto de fraude y abuso de cómputo fue anexado a las leyes de Estados Unidos por su congreso. Estas leyes se basaron en la explotación de Ian Murphy, también conocido como el capitán Zap, el cual ingresó a computadoras militares, hizo varios pedidos a algunas compañías y realizó llamadas desde el sistema telefónico del gobierno.

El CERT fue creado para alertar a los usuarios sobre incidentes[7] de seguridad.

Probablemente la mayor violación de seguridad de todos los tiempos [Tanenbaum97] sucedió la noche del 2 de Noviembre de 1988, cuando un estudiante de Cornell, llamado Robert Tappan Morris, disemino el programa de un gusano[8] (del Inglés *Worm*) que inhabilito a miles de computadoras en todo el mundo. Para reproducirse el gusano explotaba una vulnerabilidad que le permitía obtener las claves de los usuarios y para su propagación utiliza a las tablas de ruteo. Morris no tuvo mucho que investigar, pues obtuvo información confidencial de su padre, un experto de seguridad de la NSA[9] de Estados Unidos. Morris fue sentenciado a pagar una multa por 10,000 dolares, 3 años de libertad condicional y 400 horas de servicio a la comunidad. Se estima que los gastos del juicio excedieron los 150,000 dolares.

1.1.4. Los Años 90's

La ARPANet es decomisada y su tráfico fue transferido a Internet. Linus Torvalds desarrolló el kernel de Linux para ser usado como sistema operativo de GNU. Linux es más popular entre los hackers, quienes lo encuentran útil para construir sistemas seguros.

[7] Evento adverso en el cual algún aspecto de la seguridad de un equipo de cómputo puede estar amenazado
[8] Software malicioso que reproduce copias de si mismo
[9] del Inglés *National Security Agency*

Vladimir Levin y sus cómplices transfirieron 10 millones de dólares en varias cuentas de la base de datos central de *CityBank*. Levin fue arrestado y se dice que todo el dinero fue recuperado.

Un estudiante Israelita de 19 años fue arrestado y condenado por ingresar a los sistemas del gobierno de Estados Unidos, durante la guerra del Golfo Pérsico. Los oficiales militares llamaron a este suceso como el más organizado y sistemático ataque ocurrido en la historia de los sistemas del gobierno.

El control de los satélites de Inglaterra fue tomado por varios delincuentes, finalmente el gobierno retomó el control.

1.1.5. La Seguridad en la Actualidad.

En Febrero del 2000 el ataque DDOS (del Inglés *Distributed Denial of Service*) fue desencadenado en varios de los sitios más frecuentados en Internet. El ataque afectó a yahoo.com, cnn.com, amazon.com, fbi.gov y otros sitios dejándolos completamente inalcanzables para los usuarios normales.

En la actualidad se estima que 400 millones de personas usan o hacen uso de Internet al mismo tiempo. En el primer cuarto del 2002, el número de incidentes reportados por CERT llegó a 73,359 de 52,658 reportados en el 2001.

El incidente más recientemente ocurrido es el robo de datos de 40 millones de tarjetas de crédito [mas05]. Dicho incidente fue reportado al FBI el 22 de Mayo del 2005 por Visa y MasterCard. La empresa Mastercard culpa a CardSystems Solutions, ya que la falla se atribuye al procesador. Mastercard asegura que los datos sensibles (número de seguro social, fecha de nacimiento, etc.) de los clientes afectados no están incluidos en el robo.

Estudios realizados por CERT [Allen00], indican que el crecimiento de Internet ha generado un incremento de incidentes de seguridad (ver Figura 1.3).

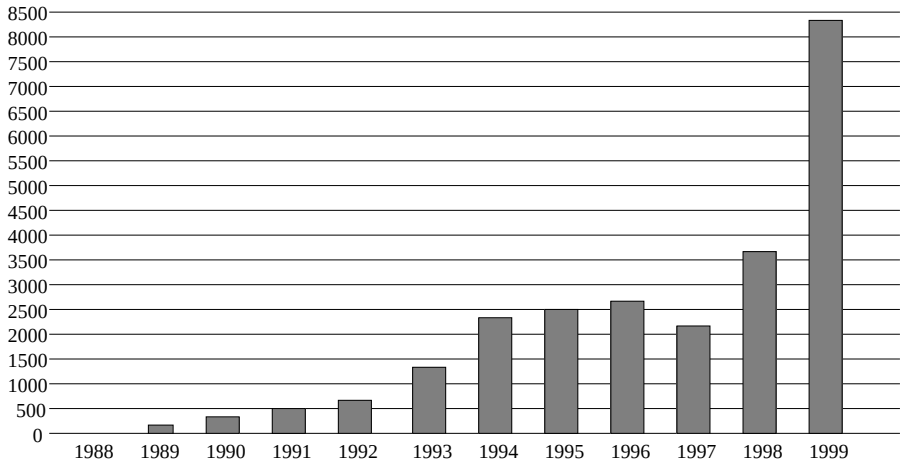


Figura 1.3: Crecimiento de incidentes en Internet según CERT 1988-1999

Los últimos estudios realizados por CERT [CERT03], revelan que del año 1998-2002 se registraron 182463 incidentes y con tendencia al aumento (ver Figura 1.4).

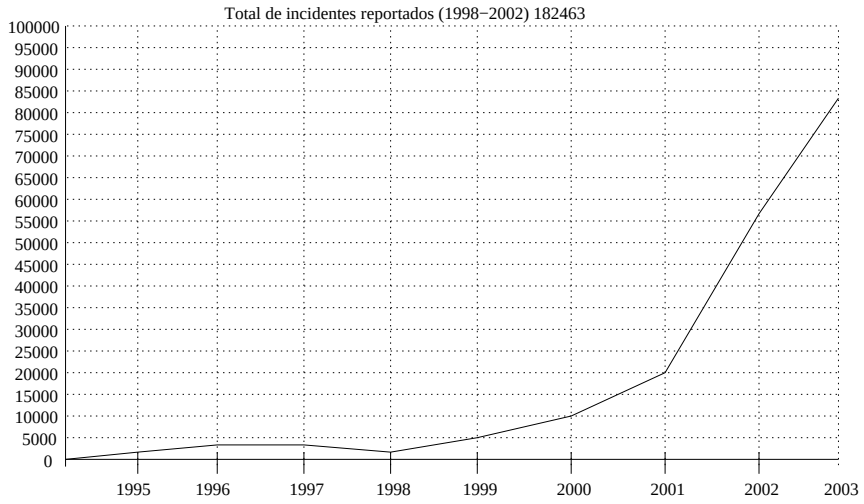


Figura 1.4: Crecimiento de incidentes en Internet según CERT 1995-2002

Los estudios de CERT [CERT03], indican que el grado de conocimientos técnicos para los atacantes es muy bajo, ya que en la actualidad existe un gran número de herramientas desarrolladas (ver Figura 1.5) para este fin. Lo cual permite obtener una alta complejidad. El contar con los medios de protección adecuados pudiera garantizar que los in-

trusos, atacantes o gente maliciosa no ingrese al sistema, evitando la observación, alteración o eliminación de información. Una librería disponible para ataque es libnet [Schiffman00], la cual ofrece la facilidad de crear e inyectar paquetes a la red. Los parámetros de los paquetes pueden ser manejados a conveniencia o gusto, lo cual permite crear explotaciones. Algunas herramientas que hacen uso de esta librería son: Nemesis [nem05] y nmap [nma05].

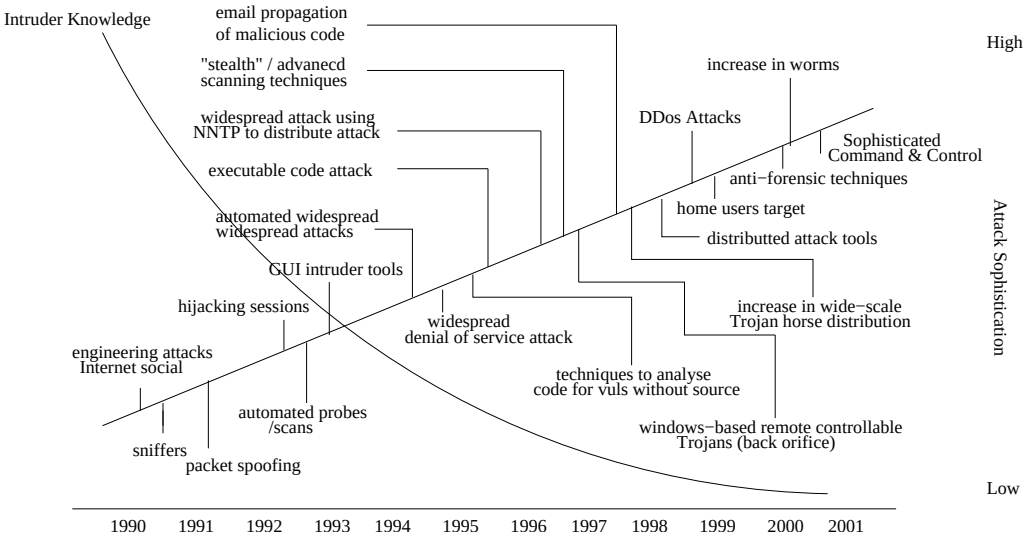


Figura 1.5: Nivel de conocimientos de atacantes y complejidad de los ataques según CERT

Con el crecimiento de la Internet y el aumento de vulnerabilidades e incidentes, la seguridad de cómputo aparece en el horizonte como un problema potencial de grandes proporciones. Es por esta razón que se requiere de nuevas alternativas para salvaguardar los recursos de los sistemas de cómputo. Son varios proyectos que actualmente se están desarrollando y que darán notables ventajas sobre los ya existentes; tal es el caso de SELinux[10] [sel05], LIDS[11] [lid05], entre otros. Dichos proyectos intentan ofrecer una nueva arquitectura de sistema operativo, que elimine los problemas de un gran número de arquitecturas actuales.

[10] del Inglés *Security Enhanced Linux*
[11] del Inglés *Linux Intrusion Detection System*

1.2. Sistema Detector de Intrusión (IDS)

Un sistema detector de intrusión (IDS)[12] [Allen00] tiene la finalidad de encontrar evidencias de intrusiones en un sistema de cómputo o red. Los IDS se pueden clasificar [Allen B. Tucker00] como *pasivos* o *activos*. Los sistemas pasivos, generalmente operan fuera de línea, analizando los eventos y verificando si éstos representan una posible intrusión. Los sistemas activos analizan los eventos y los verifican en tiempo real.

La detección de intrusión [Allen00] se pueden clasificar como *mal uso* (MID)[13] o *anomalía* (AID)[14]. El concepto de anomalía se refiere a las desviaciones del comportamiento, mientras que el mal uso se refiere al reconocimiento de patrones para ataques conocidos, muchos de los cuales intentan realizar daños en los sistemas. La anomalía por otra parte requiere contar con un historial del comportamiento, para realizar comparaciones entre éste y nuevos comportamientos. Cabe mencionar que cualquier nuevo comportamiento será anómalo, aunque este sea válido.

Al utilizar el modelo MID se debe contar con una base de datos que contenga las firmas de los ataques. Desafortunadamente, al emplear este modelo se corre el riesgo de no detectar nuevos ataques e intrusiones. El modelo AID permite conocer nuevos tipos de ataques e intrusiones, pero desafortunadamente este modelo requiere una gran cantidad de datos almacenados. Estas desventajas tienden a desaparecer con el mejoramiento de las capacidades de los sistemas de cómputo.

La detección de intrusión puede efectuarse en *Host*[15] o *Red* y la función principal de un sistema detector de intrusión usualmente no es efectuar una acción para prevenir un ataque, sino alertar al administrador sobre violaciones de seguridad causadas por un intruso o atacante.

1.2.1. Características Deseables para los IDS

Según [Sundar98] y [Crosbie95], los IDS deben contar con un cierto número de características para garantizar un funcionamiento correcto. Las características deseables de un IDS son:

1 Trabajar continuamente con un mínimo de supervisión humana.

[12] del Inglés *Intrusion Detection System*.
[13] del Inglés *Misuse Intrusion Detection*.
[14] del Inglés *Anomaly Intrusion Detection*.
[15] computadora que trabaja como anfitrión.

2 Ser tolerante a fallas, cuando se presente una falla en su funcionamiento, éste tenga la capacidad de auto iniciarse.

3 Contar con la posibilidad de la supervisión de si mismo, para detectar cambios realizados por un intruso.

4 Poder ser configurado en base a nuevas necesidades.

5 Requerir un mínimo de recursos del sistema para su funcionamiento.

6 Su implementación permita la posibilidad de adaptaciones o modificaciones.

Estas características no sólo constituyen un estándar a cumplir para los IDS, sino por un gran número de herramientas de seguridad. Toda herramienta de seguridad que no cumpla con la mayoría de estas características estará condenada a desaparecer y quedar en el olvido. El documento *"Intrusion Detection: A Bibliography"* [Me01], ofrece una lista de un gran número de investigaciones e implementaciones de IDS, desde el año 1980 al 2001. Muchas de las ideas citadas en [Me01] han permanecido, pero otras han marcado solamente su paso.

1.2.2. IDS Basados en Red (NIDS)

Los sistemas detectores de intrusión basados en Red (NIDS)[16], capturan y analizan los paquetes de red que se envían o se reciben. Para la captura de paquetes se emplea un "sniffer"[17]. Los IDS basados en Red pueden emplear para la detección de intrusión el esquema de mal uso, anomalía o ambos. El modelo de anomalía requiere de los comportamientos que exhiben los usuarios o el sistema en cuanto al uso de red se refiere, para detectar y reportar posibles desviaciones del comportamiento. Para el caso de mal uso, en los paquetes de red se buscan patrones de ataques conocidos y difundidos por la DARPA (ver Tabla 1.2). El esquema de mal uso es ampliamente utilizado en la mayoría de los NIDS.

Ventajas de NIDS

A continuación se listan algunas ventajas para los NIDS, según [Bace00]:

- Las redes que emplean NIDS pueden volverse más seguras.

[16] del Inglés *Network Intrusion Detection System.*
[17] herramienta para captura de paquetes de red, como *tcpdump* [Group05].

- Instalados en lugares estratégicos pueden analizar redes grandes.
- Las redes que emplean NIDS tienen muy poco impacto en su funcionamiento. Ya que generalmente los NIDS escuchan en la red trabajando como dispositivos pasivos[18].

La gran mayoría de IDS comerciales trabajan como NIDS.

Desventajas de NIDS

Según [Bace00], en la actualidad existen algunas desventajas para los NIDS:

- Dificultad de procesar paquetes en horas de gran tráfico, ya que el fallo en la detección de ataques lanzados en estos periodos pudiera darse. Para solucionar este problema algunas empresas han desarrollado NIDS en hardware, los cuales tienen una mayor rapidez para procesar paquetes.
- Los NIDS no pueden analizar paquetes cifrados [Lippmann98]. Este problema se incrementa en organizaciones que hacen uso de Redes Privadas Virtuales (VPN).
- Algunos NIDS tienen problemas para detectar ataques en paquetes fragmentados. Estos paquetes pueden causar inestabilidad y fallo del NIDS.
- La mayoría de los NIDS no pueden indicar si un ataque tuvo éxito o no; éstos solamente indican el inicio de un ataque. De manera manual el administrador debe investigar cada ataque para determinar si se logró una intrusión.
- Los NIDS comerciales tienen un costo, en su compra y actualizaciones.

Los NIDS proporcionan seguridad en la medida de sus posibilidades, ya que estos ofrecen ciertas ventajas y desventajas. Sin embargo independientemente de sus ventajas y desventajas, es imprescindible contar con una de estas herramientas de seguridad, siempre y cuando la información y el sistema a proteger amérita hacerlo.

1.2.3. IDS Basados en Host (HIDS)

Los IDS basados en Host (HIDS)[19] operan en un sistema de cómputo individual, su finalidad es detectar intrusos que se encuentran dentro del sistema. El intruso ya instalado

[18] Solo escucha la red y no genera tráfico a esta.
[19] del Inglés *Host Intrusion Detection System.*

en el sistema puede tomar el control de éste para realizar ataques o nuevas intrusiones. Es por esta razón que es necesario detectar rápidamente la intrusión. La detección se efectúa en base a las acciones que el intruso realiza en el Host. Los HIDS normalmente obtienen la información de dos fuentes, un sistema auditor [Zirkle05] y un sistema de reportes [Flegel00]. Ambas fuentes proporcionan información que proviene del kernel[20] del sistema operativo. Todo sistema operativo cuenta con un sistema de reportes; a diferencia de un sistema auditor, que no se tiene de manera directa, para tenerlo en algunos casos es necesario realizar cambios a nivel de kernel.

Al igual que los IDS basados en red, los basados en Host también pueden emplear para la detección de intrusión los esquemas de mal uso, anomalía o ambas. El esquema de anomalía es ampliamente usado en los HIDS. De manera general, una anomalía representa una actividad inusual que puede indicar una intrusión. Uno de los problemas con la detección por anomalía es la generación de un gran número de falsas alarmas, debido a las actividades inusuales pero legítimas. Una actividad inusual pero legítima es aquella que realiza el usuario y dicha actividad no había sido realizada previamente, ésto se puede deber al hecho que los usuarios exhibirán un comportamiento que dependerá de sus necesidades. Un usuario principiante que tome un curso de redes, posiblemente modificará su comportamiento debido a la adquisición de nuevo conocimiento, lo cual se refleja en su comportamiento.

La idea principal para la detección por anomalía es establecer un perfil normal, que represente su comportamiento normal o habitual, para poder encontrar desviaciones en su comportamiento exhibido. El comportamiento normal o habitual para un usuario tendrá registradas las preferencias del usuario con el sistema. Es decir [Ryan98], editores, compiladores, comandos, servicios, aplicaciones, direcciones IP frecuentadas.

Un intruso que supiese de la presencia de un HIDS basado en anomalía, pudiera disfrazar su comportamiento, lo cual implicaría que la modificación al comportamiento la efectúe paulatinamente. Este problema muy remoto pero pudiera darse en algunos casos, según Dorothy Denning [Denning86] más en aquellos en los que el intruso tiene la finalidad de espiar en el sistema y no afectar de alguna manera a éste u otros sistemas. Este problema es fácil de solucionar, pero la desventaja es la generación de un gran número de falsas alarmas, debido a cambios pequeños en los comportamientos.

[20] Parte principal del sistema operativo.

1.2.4. Modelo Clásico para Detección de Intrusión

Un modelo clásico de un IDS es el propuesto por Denning [Denning86], el cual propone la detección de intrusión en tiempo real, basado en la hipótesis que considera que es posible analizar y registrar las violaciones de seguridad. El modelo de Denning considera seis componentes principales:

- Sujetos: son los iniciadores de actividad, normalmente usuarios (terminal de usuario, proceso).

- Objetos: recursos administrados por el sistema (archivos, comandos, dispositivos, etc.).

- Registros de auditoría: registros generados en respuesta a las acciones realizadas por los sujetos sobre los objetos (ejecución de comando, acceso a archivo, etc.)

- Perfiles: estructuras que caracterizan el comportamiento de los sujetos con respecto a los objetos. Estos son generados automáticamente e inicializados con una lista de perfiles obtenidos previamente.

- Registros anómalos: generados cuando un comportamiento anómalo es detectado.

- Reglas de actividad: acciones tomadas cuando una condición es cumplida, tal como actualizar perfiles, detectar comportamiento anómalo, producir reportes.

El modelo incluye perfiles y reglas, los perfiles representan el comportamiento de los sujetos con respecto a los objetos, en términos de métricas y modelos estadísticos. Las reglas son utilizadas en la adquisición de nuevos comportamientos y para la detección de comportamiento anómalo. Los registros de auditoría forman parte de los eventos capturados y registrados por un sistema auditor. Los perfiles son caracterizados en base a las siguientes métricas y modelos estadísticos:

Métricas:

- Contador de eventos

  El número de eventos que ocurren durante un intervalo de tiempo. Por ejemplo, el número de veces que se ejecuta un comando durante una sesión, el número de claves falladas durante un minuto, etc.

- Intervalo de tiempo

  Es la cantidad de tiempo entre dos eventos relacionados, como la cantidad de tiempo entre ingresos sucesivos a una cuenta.

- Medición de recursos

  La cantidad de recursos consumidos por una acción durante un periodo. Tal como la cantidad de tiempo de CPU consumido por un programa durante una simple ejecución, el número de páginas impresas por un usuario en un día, etc.

Modelos estadísticos:

- Modelo Operacional

  Este modelo se basa en la experiencia para detectar anormalidades. Por ejemplo, número de claves falladas durante un breve periodo, considerando 10 fallas como un intento de intrusión.

- Modelo de la media y desviación estándar

  Este modelo se basa en la suposición de conocer todo acerca de los eventos de una misma naturaleza *x(1),...,x(n)*. Para nuevos eventos se usa la media y la desviación estándar en la determinación de anormalidades.

- Modelo Multivariable

  Este modelo es similar al modelo de la media y desviación estándar excepto que éste hace uso de varias métricas. Es por esta razón que es conocido como multivariable.

- Modelo de Proceso de Markov

  Este modelo es aplicado usando contadores de eventos. Es útil para buscar transiciones de comandos en donde el historial es importante.

- Modelo de Series de Tiempo

  Este modelo usa un intervalo de tiempo y un contador de eventos o un medidor de recursos. Una observación es anormal si los eventos ocurren en un tiempo muy corto.

El modelo de Denning intenta ofrecer una estructura general de un sistema experto detector de intrusión, independiente del sistema operativo, vulnerabilidades de éste y tipos de intrusión. Este modelo puede considerarse como un HIDS basado en reglas y búsqueda de patrones. Este ha sido un modelo general para un gran número de IDS ([Mukkamala02], [Ryan98], [Sundar98], [Crosbie95], entre otros).

1.3. Antecedentes

La tecnología de los sistemas detectores de intrusos es inmadura y se encuentra en una constante evolución [Bace00]. Los IDS han tenido un campo activo en la investigación por más de dos décadas. El inicio de esta tecnología se origino en el año de 1980, con la publicación del artículo *"Computer Security Threat Monitoring and Surveillance"*, por James Anderson [Anderson80]. Posteriormente en el año de 1987 el artículo *"An Intrusion Detection Model"* por Dorothy Denning. Este trabajo ofreció una estructura general e inspiró a muchos investigadores; en la actualidad ha servido como base de muchos productos comerciales.

Las investigaciones realizadas en los 80's y 90's arrojaron un gran número de nuevas herramientas (Haystack [Smaha88], MIDAS [Sebring88], IDES [Lunt88], W&S [Vaccaro89], Comp-Watch [Dowel90], NSM [Heberlein90], NADIR [Jackson91], Hyperview [Debar92], DIDS [Snapp92], ASAX [Habra92], USTAT [Ilgun93], DPEM [Fink94], IDIOT [Kumar94], NIDES [Anderson95], Janus [Golberg96], JiNao [Jou97], Bro [Paxon98], etc.). Sin embargo, muchas de éstas han tendido a quedarse en el olvido, ya que sus desarrolladores eran estudiantes motivados por la exploración de los conceptos de la detección de intrusión. De alguna forma, estos trabajos marcaron la dirección en las investigaciones y productos comerciales.

El enfoque principal de la tecnología de ID estaba dirigido a producir soluciones basadas en host, aunque con el crecimiento de las redes, el enfoque ha cambiado para producir sistemas basados en red. A continuación se mencionan algunos IDS sobresalientes:

- EMERALD

  EMERALD [sri05] (*Even Monitoring Enabling Responses to Anomalous Live Disturbances*), es la herramienta más reciente, obtenida de la investigación hecha por SRI Internacional. Su funcionamiento se basa en la búsqueda de desviaciones (anomalía) y patrones (mal uso).

- SNORT

  NIDS [org.05] de código abierto (mal uso), con capacidad de analizar paquetes en tiempo real, para búsqueda/match de ataques y sondeos, tales como buffer overflows, revisión de puertos, ataques CGI, etc.

- CISCO NetRanger

  NetRanger [Cisco05] realiza detección de intrusos en tiempo real, aplica una metodología con base en el conocimiento, representado por las firmas de ataques conocidos (mal uso).

- RealSecure 3.2

  RealSecure [Systems99] basa su detección en Host y Red. La detección se realiza mediante firmas de ataques (mal uso) y perfiles estadísticos (anomalía).

- Tripwire

  Tripwire [Tri05] es un HIDS para Linux, se basa en la verificación de integridad de archivos mediante el algoritmo MD5[21]. Si un archivo es alterado (tamaño, fecha modificación o eliminado) y este se incluye en los registros de Tripwire, su alteración será detectada.

En la actualidad ninguna computadora o red es completamente segura y alguien que piense estarlo, sin duda está en un error. La buena administración y el uso de herramientas que permitan detectar huecos o vulnerabilidades e intrusos, son hábitos que deben poseer los buenos administradores, para tratar de conseguir un sistema seguro.

Todos los días se conocen nuevas vulnerabilidades, tanto en sistemas operativos como en aplicaciones [CERT05]. Desafortunadamente, el ataque más difícil de eliminar es según Galaviz y Magidin [Casas02], el ataque de garrote o ingeniería social. El ataque de garrote es el menos elegante, pero ha probado ser el más efectivo; consiste en golpear, sobornar, chantajear, engañar o seducir a alguien para obtener información. Este ataque fue usado y comprobado con éxito por Kevin Mintnick [Kevin D. Mitnick02]. Mintnick demostró que el problema más grande de seguridad en los sistemas de cómputo, es el factor humano. En su libro redacta como utilizó al factor humano para sustraer, eliminar y modificar información altamente confidencial para el gobierno Estadounidense.

[21] Algoritmo que obtiene una clave de un archivo, basada en sus características.

1.4. Planteamiento del Problema

Se plantea obtener una base sólida para una herramienta de seguridad que encuentre anomalías en el comportamiento de los usuarios. Tales anomalías pueden representar intrusiones, lo cual implica una amenaza de seguridad, ya que se desconoce cuál es la intención del intruso (robar información, alterar información, realizar ataques, etc). Será necesario contar con una medida que proporcione una referencia del comportamiento que exhiban los usuarios.

De manera inicial se pretende obtener el comportamiento de los usuarios en base a ejecuciones de comandos y programas. Para ésto se requiere contar con mecanismos que tengan capacidad de auditar eventos en el sistema operativo. Será necesario realizar adaptaciones en los mecanismos, ya que se pretende obtener la información del evento de ejecución en tiempo real. También será necesario contar con los medios que permitan obtener información de los comportamientos de una manera rápida y eficiente.

1.5. Objetivos de la Tesis

Los objetivos de ésta tesis son:

- Construir perfiles de usuario que representen las ejecuciones locales de comandos y programas. Para esto se emplea a la herramienta SNARE, la cual tuvo que ser adaptada debido a las necesidades del presente trabajo (ver Capítulo 3).

- Detectar intrusiones mediante la detección de anomalías encontradas en los perfiles de usuarios. El coeficiente de correlación es la medida estadística empleada en la detección de anomalías (ver Sección 2.7.1).

- Cumplir con las características 1, 4, 5 y 6, citadas en la Sección 1.2.1.

- Procesar la información en tiempo real.

- Ofrecer una base sólida para trabajos futuros (ver Sección 5.2)

De manera muy general, lo que se pretende es obtener un HIDS que realice la detección de intrusión mediante el esquema de anomalía, usando para esto la medida estadística del coeficiente de correlación.

1.6. Descripción de Capítulos

El Capítulo Dos describe al modelo empleado en el HIDS desarrollado y sus componentes (sujetos, objetos, registros de auditoría, perfiles, registros anómalos y reglas de actividad) de éste. Se describe la medida estadística del coeficiente de correlación lineal empleada para la detección de intrusión, aplicadas a los perfiles de usuario. Se dan algunos ejemplos del coeficiente de correlación lineal y se fijan los valores que evidencian una intrusión.

El Capítulo Tres proporciona da una introducción a la herramienta de SNARE, describiendo sus partes, configuración y funcionamiento. Se dan las razones del porqué de su adaptación. También se describen las partes de la adaptación (funciones adicionadas). Se describe brevemente a la librería NDBM[22], la cual proporciona los recursos para el manejo de bases de datos utilizadas en la adaptación.

El Capítulo Cuatro habla de los resultados obtenidos, el lugar en qué se realizó la prueba, el número de usuarios participantes y su duración. Se muestra la gráfica del acumulado de los comandos y programas capturados. Se muestran algunos ejemplos que muestran los valores del coeficiente de correlación obtenidos de perfiles de usuario.

El Capítulo Cinco cita las conclusiones generales que se derivan del presente trabajo. Se exponen los trabajos futuros propuestos, que permitirán a la presente ser una buena opción para el control de la seguridad en los sistemas de cómputo.

[22]del Inglés *New Database Manager*.

Capítulo 2

Modelado

2.1. Introducción

El modelo propuesto por Denning [Denning86], ha ofrecido una estructura general para varios IDS desarrollados. Una prueba de ello es el presente trabajo, el cual también hace uso de dicha estructura. Para la detección de intrusión en Host, se usa al igual que Denning el esquema de anomalía. Dicho esquema detecta anomalías o anormalidades en los comportamientos de los usuarios.

2.2. Descripción del Modelo

Planteamos como base de este trabajo la hipótesis que considera que el comportamiento de los usuarios tiende a la estabilidad en un periodo de tiempo muy grande. Consideramos manejar dicho comportamiento en perfiles. Esto nos lleva a tener para los usuarios un perfil acumulado, el cual se caracteriza en base a al acumulado de ejecuciones. También se incluye un perfil dinámico o móvil, para obtener una caracterización dinámica o temporal del usuario en un periodo de tiempo menor que el manejado en el perfil acumulado. La finalidad es comparar ambos perfiles y obtener un valor que represente la diferencia entre las caracterizaciones acumulada y móvil. De esta forma se tendrá una medida que indique la semejanza entre ambos perfiles o caracterizaciones. Una diferencia marcada entre ellos es un indicador de intrusión.

La caracterización dinámica requiere un periodo de tiempo fijo, que puede constar de varios muestreos. Este tiempo fijo lo nombramos “Intervalo”. Pensamos que el tamaño

óptimo del intervalo es aquel que disminuye el número de falsos positivos[1] y falsos negativos[2]. Obtener este tamaño óptimo implica recaudar grandes cantidades de información, un gran número de usuarios participantes y una amplia coordinación con estos. En el presente trabajo no se consideró obtener al valor óptimo del periodo de muestreo, ya que se considera a ésta una tarea ardua que implica meses o quizás años de captura y procesamiento de información. El modelo empleado tiene seis componentes principales:

- Sujetos: usuarios.
- Objetos: comandos y programas.
- Registros de auditoría: bases de datos, generadas de acciones realizadas por los sujetos sobre los objetos.
- Perfiles (acumulado y móvil): bases de datos que caracterizan el comportamiento de los sujetos con respecto a los objetos.
- Registros anómalos: generados cuando un comportamiento anómalo es detectado en el perfil acumulado y móvil. Esto se logra con la medida estadística del coeficiente de correlación lineal.
- Reglas de actividad: acciones tomadas cuando una condición es cumplida, tal como actualizar perfiles, detectar comportamiento anómalo, producir reportes.

Son muy pocas la variaciones realizadas a los componentes del modelo de Denning. Para nuestro caso el componente perfil se divide en dos partes: perfil acumulado y móvil.

2.3. Sujetos y Objetos

Los sujetos son los iniciadores de actividad. Inicialmente en el presente trabajo solo se consideraron como sujetos válidos a los usuarios cuyo *uid* cumple que:

$$499 < uid < 65534$$

es decir, solo usuarios normales. Por recomendación de los revisores se incluye también como sujeto válido al superusuario (root), cuyo *uid*= 0. Uno de los principales problemas

[1] Evento que un IDS determina como intrusión cuando verdaderamente no lo es.
[2] Evento que un IDS determina que no es una intrusión y verdaderamente sí lo es.

[Mark Mitchell01] para ciertos programas (init, dhcp, kevent, syslogd, X, etc.) de los sistemas operativos estándares basados en Unix, es requerir la identidad del superusuario, ya que éstos realizan operaciones especiales. Esto quiere decir, que el superusuario tendrá en su comportamiento una parte que es atribuida a dichos programas. Por lo tanto, el comportamiento del superusuario se divide en dos partes: comportamiento del sistema operativo y el comportamiento del administrador.

El conjunto de los objetos se limita sólo a los programas y comandos que puedan ejecutarse en el sistema. Cabe mencionar también son considerados objetos los archivos, dispositivos, etc. Los cuales no son manejados en este trabajo.

La ejecución es la única acción que se registrada generada cuando un sujeto realiza el proceso de ejecución de un objeto.

2.4. Registros de Auditoría

Los registros de auditoría contienen al conjunto de acciones realizadas por el sujeto sobre los objetos en un muestreo de un día. Estos registros se representan mediante bases de datos nombradas bases de datos para ejecuciones diarias (ver Figura 2.1). Como se menciona en la Sección 3.4.1, la librería NDBM es empleada para el manejo de estas bases de datos. Cada usuario tendrá sus propios registros de auditoría o bases de datos para ejecuciones diarias. Una base de datos se crea en el instante en que el sujeto realiza la primera acción del día. El valor del intervalo fija al número máximo de bases de datos para cada sujeto.

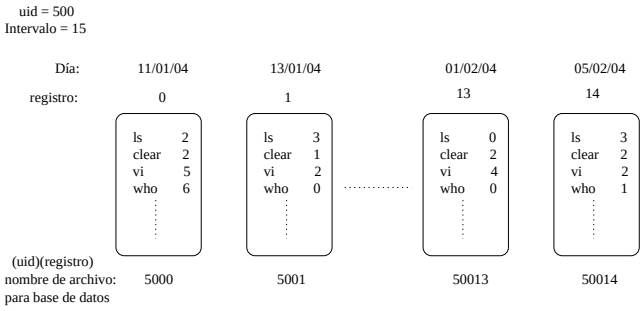


Figura 2.1: Registros de Auditoría

La Figura 2.1, muestra al conjunto de bases de datos de ejecuciones diarias del usuario *uid*=500, con un intervalo de 15 días. De tal manera que el conjunto de registros

de auditoría o bases de datos para ejecuciones diarias tendrá 15 elementos 0,1,2,...,14. El término "Registro de Auditoría" se abreviará como *rda*.

2.5. Perfiles

Los perfiles son bases de datos que representan los comportamientos de los usuarios con respecto a los objetos. Cada sujeto tiene asociado dos perfiles: acumulado y móvil. El perfil acumulado como se comenta en la Sección 2.2 es la caracterización de manera general del sujeto.

El perfil acumulado se representa por las bases de datos para ejecuciones acumuladas. En la Figura 2.2, se muestra el perfil acumulado de un sujeto.

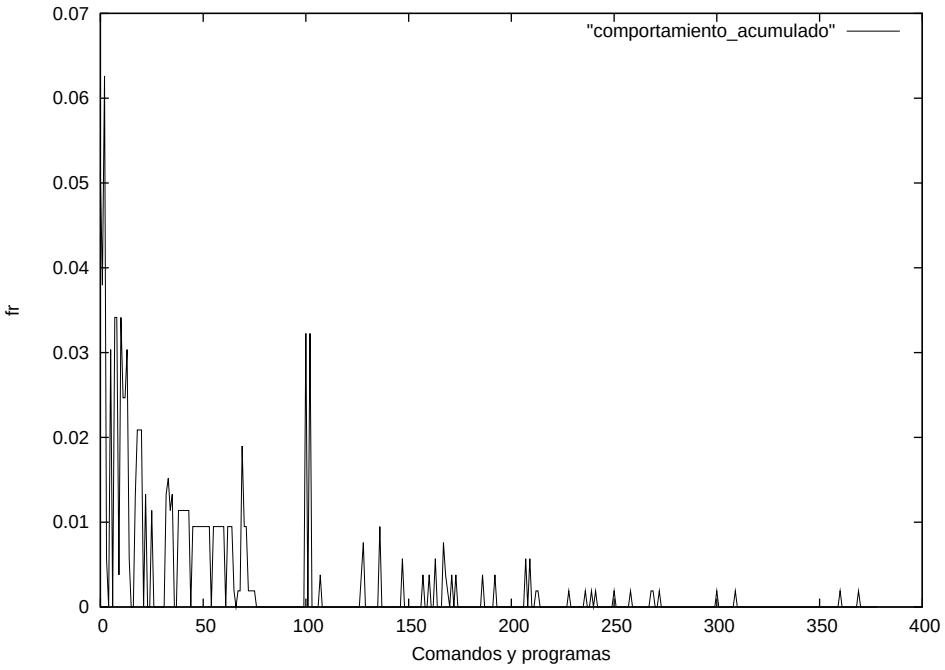


Figura 2.2: Perfil acumulado

En la Figura 2.2, el eje de las abscisas representa al conjunto de programas y comandos (ver Apéndice A) y el eje de las ordenadas es la frecuencia relativa de cada acción. El perfil acumulado es actualizado al día siguiente de la realización de las acciones.

El perfil móvil, es la caracterización dinámica del usuario en un intervalo. Este perfil se representa por las bases de datos para ejecuciones móviles. El perfil móvil tendrá la

suma de las acciones contenidas en los registros de auditoría. Su actualización se realiza en el mismo tiempo en que se actualiza el perfil acumulado y para esto se requiere agregarle las acciones del registro de auditoría del día anterior y restarle las acciones del registro de auditoría del día; ésto se hace para eliminar del perfil móvil las acciones que salen del tiempo comprendido en el intervalo. El perfil móvil se justifica por el hecho de que las acciones de los usuarios son variables, es decir algunas veces trabajan arduamente y otras veces no. La Ecuación 2.1, da la relación para el perfil móvil.

$$Móvil = Móvil + rda\,(i) - rda\,((i+1)\,mod\,(Intervalo)) \tag{2.1}$$

Si, por ejemplo el valor del intervalo corresponde a 15 muestreos, los registros de auditoría son los elementos del conjunto {0,1,2,...,14} y el registro de auditoría empleado para un usuario es 14, entonces la actualización del perfil acumulado se realiza con la adición del registro de auditoría número 14. Para la actualización del perfil móvil se requiere adicionarle también al registro de auditoría número 14 menos el registro de auditoría número 0. Los registros de auditoría, pueden verse como una cola circular, en la cual el último elemento apunta al primero.

2.6. Nuevos Sujetos

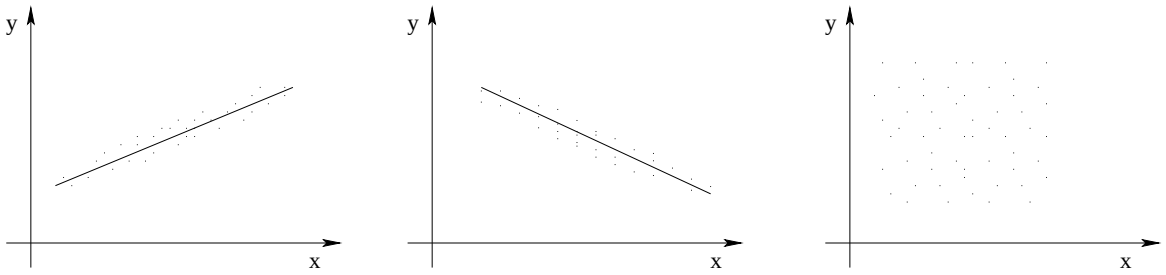
Existe una base de datos que contiene los sujetos (ver Sección 2.3) que han realizado acciones sobre los objetos. Cuando un nuevo sujeto ejecuta una acción sobre un objeto, éste es adicionado a la base de datos de los usuarios y se le crea un directorio y los archivos de los registros de auditoría y perfiles. Cada usuario tiene asignado un directorio en *"/etc/audit/database/event/exec/"* y su nombre estará dado por el valor de *uid*. Este directorio contiene las bases de datos de los eventos de ejecución correspondientes a los registros de auditoría.

2.7. Registros Anómalos

Los registros anómalos se generan cuando exista una diferencia marcada en la comparación de los perfiles acumulado y móvil. Dicha diferencia se obtiene con el coeficiente de correlación lineal (ver Sección 2.7.1). Los registros anómalos son registrados en el archivo */var/log/audit/mensaje.log*.

2.7.1. Coeficiente de Correlación Lineal

El *coeficiente de correlación lineal*, denotado por la letra r permite predecir si entre dos conjuntos de datos o muestras existe una relación o dependencia matemática [Daniel83]. Si los conjuntos de datos no tienen relación $r = 0$, por el contrario si existe relación $r = 1$. El *coeficiente de correlación* está dado por la ecuación 2.2 [Daniel83] y varía entre -1 y +1. El signo "+" representa una correlación lineal positiva y el signo "-" representa una correlación lineal negativa (ver Figura 2.3).



a) Correlación lineal positiva b) Correlación lineal negativa c) Sin correlación

Figura 2.3: Correlación lineal

La Ecuación 2.2 define el coeficiente de correlación lineal.

$$r = \frac{n\sum_{i=0}^{n-1}(x_i-\overline{x})(y_i-\overline{y}) - \left(\sum_{i=0}^{n-1}(x_i-\overline{x})\right)\left(\sum_{i=0}^{n-1}(y_i-\overline{y})\right)}{\sqrt{\left[N\sum_{i=0}^{n-1}(x_i-\overline{x})^2 - \left(\sum_{i=0}^{n-1}(x_i-\overline{x})\right)^2\right]\left[N\sum_{i=0}^{n-1}(y_i-\overline{y})^2 - \left(\sum_{i=0}^{n-1}(y_i-\overline{y})\right)^2\right]}} \tag{2.2}$$

Donde n es el tamaño de las muestras o el número de elementos de los conjuntos de datos. La variable

$$x_i$$

representa los valores de un conjunto de datos y

$$y_i$$

se emplea para los valores del otro conjunto. Para las expresiones contenidas en la Ecuación 2.2, se tiene que:

$$\overline{x} = \frac{\sum_{i=0}^{N-1} x_i}{N} \tag{2.3}$$

$$\overline{y} = \frac{\sum_{i=0}^{N-1} y_i}{N} \tag{2.4}$$

Las Ecuaciones 2.3 y 2.4 representan la media aritmética de los valores de uno y otro conjunto o muestra.

A continuación, se citan tres ejemplos para mostrar las relaciones entre conjuntos de datos, usando el coeficiente de correlación lineal: correlación lineal positiva, negativa y sin correlación.

Tabla 2.1: Datos de Muestras

n	Muestra 1	Muestra 2	Muestra 3	Muestra 4	Muestra 5	Muestra 6
0	11.50	0.60	12.50	1.30	16.25	15.36
1	10.30	0.75	14.60	0.89	12.36	11.25
2	9.20	0.85	18.50	1.23	15.25	14.36
3	8.50	1.20	12.30	2.50	13.23	12.90
4	7.60	1.75	9.80	12.63	8.56	7.40
5	7.00	2.30	12.30	16.25	14.33	13.50
6	6.80	2.50	11.30	2.36	5.26	5.00
7	6.00	3.30	9.50	5.89	10.38	9.98
8	5.50	4.50	10.50	2.58	5.63	6.12
9	4.30	6.30	14.30	15.36	7.45	7.70
10	4.00	7.50	9.50	12.69	9.78	8.36
11	3.90	8.30	3.50	5.96	5.21	5.13
12	3.80	9.20	4.60	1.50	2.36	1.98
13	2.30	9.80	6.80	2.36	12.21	13.10
14	1.80	10.30	10.80	4.50	10.30	10.70
15	0.90	10.50	14.60	6.33	4.27	4.12
16	0.86	10.90	8.63	12.34	6.93	7.14
17	0.68	11.10	12.63	2.33	15.32	13.56
18	0.55	11.50	11.25	1.23	13.68	11.25
19	0.35	12.30	5.25	2.36	9.65	8.76

La Figura 2.4, contiene las muestras 1 y 2. Para este caso el valor de r=-0.89 (ver Figura 2.3 b), lo cual demuestra una correlación lineal negativa y gráficamente se representa por una línea recta con pendiente negativa (ver Figura 2.3).

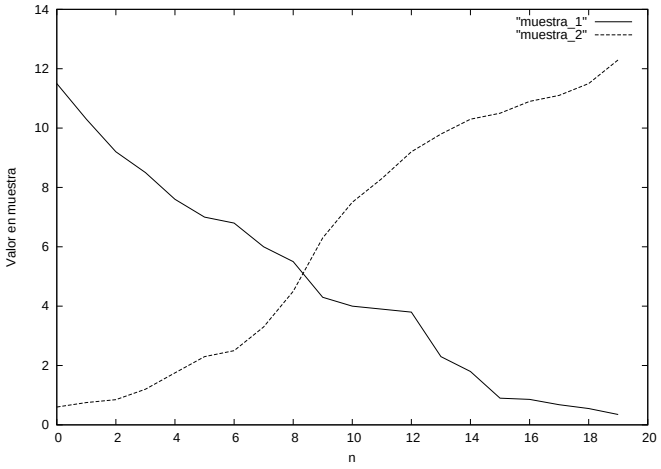


Figura 2.4: Muestra 1 vs 2.

Para la correlación lineal negativa se sabe que los valores en de los conjuntos se contraponen, es decir cuando una tiene un valor máximo la otra tiene un valor mínimo y viceversa.

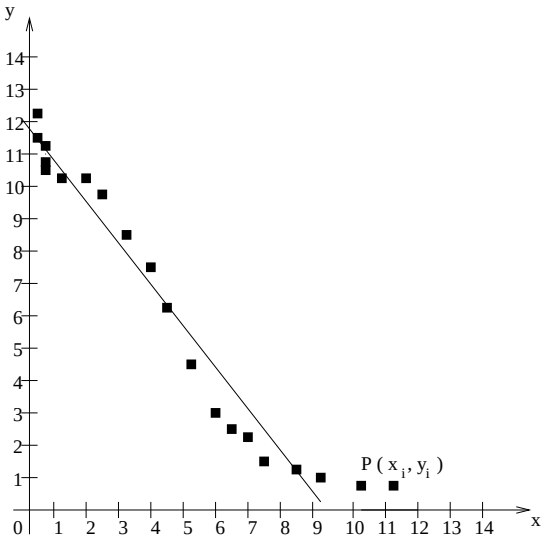


Figura 2.5: Correlación lineal negativa de las muestras 1 y 2.

La Figura 2.6, contiene las muestras 3 y 4. Para este caso el valor de r=0.01 (ver Figura 2.3 c), lo cual demuestra que no existe correlación y gráficamente se representa por

un conjunto de puntos dispersos (ver Figura 2.7).

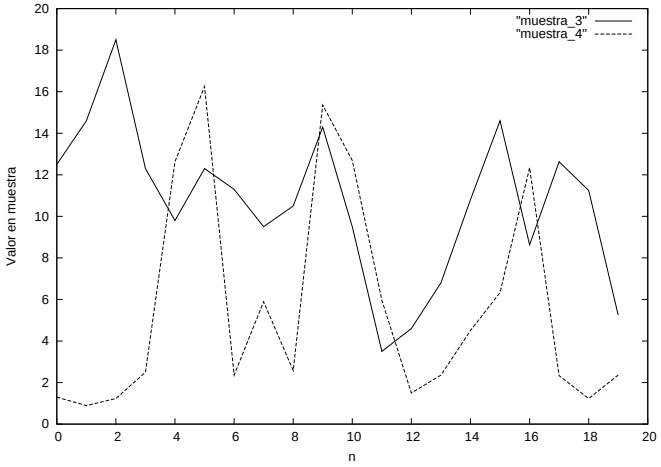


Figura 2.6: Muestra 3 vs 4.

Cuando entre dos conjuntos de datos no existe correlación, significa que estos no se parecen en lo más mínimo.

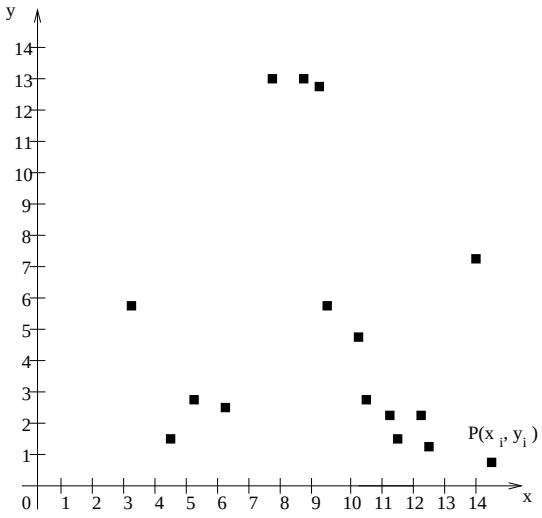


Figura 2.7: Sin correlación lineal entre las muestras 3 y 4.

La Figura 2.8, contiene las muestras 5 y 6. Para este caso el valor de r=0.98 (ver Figura 2.3 a), lo cual demuestra una correlación lineal positiva y gráficamente se representa por una línea recta con pendiente positiva (ver Figura 2.9).

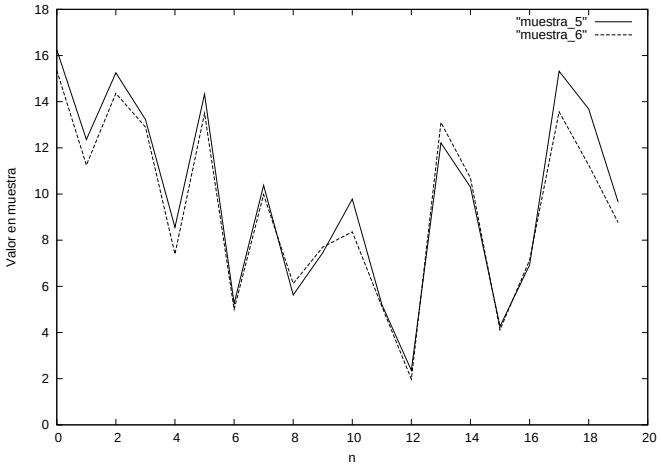


Figura 2.8: Muestra 5 vs 6.

Para la correlación lineal positiva se sabe que ambas conjuntos de datos son muy semejantes.

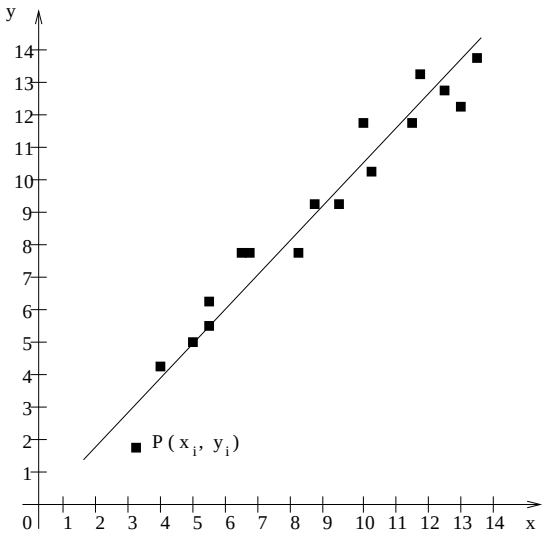


Figura 2.9: Correlación lineal positiva entre las muestras 5 y 6.

Para interpretar el coeficiente de correlación **Colton** [Yañez00], ha dado los siguientes lineamientos generales:

- Valor de r de 0 a 0.25 implica que no existe correlación entre ambas variables.

- Valor de r de 0.25 a 0.50 implica una correlación baja a moderada.

- Valor de r de 0.50 a 0.75 implica correlación moderada a buena.

- Valor de r de 0.75 o mayor, implica una muy buena a excelente correlación.

- Estos rangos de valores se pueden extrapolar a correlaciones negativas también.

2.7.2. Valores del Coeficiente de Correlación

En el presente trabajo se emplea al *coeficiente de correlación* y el criterio de Colton para determinar que tanto se parecen los perfiles acumulado y móvil. Consideramos que el valor del *coeficiente de correlación* debe variar solo entre +0.5 y +1, ya que para este caso que se tiene una correlación moderada a excelente, es decir son muy semejantes. Un valor negativo de r denota diferencia entre los comportamientos, representando acciones contrapuestas, lo que se realiza con mayor frecuencia ahora se realiza con menor y viceversa. Para el valor de cero es peor aun, los perfiles no se parecen nada. De acuerdo al valor del coeficiente de correlación entre los perfiles y móvil, se pueden tomar acciones que pueden ir desde terminar la sesión de usuario para diferencias pequeñas hasta cancelar su cuenta si el cambio del comportamiento fuera drástico. En nuestra implementación sólo se maneja la parte de alarmas, la cual alerta al administrador sobre las anomalías encontradas y la parte de control se considerada como un trabajo futuro.

2.8. Reglas de Actividad

Las Reglas de actividad son acciones que se realizan cuando una condición es cumplida. La Tabla 2.2, muestra las condiciones y reglas de actividad del presente trabajo. La condición 0, es la primer condición que debe cumplirse y una vez cumplida, las acciones de actualizar perfiles se ejecutan para todos los sujetos que realizaron acciones. La detección del comportamiento anómalo se realiza cuando el valor de la variable registro es igual al de intervalo. Finalmente cuando se detectan anomalías se genera un reporte anómalo y se informa al root por correo electrónico.

Tabla 2.2: Condiciones y Reglas de Actividad

No. Condición	Condición	Evaluación	Regla de Actividad
0	¿Nuevo día?	V	-Evaluar condición 1 para cada usuario
1	¿realizo acciones?	V	-Actualizar perfiles -Evaluar condición 2
2	¿registro==Intervalo?	V	-Detectar comportamiento anómalo -Evaluar condición 3
3	¿Comportamiento anómalo?.	V	-Producir reporte anómalo -Informar a root

Para evaluaciones falsas de las condiciones de la Tabla 2.2 no existen reglas de actividad.

2.9. Conclusión

En este capítulo, se describió al modelo empleado en el IDS desarrollado, así como sus partes. Se describieron los perfiles de usuario, los cuales representan la caraterización de los usuarios. Se comentó la medida estadística del coeficiente de correlación, empleado para tener una referencia en la semejanza entre los perfiles acumulado y móvil. También se comentó el criterio de Colton empleado como escala de comparación entre los comportamientos. Este criterio indica los valores del coeficiente de correlación para comportamientos muy parecidos, medio parecidos, diferentes y muy diferentes.

Capítulo 3

Implementación usando SNARE

3.1. Introducción

El equipo de InterSect Alliance [Int04] tiene una amplia experiencia en auditoría y detección de intrusión para un gran número de plataformas: Solaris, Windows NT, Windows 2000, Novell Netware, AIX, MVS (ACF2/RACF). Su experiencia se ha notado en un gran número de agencias de seguridad nacional y defensa, firmas financieras, departamentos de gobierno y proveedores de servicio de Internet.

El equipo de InterSect Alliance y un gran número de usuarios del sistema operativo Linux, consideran que éste carece de algunas características de seguridad, para mantenerlo en uso en un gran número de organizaciones. Una de estas características es la habilidad de detección de intrusión en Host. Desafortunadamente la mayoría de los IDS desarrollados en Linux se basan en la detección de intrusión en Red y muy pocos realizan la detección en Host.

El equipo de InterSect Alliance está tratando de brindar un registro que cumpla con las especificaciones establecidas en el nivel C2 del libro *"Departament of Defense Trusted Computer System Evaluation Criteria"* [ofDefense USA83] (ver Apéndice C), el cual provee los requisitos de seguridad para los sistemas de cómputo. El proyecto es llamado SNARE para Linux [Int04] (de Inglés *System iNtrusion Analysis & Reporting Enviroment*). SNARE funciona bajo los términos de licencia pública de GNU [Stallman01]. SNARE es una herramienta de seguridad que captura y registra la información de los eventos (ejecuciones, conexiones de red, apertura de archivos, etc.)[1] realizados por los usuarios de un sistema

[1]para mayor información ver Tabla 3.1.

operativo.

3.2. Operación de SNARE

Como se mencionó en la Sección 3.1, SNARE tiene la finalidad de capturar y registrar la información de los eventos realizados por los usuarios de un sistema de operativo. La *información* que ofrece SNARE se obtiene a nivel del kernel. Independientemente del evento, SNARE obtiene la *información* del usuario (*uid*[2], *gid*[3]) y de los procesos (*pid*[4], *ppid*[5]) relacionados con dicho evento. SNARE opera con las acciones complementarias de tres aplicaciones:

- El parche del kernel (0.9.3+), o el subsistema del kernel para auditoría (auditmodule.o-0.9.2 y anteriores).

- El demonio *auditd* (procesa y registra la información localmente o la envía a un equipo remoto).

- Una herramienta gráfica para configuración y reporte.

El módulo dinámico del kernel se diseña para interceptar las llamadas al sistema operativo y registrar de manera temporal los detalles asociados a un evento, en el buffer provisto por el dispositivo *"/proc/audit"*. Estos detalles serán accedidos por el demonio *auditd*, para ser procesados y registrados en la bitácora de SNARE, provista por el archivo */var/log/audit/audit.log* (ver Figura 3.1) o los envía a un equipo remoto.

[2]del Inglés *user identifier*.
[3]del Inglés *group identifier*.
[4]del Inglés *process identifier*.
[5]del Inglés *parent process identifier*

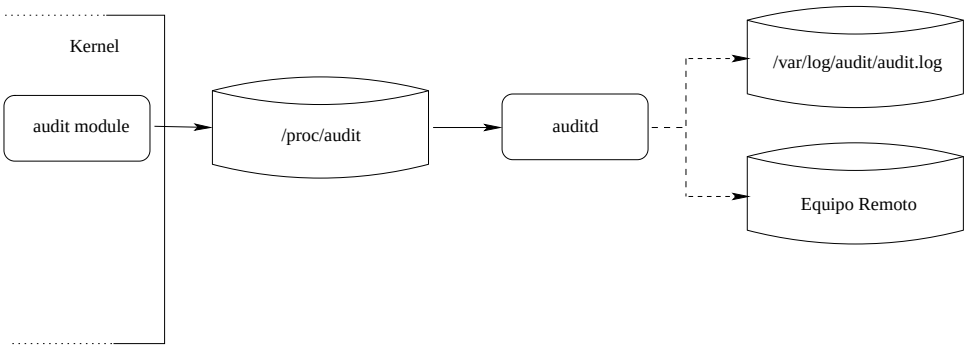


Figura 3.1: Operación de SNARE

Algunas partes de SNARE tienen nombres relacionados con la palabra auditoría por ejemplo, auditd, subsistema del kernel para auditoría. No es que SNARE efectué la auditoría, sino que ofrece los elementos para que la auditoría pueda efectuarse. Ya que el concepto de *auditoría* según [Allenb.Tucker02], se refiere a la revisión de la *información* correspondiente a los eventos efectuados en un sistema, para determinar cuándo y cómo se realizaron o intentaron violaciones de seguridad.

3.2.1. Características del Subsistema del Kernel para Auditoría

Las características del subsistema del kernel para auditoría son:

(a) Los eventos con un path menor o igual a 1024 bytes son procesados. En la versión 0.9 los paths mayores son truncados.

(b) El módulo recibe del demonio auditd, una lista que contiene a las llamadas del sistema para ser capturadas y posteriormente auditadas.

Por otra parte, el parche del kernel modifica las llamadas al sistema y de manera separada interactúa con un proceso que almacena la información en el buffer temporal *"/proc/audit"*. La versión de SNARE utilizada en el presente trabajo utiliza al subsistema del kernel para auditoría.

3.2.2. Reportes para Auditoría

La información capturada y almacenada por el subsistema del kernel para auditoría, es procesada por el demonio *auditd*. El demonio *auditd* convierte la información a texto

llano y la registra localmente en la bitácora provista por el archivo */var/log/audit/audit.log* o la envía a un equipo remoto. El demonio *auditd* utiliza tres diferentes campos separadores usados para facilitar la revisión de la información. El carácter "Tab"[6] separa bloques de datos. El carácter ","[7] separa segmentos de datos en cada bloque de datos. El carácter " "[8] separa datos en cada segmento de datos. A continuación se muestra un ejemplo de esto:

```
localhost.localdomain{0} LinuxAudit{1} objective,critical,Sun Nov 7 15:58:09 2004,
The program /usr/bin/top has been executed by the user invitado{2} event,execve(),Sun
Nov 7 15:58:09 2004{3} user,invitado(500),invitado(500),invitado(500),invitado(500){4}
process,22185,22176,top path,/usr/bin/top{5} arguments,top{6} return,0{7}
sequence,24738{8}
```

El bloque de datos {3} (event,execve(),Sun Nov 7 15:58:09 2004), muestra que se realizó un evento de ejecución y la hora en que se realizó. Este evento fue realizado por el usuario invitado, lo cual se muestra en el bloque de datos {4}. La generación de reportes se deshabilita en el presente trabajo, ya que en lugar de registrar los eventos en la bitácora */var/log/audit/audit.log*, estos son procesados para ser registrados en una base de datos correspondiente al usuario que lo efectuó (Ver Sección 3.4).

3.3. Configuración de SNARE

La configuración de SNARE se registra en el archivo */etc/audit/audit.conf*. Este archivo contiene los detalles necesarios para un funcionamiento correcto de SNARE. Es decir, las llamadas al sistema que serán requeridas por el subsistema del kernel para auditoría y el destino de la información procesada por el demonio *auditd*. Los parámetros configurables de SNARE son:

- Configuración por objetivos (ver Sección 3.3.1).

- El destino de los eventos auditados (archivo o red).

La configuración manual de audit.conf es posible, pero se debe tener cuidado especial en adicionar de manera correcta los datos necesarios. Un error en la especificación de la configuración posiblemente no afecte el funcionamiento de SNARE, pero pudiera ser

[6]Carácter ASCII con valor Hexadecimal 1D.
[7]Carácter ASCII con valor Hexadecimal 2C.
[8]Carácter ASCII con valor Hexadecimal 20.

que las llamadas al sistema requeridas no sean capturadas. InterSect Alliance recomienda utilizar la herramienta gráfica de SNARE para reporte y configuración, ya que ésta es la forma más efectiva y simple para configurar a SNARE. En el presente trabajo, no utilizó a la herramienta gráfica de SNARE, dado que la configuración para SNARE utilizada en éste no requiere de un gran número de objetivos.

3.3.1. Configuración por Objetivos

Un forma más refinada de auditoría se logra definiendo objetivos. Los objetivos limitan la captura de las llamadas al sistema. Se puede especificar cualquier número de objetivos. A continuación se agrupan las llamadas al sistema, que pueden ser requeridas para auditoría:

- Apertura o creación de archivos o directorios.
- Apertura de archivo para lectura.
- Escritura o creación de un archivo o directorio.
- Remover un archivo o directorio.
- Iniciar o detener ejecuciones de programas.
- Modificar el sistema, archivos o atributos de directorios.
- Cambio de usuario o identidad de grupo.
- Apertura de conexiones de red.
- Aceptar conexiones de red.
- Eventos administrativos.

En la Tabla 3.1 se muestra la lista detallada de las llamadas al sistema que son modificadas para que puedan ser capturadas y auditadas.

Tabla 3.1: Llamadas al Sistema Requeridas para Auditoría

Llamada al Sistema	Descripción
open	abrir y posiblemente crear un archivo o dispositivo
creat	abrir y posiblemente crear un archivo o dispositivo
execve	ejecuta un programa
exit	terminación del proceso actual
mkdir	creación de directorio
unlik	borrar un nombre y la referencia a un archivo
mknod	creación de un directorio o un archivo especial o ordinario
rmdir	borrar un directorio
chown	cambio de propietario para un archivo
lchown	cambio de propietario para un archivo
chown32	cambio de propietario para un archivo
lchown32	cambio de propietario para un archivo
chmod	cambio de los permisos de un archivo
symlink	crear un nuevo nombre para una archivo
link	crear un nuevo nombre para una archivo
rename	cambiar el nombre o la localización de un archivo
reboot	reiniciar o habilitar/deshabilitar Ctrl-Alt-Del
truncate	truncate un archivo a un tamaño especifico
chroot	cambio al directorio de root
setuid	cambio de identificación de usuario
setreuid	cambio de identificación real o efectiva de usuario
setresuid	cambio de identificación real, efectiva y registro de ID usuario
setuid32	cambio de identificación de usuario
setreuid32	cambio de identificación real o efectiva de usuario
setresuid32	cambio de identificación real,efectiva y registro de ID usuario
setgid	cambio de identificación de grupo
setregid	cambio de identificación real o efectiva de grupo
setresgid	cambio de identificación real, efectiva y registro de ID grupo
setgid32	cambio de identificación de grupo
setregid32	cambio de identificación real o efectiva de grupo
setresgid32	cambio de identificación real, efectiva y registro de ID grupo
truncate64	truncar un archivo a un tamaño especifico
socketcall	funciones de red, incluido conexión y aceptación
create_module	cargar un modulo del kernel
delete_module	remover un modulo del kernel

En el presente trabajo (ver Sección 1.5), se construyen perfiles de usuario en base a ejecuciones de comandos y programas. Por esta razón, la configuración requerida por SNARE se resume sólo a especificar la captura de la llamada al sistema de "*execve*". A

continuación se muestra la configuración empleada:

```
[HostID]
name=

[Output]
file=/var/log/audit/audit.log

[Remote]
allow=1
listen_port=6161

[Objectives]
criticality=4 event=execve return=* user!=root match=.*
criticality=4 event=connect return=* user!=root match=.*
Criticality=4 event=accept  return=* user!=root match=.*
```

Como se muestra en la configuración, sólo se especifican tres eventos como objetivos: *execve*, *connect* y *accept*. Para los eventos de red (*connect* y *accept*), se realizaron de manera preliminar ciertas implementaciones en el código de *auditd*. De igual manera se realizaron varios experimentos, relacionados con estos eventos. El objetivo principal del presente trabajo, planteaba manejar los eventos *execve*, realizados por los usuarios normales, excluyendo al usuario root y a los usuarios estándares del sistema. Posteriormete se consideró adecuado incluir también en la prueba al superusuario (ver Sección 2.3).

3.4. Adaptación

Para la implementación del presente trabajo fue necesario realizar adaptaciones a SNARE, por las razones que enseguida se citan:

- Procesar la información en tiempo real.

- No utilizar un elemento externo a SNARE, que procese la información. Cubriendo con esto las características 1 y 5, descritas en la Sección 1.2.1.

- Registrar en bases de datos la información de los eventos de ejecución realizados por los usuarios.

- Detectar anomalías en los perfiles de usuarios.

Las adaptaciones hechas a SNARE se realizaron en el demonio *auditd*. La Figura 3.2, muestra como se reemplazo el destino para la información capturada por SNARE. Nuestro esquema emplea bases de datos como destino final de la información.

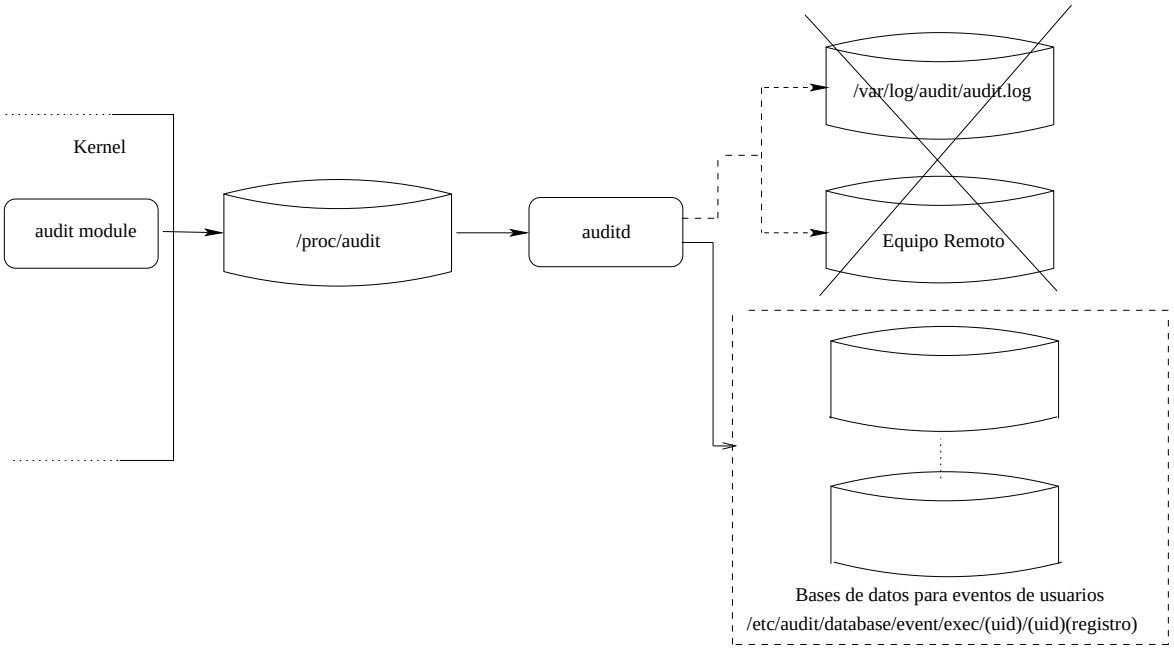


Figura 3.2: Adaptación de SNARE

Cada usuario tiene un directorio en *"/etc/audit/database/event/exec/"* y para su nombre se emplea el valor de *uid*, tal como se muestra en la figura 3.2. En este directorio se tiene la base de datos para los eventos de ejecución.

Los nombres para los archivos de las bases de datos se forman de la combinación del valor de *uid* y el valor de un contador llamado *registro* (ver Estructuras Adicionadas en 3.4.2). La implementación de las bases de datos se realizó con la librería NDBM[9], descritas en la Sección 3.4.1.

3.4.1. Librería NDBM

La librería DBM es reemplazada en los sistemas operativos *nix[10], por la librería NDBM [AIX01] [IEEE04]. NDBM permite crear, almacenar, recuperar y modificar una

[9] del Inglés *New Database Manager*.
[10] Todos los sistemas operativos estándares basados en Unix.

base de datos. Esta librería, proporciona la implementación de las bases de datos utilizadas en la adaptación hecha a SNARE. El no utilizar esta librería implicaba contar con una implementación externa para el manejo de las bases de datos.

Funciones de la Librería NDBM

A continuación se muestran a las funciones de la librería ndbm.h que permiten crear, almacenar, recuperar y modificar una base de datos:

```
int dbm_clearerr(DBM *db)
void dbm_close(DBM *db)
int dbm_delete(DBM *db, datum key)
int dbm_error(DBM *db)
datum dbm_fetch(DBM *db, datum key)
datum dbm_firstkey(DBM *db)
datum dbm_nextkey(DBM *db)
DBM  *dbm_open(const  char  *file,  int  open_flags, mode_t file_mode)
int  dbm_store(DBM  *db,  datum  key,  datum  content, int store_mode)
```

Bases de Datos ndbm

El elemento básico de las bases de datos **ndbm** es un bloque de datos con una clave asociada. En una base de datos **ndbm** debe existir una clave única para cada bloque de datos almacenado. No existe restricción sobre el tamaño de las claves.

Para manipular estos bloques como datos, la librería ndbm define una estructura llamada **datum**. Un **datum** consiste de dos miembros, *dptr* y *dsize*. El miembro *dptr* es un apuntador a un bloque de tamaño *dsize* en bytes. Arbitrariamente los datos binarios, así como las cadenas de caracteres, pueden ser almacenados en el bloque al que apunta *dptr*.

```
typedef struct {
        char *dptr;
        int dsize;
    } datum;
```

Las bases de datos son almacenadas en dos archivos. Un archivo con extensión **.pag**, contiene un directorio de mapa de bits de las claves. El segundo archivo con extensión **.dir**, contiene los datos.

Descripción de las Funciones de la Librería NDBM

La función *dbm_open()* intenta abrir una base de datos. El argumento *file* define la referencia absoluta a la base de datos. La función abre dos archivos llamados *file*.**dir** y *file*.**pag**. El argumento *open_flags*, tiene el mismo significado que el argumento *flags* para la función *open ()*, excepto para los accesos de solo escritura. Para el argumento *file_mode* se tiene el mismo significado como el tercer argumento de *open()*.

La función *dbm_close()* cierra una base de datos. El argumento de *db* es un puntero a una estructura **dbm**, previamente regresada de la llamada a la función *dbm_open()*.

La función *dbm_fetch()* lee un registro de la base de datos. El argumento *key* es una estructura **datum** inicializada como la clave.

La función *dbm_store()* intenta escribir un registro en la base de datos. El argumento *content* es una estructura **datum** inicializada como el dato para el registro de la base de datos.

La función *dbm_firstkey()* regresa la primera clave de la base de datos.

La función *dbm_nextkey()* regresa la siguiente clave de la base de datos.

Las funciones utilizadas para el manejo de las bases datos fueron: *dbm_close()*, *dbm_fetch()*, *dbm_firstkey()*, *dbm_nextkey()*, *dbm_open()* y *dbm_store()*.

3.4.2. Adaptación de auditd.h

El código de auditd.h contiene la declaración de constantes, estructuras y funciones utilizadas por el demonio *auditd*. Se tuvo la necesidad de adicionar ciertas constantes y estructuras, que a continuación se citan.

Constantes adicionadas

La constante DATABASE_EXEC representa la ruta para las bases de datos de los eventos de ejecución, para los usuarios del sistema. La constante DATABASE_USERS representa la ruta de la base de datos de los usuarios del sistema. Las constantes MENSAJE y MENSAJE_LOG representan la ruta para la bitácora diaria, de los valores del coeficiente de correlación obtenidos y la bitácora acumulada de éstos valores, respectivamente.

```
#define DATABASE_EXEC "/etc/audit/database/event/exec/"
#define DATABASE_USERS "/etc/audit/database/users/users"
#define MENSAJE "/var/log/audit/mensaje"
#define MENSAJE_LOG "/var/log/audit/mensaje.log"
```

Estructuras adicionadas

La estructura *user*, es utilizada en la base de datos los usuarios del sistema. Esta estructura cuenta con una cadena para almacenar el nombre del usuario del sistema (*nameuser*) y un valor entero para el identificador del usuario (*uid*). El elemento path, almacena la ruta para la base de datos de ejecución de este usuario. Esta ruta se obtiene de la concatenación de la constante DATABASE_EXEC y el elemento *nameuser*. Los elementos de tipo entero *mday*, *mon* y *year*, almacenan el día del mes, mes y año, respectivamente, del último evento realizado por el usuario. Esto, con la finalidad de saber si un nuevo evento requiere ser guardado en una nueva base de datos. El elemento entero *registro*, almacena un valor asociado con la fecha. La ruta adecuada de la base de datos para las ejecuciones de los usuarios, se obtiene de la concatenación de los elementos *path*, *uid* y *registro*. La llave utilizada para estas bases de datos, está formada por el valor entero de *uid*.

```
struct user{
 char nameuser[MAX_PATH];
 int uid;
 char path[MAX_PATH];
 int mday;
 int mon;
 int year;
 int registro;
};
```

La estructura *exec*, es utilizada en las bases de datos de para los eventos de ejecución realizados por los usuarios del sistema. Los elementos *user* y *uid*, tienen el mismo significado que los utilizados en la estructura *user*. El elemento *processname*, almacena el nombre del proceso que se ejecutó y *t_path* la ruta de éste mismo. Los elementos *pid* y *ppid*, almacenan los valores tipo *pid_t*, para el proceso actual que se está ejecutando y el proceso padre, respectivamente. El elemento entero *returncode*, almacena el valor que retorna el kernel, al haber ejecutado algún proceso. Finalmente el elemento *frequency*, almacena el número de veces que el proceso se ejecuta en un día. La llave utilizada para estas bases de datos, está formada por el contenido del elemento *t_path*.

```
struct exec{
 char user[MAX_PATH];
 int uid;
 char processname[MAXCOMMAND];
 char t_path[MAX_PATH];
```

```
 pid_t pid;
 pid_t ppid;
 int returncode;
 double frequency;
};
```

La estructura *nt*, fue utilizada en varios experimentos previos a la implementación final. Los elementos *uid*, *processname*, *t_path*, *pid*, *ppid*, *returncode* y *frequency*, poseen el mismo significado que los elementos de la estructura *exec*. Los elementos *dst_ip[40]*, *dst_port* y *protocol*, representan a la dirección ip destino, puerto destino y el protocolo de comunicación utilizado, respectivamente. Esta estructura se utilizó para contar con las preferencias de red de los usuarios del sistema.

```
struct nt{
 char user[MAX_PATH];
 int uid;
 char dst_ip[40];
 int dst_port;
 int protocol;
 char processname[MAXCOMMAND];
 char t_path[MAX_PATH]
 pid_t pid;
 pid_t ppid;
 int returncode;
 double frequency;
};
```

3.4.3. Adaptación de auditd.c

La primera adaptación se realizó en la función *process_ex_class()*, contenida en el código de auidtd.c. El demonio *auditd* se encuentra permanentemente verificando si un evento nuevo se ha generado, esto lo realiza con la función safe_read(). Por nuestra parte, adicionamos la ejecución de la función *realiza_acumulado()*, descrita posteriormente al igual que las funciones *add_user()* y *registra_exec()*.

Inicio — Función main() de auditd.c — Inicio

```
int main(int argc, char *argv[])
{
  ...
  ...
        while (!done) {
                done = safe_read(device_fd, &header, sizeof(header_token));
```

```
                if (!done) {
                        switch (header.event_class) {
                        case AUDIT_CLASS_IO:
                                process_io_class(&header);
                                break;
                        case AUDIT_CLASS_EXEC:
                                process_ex_class(&header);
                                break;
                        case AUDIT_CLASS_PC:
                                process_pc_class(&header);
                                break;
                        case AUDIT_CLASS_CH:
                                process_ch_class(&header);
                                break;
                        case AUDIT_CLASS_CP:
                                process_cp_class(&header);
                                break;
                        case AUDIT_CLASS_SU:
                                process_su_class(&header);
                                break;
                        case AUDIT_CLASS_NET:
                                process_nt_class(&header);
                                break;
                        case AUDIT_CLASS_AD:
                                process_ad_class(&header);
                                break;
                        default:
                                if(header.event_class > 0) {
                                        printf("Auditd: Problem - Data read by daemon is not a
    recognised audit class - %d\n",header.event_class);
                                }
                                if (header.event_size > MAX_AUDITREC) {
                                        header.event_size = MAX_AUDITREC;
                                }
                        }
                }
                if (signal_flags) {
                        sync_signal_handler();
                }
         realiza_acumulado();
        }
        abort_audit("Audit Daemon has terminated unexpectedly", 1);
        return(1);
}
```

Fin Función main() de auditd.c Fin

La función *process_ex_class()*, se modificó para obtener la información del evento de ejecución y ser canalizada a la base de datos válida del usuario real. Un usuario real se determina con su identificador de usuario *uid*, si este es mayor de 499 [Red03b] y menor de 65534 (ver Tabla 3.2), entonces es un usuario del sistema y se procede a ejecutar las funciones *add_user()*, *registra_exec()*. Por recomendación se incluye también al superusuario (root) como usuario a ser registrado cuyo *uid*= 0.

De manera adicional en el Apéndice B se muestra la tabla para los grupos estándar del sistema.

Inicio Función process_ex_class() de auditd.c Inicio

```
int process_ex_class(header_token * header)
{
        ex_class ex_event;
        Node *objectiveNode = NULL;
        char eventdata[MAX_AUDITREC];
        char tempdata[MAX_AUDITREC];
        char objectivetext[MAX_AUDITREC];
        char pathstore[PATH_MAX];
        char userstore[MAX_USERNAME];
        char timestr[TIMEBUF] = "";
        int ObjectiveReturn = RETURNCODE_SUCCESS;
        int do_objectives = 0;
        struct user usuario_nuevo;
        ex_event.t_header = *header;

        if (safe_read
            (device_fd, (void *) &ex_event.t_header + sizeof(header_token),
             ex_event.t_header.event_size)){
                return 0;
        }

        if(ex_event.t_header.user_id > 499 && ex_event.t_header.user_id < 65534
|| ex_event.t_header.user_id == 0){
                add_user(ex_event.t_header.user_id,&usuario_nuevo);
                registra_exec(usuario_nuevo,ex_event);
        }
        return (0);
}
```

Fin Función process_ex_class() de auditd.c Fin

Tabla 3.2: Usuarios Estándar del Sistema

Usuario	UID	GID	Directorio personal	Shell
root	0	0	/root	/bin/bash
bin	1	1	/bin	/sbin/nologin
daemon	2	2	/sbin	/sbin/nologin
adm	3	4	/var/adm	/sbin/nologin
lp	4	7	/var/spool/lpd	/sbin/nologin
sync	5	0	/sbin	/bin/sync
shutdown	6	0	/sbin	/sbin/shutdown
halt	7	0	/sbin	/sbin/halt
mail	8	12	/var/spool/mail	/sbin/nologin
news	9	13	/var/spool/news	
uucp	10	14	/var/spool/uucp	/sbin/nologin
operator	11	0	/root	/sbin/nologin
games	12	100	/usr/games	/sbin/nologin
gopher	13	30	/usr/lib/gopher-data	/sbin/nologin
ftp	14	50	/var/ftp	/sbin/nologin
nobody	99	99	/	/sbin/nologin
rpm	37	37	/var/lib/rpm	/bin/bash
vcsa	69	69	/dev	/sbin/nologin
ntp	38	38	/etc/ntp	/sbin/nologin
canna	39	39	/var/lib/canna	/sbin/nologin
nscd	28	28	/	/bin/false
rpc	32	32	/	/sbin/nologin
postfix	89	89	/var/spool/postfix	/bin/true
named	25	25	/var/named	/bin/false
amanda	33	6	/var/lib/amanda	/bin/bash
postgres	26	26	/var/lib/pgsql	/bin/bash
sshd	74	74	/var/empty/sshd	/sbin/nologin
rpcuser	29	29	/var/lib/nfs	/sbin/nologin
nsfnobody	65534	65534	/var/lib/nfs	/sbin/nologin
pvm	24	24	/usr/share/pvm3	/bin/bash
apache	48	48	/var/www	/bin/false
xfs	43	43	/etc/X11/fs	/sbin/nologin
desktop	80	80	/var/lib/menu/kde	/sbin/nologin
gdm	42	42	/var/gdm	/sbin/nologin
mysql	27	27	/var/lib/mysql	/bin/bash
webalizer	67	67	/var/www/html/usage	/sbin/nologin
mailman	41	41	/var/mailman	/bin/false
mailnull	47	47	/var/spool/mqueue	/sbin/nologin
smmsp	51	51	/var/spool/mqueue	/sbin/nologin

Tabla 3.3: Continuación de Tabla 3.2, Usuarios Estándar del Sistema

Usuario	UID	GID	Directorio personal	Shell
squid	23	23	/var/spool/squid	/dev/null
ldap	55	55	/va/lib/ldap	/bin/false
netdump	34	34	/var/crash	/bin/bash
pcap	77	77	/var/arpwatch	/sbin/nologin
ident	98	98	/	/sbin/nologin
privoxy	75	75	/etc/privoxy	
radvd	12	100	/	/bin/false
fax	78	78	/var/spool/fax	/sbin/nologin
wnn	49	49	/var/lib/wnn	/bin/bash

La función *add_user()*, se encarga de agregar usuarios reales a la base de datos de usuarios e inicializar los directorios y archivos, así como también cambiar el valor de la variable registro, contenida en *struct user* (ver Sección 3.4.2).

Inicio Función add_user() de auditd.c Inicio

```
int add_user(int uid,struct user * usuario_nuevo)
{
 struct passwd *infouser;
 char key[MAX_PATH];
 char path[100];
 datum key_datum;
 datum data_datum;
 DBM *dbm_ptr;
 DBM *temp;
 time_t te;
 struct tm *tes;
 te = time(NULL);
 tes = localtime(&te);

 infouser = getpwuid(uid);
 dbm_ptr = dbm_open(DATABASE_USERS,O_RDWR | O_CREAT,0666);
        if(!dbm_ptr)
                return FAILED;

 sprintf(key,"%i",uid);
 key_datum.dptr = key;
 key_datum.dsize = strlen(key);
 data_datum = dbm_fetch(dbm_ptr,key_datum);
```

```
 if(!data_datum.dptr){
        sprintf(path,"%s%i/",DATABASE_EXEC,uid);
        strcpy(usuario_nuevo->nameuser,infouser->pw_name);
        strcpy(usuario_nuevo->path,path);
        usuario_nuevo->uid=uid;
        usuario_nuevo->registro=0;
        usuario_nuevo->mday=tes->tm_mday;
        usuario_nuevo->mon=tes->tm_mon;
        usuario_nuevo->year=tes->tm_year;
        mkdir(path,0666);
        sprintf(path,"%s%i/acumulado_%i",DATABASE_EXEC,uid,uid);
        temp = dbm_open(path,O_RDWR | O_CREAT,0666);
        dbm_close(temp);
        sprintf(path,"%s%i/movil_%i",DATABASE_EXEC,uid,uid);
        temp = dbm_open(path,O_RDWR | O_CREAT,0666);
        dbm_close(temp);
 }else{
    memcpy(usuario_nuevo,data_datum.dptr,data_datum.dsize);
      if((tes->tm_mday != usuario_nuevo->mday) ||
(tes->tm_mon != usuario_nuevo->mon)||(tes->tm_year != usuario_nuevo->year))
        {
         usuario_nuevo->registro+=1;
                usuario_nuevo->mday=tes->tm_mday;
                suario_nuevo->mon=tes->tm_mon;
                usuario_nuevo->year=tes->tm_year;
                if(usuario_nuevo->registro > intervalo)
                         {
                          usuario_nuevo->registro=0;
                         }
         }
 }
 data_datum.dptr = (void *)usuario_nuevo;
 data_datum.dsize = sizeof(struct user);
 dbm_store(dbm_ptr, key_datum, data_datum, DBM_REPLACE);
 dbm_close(dbm_ptr);
 return 0;
}
```

Fin ——— Función add_user() de auditd.c ——— Fin

. . . . . .

La función *registra_exec()*, se encarga de agregar la información del evento generado en la tabla correspondiente del usuario o aumentar el valor de la frecuencia, si el registro para dicho evento existe.

. . . . . .

Inicio Función registra_exec() de auditd.c Inicio

```
int registra_exec(struct user usuario_nuevo,ex_class ex_event)
{
 struct exec temp;
 char key[MAX_PATH];
 char path[100];
 datum key_datum;
 datum data_datum;
 DBM *dbm_ptr;

 sprintf(path,"%s%i%i",usuario_nuevo.path,ex_event.t_header.user_id,usuario_nuevo.registro);

 dbm_ptr = dbm_open(path,O_RDWR | O_CREAT,0666);
        if(!dbm_ptr)
                return FAILED;

 memset(&temp,'\0',sizeof(struct exec));
 //generacion de llave para dato
 sprintf(key,"%s",ex_event.t_path.path);
 key_datum.dptr = key;
 key_datum.dsize = strlen(key);
 data_datum = dbm_fetch(dbm_ptr,key_datum);
 if(data_datum.dptr){
        memcpy(&temp,data_datum.dptr,data_datum.dsize);
        temp.frequency++;
        temp.pid = ex_event.t_header.pid;
        temp.ppid = ex_event.t_header.ppid;
        temp.returncode = ex_event.t_header.returncode;
 }else{
        strcpy(temp.user,usuario_nuevo.nameuser);
        temp.uid = ex_event.t_header.user_id;
        strcpy(temp.processname,ex_event.t_header.processname);
        strcpy(temp.t_path,ex_event.t_path.path);
        temp.pid = ex_event.t_header.pid;
        temp.ppid = ex_event.t_header.ppid;
        temp.returncode = ex_event.t_header.returncode;
        temp.frequency=1.00;
        }
 data_datum.dptr = (void *)&temp;
 data_datum.dsize = sizeof(struct exec);
 dbm_store(dbm_ptr, key_datum, data_datum, DBM_REPLACE);
 dbm_close(dbm_ptr);
 acumulado_global(ex_event);
 return 0;
}
```

Fin ——— Función registra_exec() de auditd.c ——— Fin

La función *realiza_acumulado()*, se adicionó para obtener el coeficiente de correlación de los usuarios que realizaron actividades de ejecución. Las actualizaciones y cálculo se realizan a las 12:00 a.m. de cada día. De manera adicional se envía un mensaje al administrador de los valores obtenidos del coeficiente de correlación.

Inicia ——— Función realiza_acumulado() de auditd.c ——— Inicia

```
int realiza_acumulado(void)
{
 tiempo = time(NULL);
 tiempo_actual=localtime(&tiempo);

 if((ban==0) && (tiempo_actual->tm_hour == 0) && (tiempo_actual->tm_min == 0))
  {
   obten_acumulados_moviles();
   ban=1;
  }

 if((ban==1) && (tiempo_actual->tm_hour == 0) && (tiempo_actual->tm_min == 1))
  {
   ban=0;
   copia_estructura_tm(tiempo_actual,&tiempo_inicial);
  }
 return 0;
}
```

Fin ——— Función realiza_acumulado() de auditd.c ——— Fin

3.5. Conclusión

En este capítulo, se dio una introducción a la herramienta de SNARE, cubriendo su operación y configuración. Por otro lado, se mencionaron las razones de su adaptación a SNARE. Se describieron las partes involucradas en la adaptación (funciones adicionadas). Así como también, los elementos utilizados en la adaptación (por ejemplo, la librería NDBM, usuarios estándar del sistema, etc.).

Capítulo 4

Resultados

4.1. Introducción

El producto final del presente trabajo fué desarrollado en Lenguaje C y las pruebas se realizarón en la Sección de Investigación y Desarrollo de la Facultad de Ingeniería Eléctrica de la UMSNH. Los comportamientos registrados en los experimentos con usuarios corresponden a usuarios normales (ver Sección 4.3). Para los experimentos con supuerusuario (ver Sección 4.3) el comportamiento corresponde al sistema operativo y administrador.

4.2. Experimentos con Usuarios

Inicialmente este trabajo considero registrar el comportamiento de usuarios normales, por ser necesario contar con una referencia que indique si las cuentas de usuario han sido comprometidas. Los datos obtenidos corresponden a ejecuciones de 24 usuarios durante un mes.

4.2.1. Datos Obtenidos

Las experimentos con usuarios produjeron 379 comandos y programas ejecutados durante éstas, la Tabla 4.1, muestra la distribución de frecuencias para estas ejecuciones. El primer renglón indica que de estos 379 comandos y programas, 209 se ejecutaron entre 2 y 8 veces. El penúltimo renglón muestra que 28 comandos y programas se ejecutaron entre 100 y 600 veces. Esto nos indica que la gran mayoría de los participantes ejecutan de manera

muy constante 28 comandos y programas. La lista de los 379 comandos y programas se presenta en el Apéndice A.

Tabla 4.1: Distribución de Frecuencias de Ejecuciones

Número de ejecuciones	Número de comandos
2-8	209
10-18	39
20-49	47
50-100	56
100-600	28
	Total 379

En la Figura 4.1 se muestra el acumulado de las ejecuciones de los 379 comandos y programas, para la cual el eje de las abscisas representa los comandos y programas ejecutados y ordenados descendientemente por número de ejecuciones acumuladas. El eje de las ordenadas representa la frecuencia relativa de las ejecuciones. En total se registraron 12887 ejecuciones.

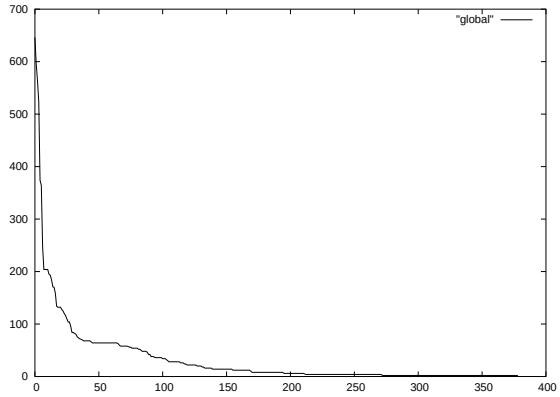


Figura 4.1: Acumulado de ejecuciones realizadas

Las Figuras 4.2 y 4.3, muestran los histogramas de 4 usuarios participantes. El Apéndice D tiene los 24 histogramas de los usuarios participantes.

La Figura 4.2, muestra los histogramas de los participantes 0 y 6, estos usuarios reflejan un grado alto de conocimiento. Debido a que manejan un buen número de comandos y programas.

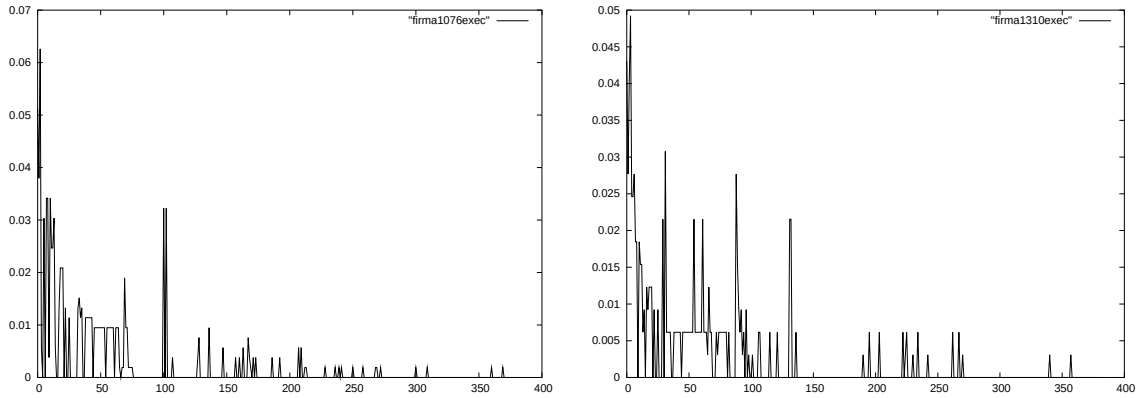


Figura 4.2: Histogramas de ejecuciones de los usuarios 0 y 6

El histograma del usuario 17 mostrado en la Figura 4.3, denota un grado menor de conocimientos, mientras que para el usuario 18 se observa un grado mayor.

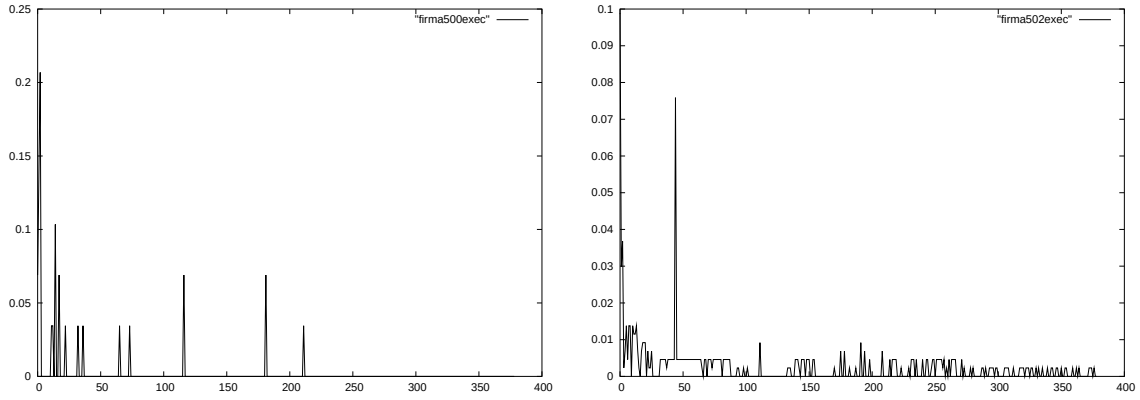


Figura 4.3: Histogramas de ejecuciones de los usuarios 17 y 18

4.2.2. Resultados Obtenidos

La Figura 4.4 muestra el perfil acumulado de un usuario x y su perfil móvil se muestra en la Figura 4.5.

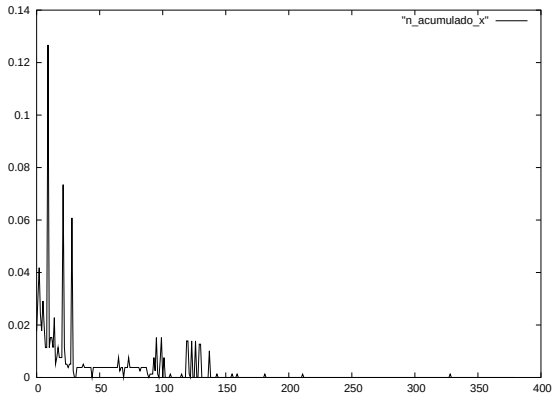


Figura 4.4: Perfil acumulado del usuario x

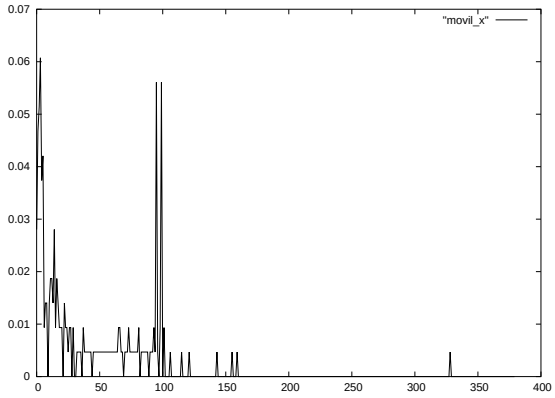


Figura 4.5: Perfil móvil del usuario x

El valor del coeficiente de correlación r entre los perfiles acumulado de la Figura 4.4 y el comportamiento móvil de la Figura 4.5 es 0.19497040 (ver Figura 4.6). Lo cual demuestra una anomalía. Dada esta condición, es necesario alertar al administrador (ver Sección 2.7.2). Dicha anomalía puede deberse a nuevos conocimientos adquiridos por el usuario.

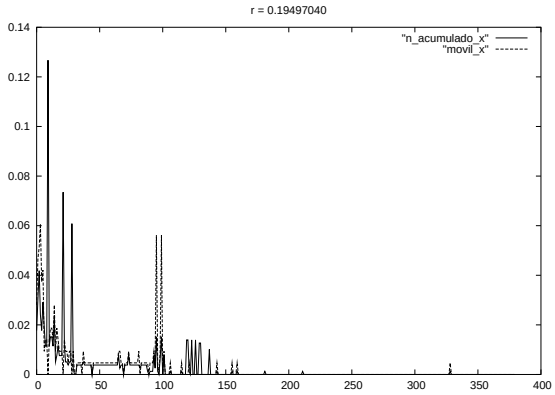


Figura 4.6: Perfiles acumulado y móvil del usuario x.

El valor del coeficiente de correlación entre los perfiles acumulado_y y el móvil_y es 0.96444667 (Figura 4.7). Para este caso existe un comportamiento muy similar entre los perfiles acumulado y móvil. El usuario y muestra una pequeña variación en su comportamiento.

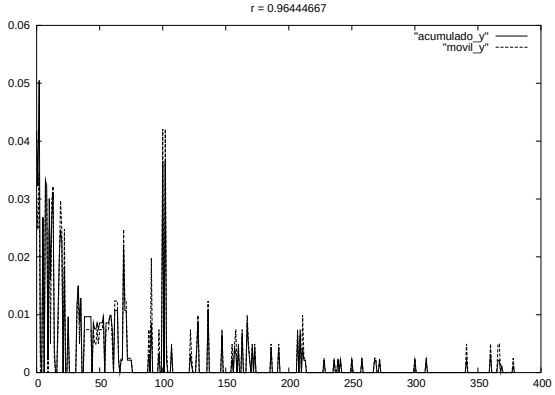


Figura 4.7: Perfiles acumulado y móvil del usuario y.

Los usuarios participantes en la prueba son susceptibles a mostrar cambios en el comportamiento, esto es debido a la naturaleza de sus actividades (estudiantes). Las variaciones del comportamiento pueden denotar una intrusión. Aunque como se ha mencionado, las variaciones del comportamiento son muy frecuentes en usuarios que realizan actividades variantes, a diferencia de usuarios con actividades invariantes. De cualquier forma una intrusión o no, el hecho es que existen cambios en el comportamiento (anomalías), para lo

cual se puede generar una investigación para conocer el porqué de los cambios.

4.3. Experimentos con Superusuario

El experimento con el superusuario produjo 420 comandos y programas. Como se observa el número de comandos y programas del superusuario es mayor que el correspondiente a los 24 usuarios participantes. La Figura 4.8, muestra los perfiles acumulado y móvil del usuario root. Como se mencionó en la Sección 2.3, el comportamiento del root caracteriza al comportamiento del sistema operativo y el administrador. Ya que ciertos programas requieren la identidad del root.

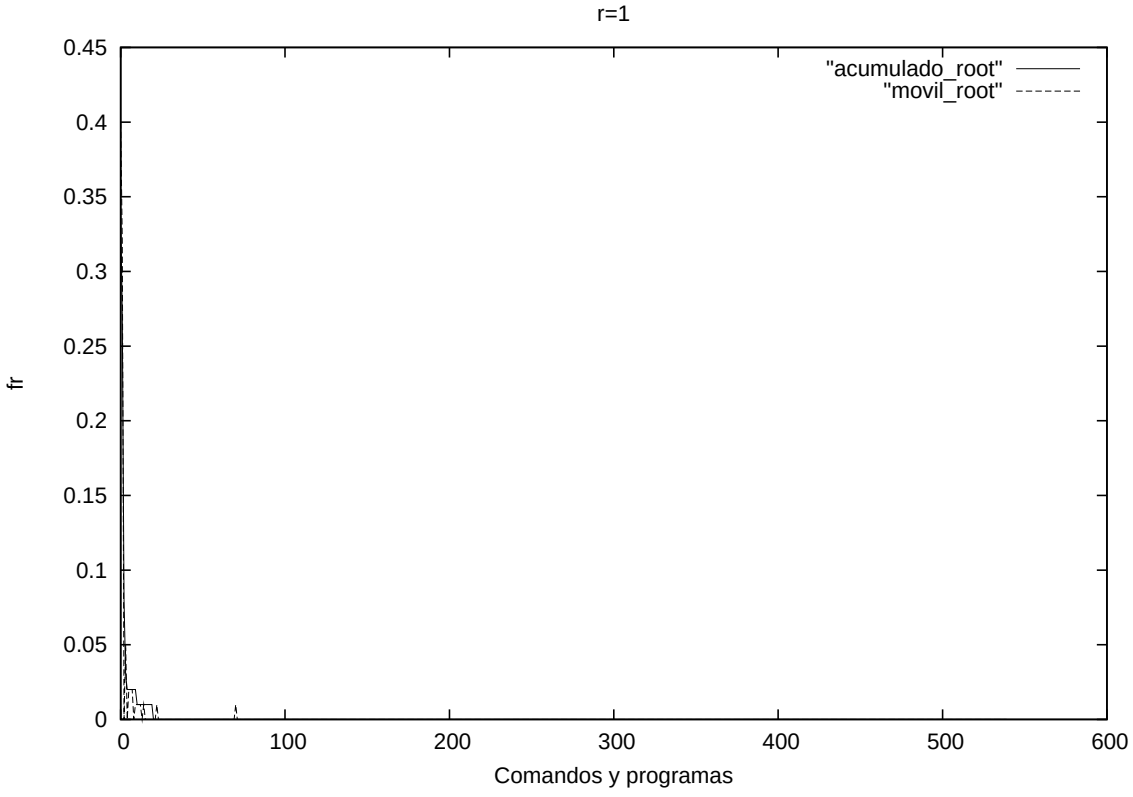


Figura 4.8: Perfiles acumulado y móvil del root.

El valor obtenido del coeficiente de correlación entre los perfiles acumulado y móvil del root es 1. Esto implica que la actividad del rootkit no fue detectada como una anorma-

lidad. La razón de ésto es porque un rootkit se comporta como una instalación de cualquier programa (ver Figura 4.9), es decir, ejecuta al compilador, mueve y elimina algunos archivos. Lo anterior también es realizado en una instalación normal de cualquier programa. La otra razón es que la gran mayoría de programas que realizan actividades necesarias para el sistema operativo requieren la identidad del root. Esto implica que predomine en el comportamiento root la parte atribuida al sistema operativo, minizando la parte correspondiente al administrador. Para poder detectar que la cuenta de root ya fue comprometida se debe de modelar de una manera más completa a los usuarios, tal como se plantea en la Sección 5.2. Esto es debido a que el intruso puede modificar el comportamiento del administrador en cuestiones de red, al realizar éste transferencias de archivos a otros sistemas.

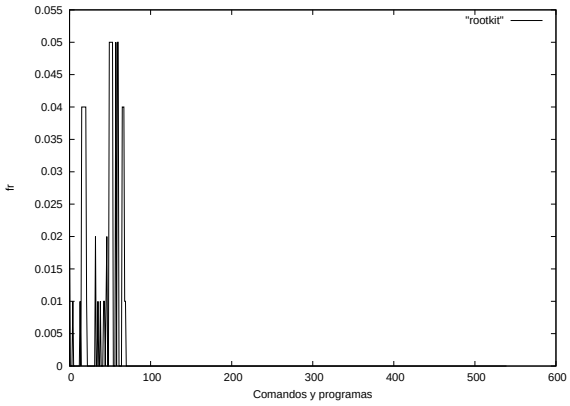


Figura 4.9: Comandos y Programas ejecutados por rootkit

Otra posible solución para conocer que la cuenta de root ha sido comprometida es utilizar firmas que contengan un conjunto de archivos protegidos. Los cuales no podran ser alterados ni eliminados del sistema.

4.4. Conclusión

En este Capítulo se mostraron algunos ejemplos de comparación entre los perfiles acumulado y móvil. Se mostraron cuatro histogramas correspondientes a usuarios participantes. Además se muestra la prueba realizada con un rootkit en la cuenta del root. En este caso no se detectó ninguna anomalía en el comportamiento exhibido por el rootkit. Esto debido a su semejanza o igualdad en el comportamiento de instalaciones de programas. Para

poder detectar que la cuenta de root ya fue comprometida es necesario modelar a los usuarios de una manera más completa y emplear firmas que indiquen la presencia de un rootkit. En particular, se logro el objetivo de ofrecer al presente trabajo como una base sólida de una herramienta de seguridad. Contar con una herramienta completa de seguridad es necesario realizar una gran cantidad de pruebas, las cuales demandan demasiados recursos (tiempo, usuarios, equipo, etc). El presente trabajo no es más que una pequeña contribución al basto mundo de ideas de seguridad en los sistemas de cómputo.

Capítulo 5

Conclusiones

5.1. Conclusiones Generales

Sin duda el arma más efectiva para cualquier atacante o intruso es la habilidad de encontrar defectos de un sistema, que posiblemente no sean evidentes para quienes lo diseñaron, o para quienes lo utilizan a diario.

Desafortunadamente, todos los días las máquinas conectadas a Internet son transgredidas, no por que sean inseguras, sino por que los pasos necesarios para exponerlos seguros al mundo real (Internet), no siempre son tan obvios.

El sistema requiere un mínimo de supervisión humana, no consume grandes periodos de tiempo al CPU. Las bases de datos de los usuarios no crecen de manera infinita y tarde o temprano alcanzan un tamaño máximo. Con ésto se puede garantizar que requieren un tamaño finito para su almacenamiento.

Por lo que respecta al presente trabajo, se cubrieron las características 1, 4, 5 y 6 deseables para los IDS, descritas en la Sección 1.2.1.

Este trabajo proporciona una buena referencia de las desviaciones del comportamiento de los usuarios. Cuando éstos aprenden nuevos conocimientos y los ponen en práctica, el sistema detecta los cambios. Así también que si un usuario es suplantado por un intruso. Cabe mencionar que para este caso, la detección depende de que tan diferente se comporta el intruso. Es importante notar con cierta probabilidad que el intruso puede comportarse como el usuario suplantado. Lo anterior debido al hecho de existir similitudes en el comportamiento del usuario suplantado e intruso.

Personalmente se concluye que la seguridad en los sistemas de cómputo ofrecen

un camino desafiante para todo aquel que desee incursionar en este campo. Para ofrecer buenas herramientas de seguridad se requiere de gran disciplina y dedicación, tanto para el aprendizaje como para las implementaciones y pruebas.

5.2. Trabajos Futuros

El presente trabajo puede servir como base para los siguientes trabajos futuros:

Adicionar más llamadas al sistema (ver Tabla 3.1), para contar con una auditoría más amplia.

Determinar el tamaño óptimo del periodo de muestreo para el perfil móvil, que disminuya el número de falsos positivos y negativos.

Incorporar un módulo de control que tome acciones cuando el valor del coeficiente se salga de los valores permisibles.

Integrar la información que ofrecen los registros del sistema (klog, syslog), para cubrir detección de intrusión por mal uso (ver sección 1.2). Otro de los experimentos preliminares realizados en el presente trabajo, fue el utilizar los registros del sistema. Para lo cual fue necesario, realizar modificaciones en el código de syslogd.

Integrar la detección de intrusión en red, mediante la librería lpcap [lpc05] (ver Sección 1.2.2).

Modelar de una manera más completa a los usuarios, preferencias de red (páginas web frecuentadas, ssh, sftp), tiempo de sesiones, frecuencia de sesiones, etc.

A petición de los revisores se propone también como trabajo futuro el utilizar correlaciones cruzadas o suma de diferencias al cuadrado. Esto con la finalidad de obtener una comparación con el coeficiente de correlación lineal.

Otro trabajo interesante es incorporar aprendizaje automático, para que el sistema aprenda los cambios del comportamiento del usuario.

Apéndice A

Comandos y Programas Ejecutados

En este apéndice se listan los 379 comandos y programas ejecutados durante un mes por 24 usuarios de la Sección de Investigación y Desarrollo de la Facultad de Ingeniería Eléctrica de la UMSNH. De igual manera se muestra el acumulado global para cada comando y programa.

Tabla A.1: Comandos y Programas Ejecutados por los 24 Usuarios Participates de la Prueba

Número	Path de Comando o Programa	Acumulado Global
1	/bin/sh	646
2	/bin/grep	596
3	/usr/bin/id	564
4	/bin/sed	524
5	/bin/basename	374
6	/usr/bin/cut	364
7	/bin/ls	248
8	/usr/X11R6/bin/xrdb	204
9	/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/tradcpp0	204
10	/usr/java/jdk1.5.0/bin/javac	204
11	/usr/bin/cpp	204
12	/bin/bash	195
13	/sbin/consoletype	193
14	/usr/bin/which	184
15	/usr/bin/tput	171

16	/usr/lib/mozilla-1.2.1/mozilla-xremote-client	170
17	/usr/bin/perl	158
18	/bin/egrep	134
19	/usr/local/bin/xrdb	132
20	/bin/xrdb	132
21	/usr/bin/xrdb	132
22	/usr/bin/vim	128
23	/usr/bin/dircolors	125
24	/usr/bin/mozilla	120
25	/usr/local/bin/mozilla	116
26	/bin/cp	110
27	/usr/java/j2re1.4.1_03/bin/java	104
28	/usr/java/j2re1.4.1_03/plugin/i386/ns610/../../../bin//java	104
29	/usr/java/jdk1.5.0/bin/java	96
30	/usr/bin/dirname	84
31	/usr/bin/gunzip	84
32	/bin/umount	82
33	/bin/hostname	81
34	/usr/X11R6/bin/mkfontdir	76
35	/bin/chmod	74
36	/usr/X11R6/bin/xsetroot	72
37	/usr/bin/clear	71
38	/usr/lib/mozilla-1.2.1/mozilla-bin	70
39	/usr/bin/ssh-agent	68
40	/usr/X11R6/bin/xmbind	68
41	/bin/true	68
42	/usr/X11R6/bin/xmodmap	68
43	/usr/bin/printenv	68
44	/etc/X11/xdm/Xsession	68
45	/usr/X11R6/bin/xscreensaver-getimage	66
46	/usr/bin/gnome-settings-daemon	64
47	/usr/local/bin/gnome-settings-daemon	64
48	/usr/bin/mkfontdir	64
49	/bin/xscreensaver-gl-helper	64
50	/usr/bin/xscreensaver	64

51	/usr/X11R6/lib/xscreensaver/xscreensaver-gl-helper	64
52	/usr/bin/xscreensaver-gl-helper	64
53	/usr/local/bin/xscreensaver-gl-helper	64
54	/usr/libexec/bonobo-activation-server	64
55	/usr/bin/nautilus	64
56	/usr/bin/gnome-session	64
57	/usr/bin/metacity	64
58	/usr/libexec/gconf-sanity-check-2	64
59	/bin/xscreensaver	64
60	/usr/X11R6/bin/xscreensaver-gl-helper	64
61	/usr/local/bin/mkfontdir	64
62	/usr/local/bin/nautilus	64
63	/bin/mkfontdir	64
64	/usr/X11R6/bin/xscreensaver	64
65	/usr/local/bin/xscreensaver	64
66	/usr/bin/wc	63
67	/usr/sbin/chkfontpath	60
68	/usr/libexec/gconfd-2	58
69	/sbin/pam_timestamp_check	58
70	/bin/rm	58
71	/etc/X11/gdm/Sessions//Default	58
72	/etc/X11/xinit/Xclients	58
73	/usr/bin/pam-panel-icon	58
74	/bin/stty	57
75	/usr/bin/eggcups	56
76	/usr/bin/rhn-applet-gui	56
77	/usr/local/bin/gnome-panel	54
78	/usr/local/bin/eggcups	54
79	/usr/bin/magicdev	54
80	/usr/bin/gnome-panel	54
81	/usr/local/bin/magicdev	54
82	/bin/netstat	52
83	/usr/libexec/notification-area-applet	52
84	/usr/bin/gconftool-2	50
85	/usr/local/bin/pam-panel-icon	48

86	/usr/local/bin/rhn-applet-gui	48
87	/usr/local/bin/gnome-wm	48
88	/usr/bin/gnome-wm	48
89	/usr/libexec/nautilus-throbber	46
90	/bin/mkdir	42
91	/bin/uname	42
92	/bin/cat	38
93	/usr/bin/test	38
94	/usr/bin/gnome-terminal	38
95	/usr/bin/esd	36
96	/usr/bin/gnome-background-properties	36
97	/usr/lib/openoffice/program/soffice.bin	36
98	/usr/bin/man	36
99	/usr/lib/vte/gnome-pty-helper	36
100	/usr/local/bin/gnome-background-properties	36
101	/usr/bin/gnome-session-save	34
102	/usr/local/bin/gnome-terminal	34
103	/usr/local/bin/gnome-session-save	34
104	/usr/bin/less	32
105	/usr/bin/locale	30
106	/usr/bin/gtbl	28
107	/usr/bin/ooffice	28
108	/bin/mount	28
109	/usr/bin/grotty	28
110	/usr/bin/troff	28
111	/usr/bin/groff	28
112	/usr/kerberos/bin/xrdb	28
113	/usr/bin/nroff	28
114	/usr/bin/xpdf	28
115	/usr/bin/awk	26
116	/usr/bin/tr	26
117	/bin/cut	26
118	/bin/date	24
119	/usr/bin/lpstat	24
120	/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/cc1	22

121	/usr/local/bin/as	22
122	/usr/sbin/lpc	22
123	/bin/ps	22
124	/usr/bin/gcc	22
125	/usr/bin/eject	22
126	/usr/local/bin/eject	22
127	/usr/bin/as	22
128	/usr/X11R6/bin/iceauth	20
129	/bin/df	20
130	/usr/bin/ld	20
131	/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/collect2	20
132	/usr/bin/usermount	18
133	/usr/local/bin/usermount	18
134	/usr/sbin/userhelper	16
135	/usr/kerberos/bin/mozilla	16
136	/usr/bin/consolehelper-gtk	16
137	/usr/local/bin/metacity	16
138	./hola	16
139	/bin/tar	16
140	/usr/kerberos/bin/magicdev	14
141	/usr/kerberos/bin/gnome-panel	14
142	/usr/kerberos/bin/gnome-wm	14
143	/usr/kerberos/bin/gnome-terminal	14
144	/usr/local/bin/ooffice	14
145	/usr/kerberos/bin/nautilus	14
146	/usr/kerberos/bin/xscreensaver-gl-helper	14
147	/usr/kerberos/bin/gnome-settings-daemon	14
148	/usr/bin/dcopserver_shutdown	14
149	/usr/kerberos/bin/mkfontdir	14
150	/usr/kerberos/bin/xscreensaver	14
151	/usr/kerberos/bin/pam-panel-icon	14
152	/home/emmanuel/.openoffice/soffice	14
153	/usr/bin/lpr	14
154	/usr/kerberos/bin/eggcups	14
155	/usr/kerberos/bin/rhn-applet-gui	14

156	/usr/bin/qcad	12
157	/usr/kerberos/bin/ooffice	12
158	/usr/local/bin/df	12
159	/usr/local/bin/gzip	12
160	/usr/local/bin/qcad	12
161	/usr/bin/lnusertemp	12
162	/usr/bin/env	12
163	/usr/bin/file-roller	12
164	/usr/X11R6/bin/xset	12
165	/usr/local/bin/file-roller	12
166	/usr/local/bin/lpstat	12
167	/usr/bin/gzip	12
168	/usr/sbin/utempter	12
169	/usr/bin/df	12
170	/usr/bin/kdeinit	10
171	/usr/bin/poweroff	8
172	/bin/echo	8
173	/bin/kbd_mode	8
174	/usr/bin/kfmclient	8
175	/home/usuarios/a0468892x/bin/qcod	8
176	/usr/X11R6/lib/xscreensaver/vines	8
177	/bin/loadkeys	8
178	/usr/X11R6/bin/qcod	8
179	/usr/X11R6/lib/xscreensaver/starfish	8
180	/bin/dumpkeys	8
181	/bin/setfont	8
182	/usr/X11R6/bin/xauth	8
183	/usr/bin/dir	8
184	/bin/unicode_start	8
185	/usr/bin/qcod	8
186	/usr/local/bin/qcod	8
187	/bin/mktemp	8
188	/usr/X11R6/lib/xscreensaver/metaballs	8
189	/bin/gzip	8
190	/bin/qcod	8

191	/usr/lib/openoffice/program/soffice	8
192	/usr/X11R6/lib/xscreensaver/grav	8
193	/usr/bin/xargs	8
194	/usr/bin/who	8
195	/usr/X11R6/lib/xscreensaver/attraction	8
196	/usr/share/apps/switchdesk/Xclients.gnome	6
197	/usr/local/bin/ooimpress	6
198	/usr/bin/top	6
199	/usr/X11R6/lib/xscreensaver/lightning	6
200	/bin/more	6
201	/usr/libexec/gnome-print-manager-remote	6
202	/usr/bin/ooimpress	6
203	/usr/bin/plugger	6
204	/etc/X11/gdm/Sessions//GNOME	6
205	/usr/bin/scrollkeeper-config	6
206	/usr/local/bin/plugger	6
207	/usr/kerberos/bin/plugger	6
208	/usr/local/bin/rm	6
209	/usr/X11R6/lib/xscreensaver/wander	6
210	/usr/bin/rm	6
211	/home/usuarios/a0468892x/.openoffice/soffice	6
212	/usr/bin/ssh	5
213	/usr/bin/ksplash	4
214	/usr/bin/ktip	4
215	/usr/X11R6/lib/xscreensaver/xlyap	4
216	/bin/rpm	4
217	/usr/X11R6/lib/xscreensaver/squiral	4
218	/usr/X11R6/lib/xscreensaver/forest	4
219	/usr/X11R6/lib/xscreensaver/distort	4
220	/usr/X11R6/lib/xscreensaver/halo	4
221	/usr/local/bin/gnome-print-manager	4
222	/usr/java/j2re1.4.1_03/bin/java_vm	4
223	/usr/bin/gfloppy	4
224	/usr/X11R6/lib/xscreensaver/worm	4
225	/usr/local/bin/oocalc	4

226	/bint/arch	4
227	/usr/libexec/metacity-dialog	4
228	/usr/sbin/adduser	4
229	/usr/bin/desktop-create-kmenu	4
230	/usr/X11R6/lib/xscreensaver/crystal	4
231	/usr/bin/oocalc	4
232	/usr/X11R6/lib/xscreensaver/rd-bomb	4
233	/usr/X11R6/lib/xscreensaver/rubik	4
234	/usr/X11R6/lib/xscreensaver/spotlight	4
235	/sbin/mkdosfs	4
236	/usr/X11R6/lib/xscreensaver/truchet	4
237	/usr/bin/startkde	4
238	/usr/bin/tryprint.pl	4
239	./s1	4
240	/usr/bin/artswrapper	4
241	/usr/X11R6/lib/xscreensaver/lisa	4
242	/usr/share/apps/switchdesk/Xclients.kde	4
243	/usr/bin/oowriter	4
244	/usr/X11R6/lib/xscreensaver/maze	4
245	/usr/X11R6/lib/xscreensaver/helix	4
246	/usr/local/bin/tryprint.pl	4
247	/usr/bin/gs	4
248	/usr/X11R6/lib/xscreensaver/blaster	4
249	/usr/X11R6/lib/xscreensaver/barcode	4
250	/usr/X11R6/lib/xscreensaver/superquadrics	4
251	/usr/bin/artsd	4
252	/usr/X11R6/lib/xscreensaver/ripples	4
253	/usr/X11R6/lib/xscreensaver/mountain	4
254	/usr/X11R6/lib/xscreensaver/sonar	4
255	/usr/X11R6/lib/xscreensaver/gears	4
256	/usr/X11R6/lib/xscreensaver/critical	4
257	/usr/X11R6/lib/xscreensaver/epicycle	4
258	/usr/X11R6/lib/xscreensaver/gltext	4
259	/etc/X11/gdm/Sessions//KDE	4
260	/usr/X11R6/lib/xscreensaver/greynetic	4

261	/usr/bin/gnome-print-manager	4
262	/usr/X11R6/lib/xscreensaver/triangle	4
263	/home/usuarios/a9900750x/.openoffice/soffice	4
264	/usr/X11R6/lib/xscreensaver/galaxy	4
265	/usr/X11R6/lib/xscreensaver/halftone	4
266	/usr/X11R6/lib/xscreensaver/morph3d	4
267	/usr/X11R6/lib/xscreensaver/jigsaw	4
268	/usr/local/bin/gfloppy	4
269	/usr/bin/autorun	4
270	/usr/bin/kwrapper	4
271	/usr/local/bin/oowriter	4
272	/usr/X11R6/lib/xscreensaver/boxed	4
273	/usr/bin/kdeinit_shutdown	2
274	/usr/X11R6/lib/xscreensaver/menger	2
275	/usr/bin/scrollkeeper-get-cl	2
276	/usr/X11R6/lib/xscreensaver/qix	2
277	/usr/bin/scrollkeeper-get-content-list	2
278	/usr/lib/openoffice/program/setup	2
279	/usr/X11R6/lib/xscreensaver/speedmine	2
280	/usr/local/bin/manpath	2
281	/usr/X11R6/lib/xscreensaver/cynosure	2
282	/sbin/modprobe	2
283	/sbin/iptables	2
284	/usr/local/bin/htmlview	2
285	/usr/bin/ggv	2
286	/usr/bin/htmlview	2
287	/usr/kerberos/bin/oowriter	2
288	/usr/X11R6/lib/xscreensaver/hypercube	2
289	/usr/X11R6/lib/xscreensaver/strange	2
290	/usr/bin/uptime	2
291	/usr/X11R6/lib/xscreensaver/xmatrix	2
292	/usr/libexec/yelp-db2html	2
293	/sbin/pidof	2
294	/usr/X11R6/lib/xscreensaver/slidescreen	2
295	/usr/X11R6/lib/xscreensaver/lissie	2

296	/usr/X11R6/lib/xscreensaver/spiral	2
297	/usr/X11R6/lib/xscreensaver/flame	2
298	/home/usuarios/a0122828d/.openoffice/soffice	2
299	/usr/X11R6/lib/xscreensaver/fadeplot	2
300	/usr/X11R6/lib/xscreensaver/moire	2
301	/usr/bin/artsshell	2
302	/usr/local/bin/kpoker	2
303	/usr/X11R6/bin/xdpyinfo	2
304	/usr/bin/xmms	2
305	/usr/local/bin/gnome-search-tool	2
306	/usr/X11R6/lib/xscreensaver/braid	2
307	/usr/X11R6/lib/xscreensaver/eruption	2
308	/usr/X11R6/lib/xscreensaver/flipscreen3d	2
309	/usr/X11R6/lib/xscreensaver/cubenetic	2
310	/usr/bin/kde-config	2
311	/usr/bin/xcdroast	2
312	/bin/mv	2
313	/usr/X11R6/lib/xscreensaver/imsmap	2
314	/usr/X11R6/lib/xscreensaver/sproingies	2
315	/usr/bin/ksirtet	2
316	/usr/bin/scrollkeeper-get-toc-from-docpath	2
317	/usr/bin/whoami	2
318	/usr/X11R6/lib/xscreensaver/pulsar	2
319	/usr/X11R6/lib/xscreensaver/kumppa	2
320	/usr/X11R6/lib/xscreensaver/glplanet	2
321	/usr/X11R6/lib/xscreensaver/pedal	2
322	/usr/bin/manpath	2
323	/usr/X11R6/lib/xscreensaver/stairs	2
324	/usr/X11R6/lib/xscreensaver/penetrate	2
325	/usr/X11R6/lib/xscreensaver/molecule	2
326	/bin/su	2
327	/usr/X11R6/lib/xscreensaver/stonerview	2
328	/usr/X11R6/lib/xscreensaver/coral	2
329	/home/usuarios/a0468881d/.openoffice/soffice	2
330	/usr/lib/openoffice/program/setup.bin	2

331	/usr/X11R6/lib/xscreensaver/anemone	2
332	/usr/local/bin/ggv	2
333	/usr/X11R6/lib/xscreensaver/pyro	2
334	/usr/bin/kpoker	2
335	/usr/local/bin/scrollkeeper-config	2
336	/usr/X11R6/lib/xscreensaver/euler2d	2
337	/usr/kerberos/bin/ggv	2
338	/usr/local/bin/ksirtet	2
339	./.bash_profile	2
340	/usr/X11R6/lib/xscreensaver/rocks	2
341	/usr/libexec/gnome_segv2	2
342	/usr/bin/java	2
343	/usr/X11R6/lib/xscreensaver/twang	2
344	/usr/X11R6/lib/xscreensaver/gflux	2
345	/usr/bin/sftp	2
346	/usr/local/bin/xmms	2
347	/usr/X11R6/lib/xscreensaver/xspirograph	2
348	/usr/X11R6/lib/xscreensaver/kaleidescope	2
349	/usr/bin/halt	2
350	/usr/local/bin/scrollkeeper-get-toc-from-docpath	2
351	/usr/X11R6/lib/xscreensaver/polyominoes	2
352	/usr/local/bin/scrollkeeper-get-content-list	2
353	/usr/bin/head	2
354	/usr/X11R6/lib/xscreensaver/xteevee	2
355	/usr/X11R6/lib/xscreensaver/nerverot	2
356	/usr/bin/lessecho	2
357	/usr/libexec/ggv-postscript-viewer	2
358	ls/mnt/ploppy/	2
359	/usr/local/bin/gnome-help	2
360	/usr/X11R6/lib/xscreensaver/blitspin	2
361	/usr/local/bin/kshell	2
362	/bin/	2
363	/usr/X11R6/lib/xscreensaver/sierpinski	2
364	/usr/bin/gnome-help	2
365	/usr/X11R6/lib/xscreensaver/zoom	2

366	/usr/bin/unzip	2
367	/usr/lib/openoffice/program/gnomeint	2
368	/usr/bin/kate	2
369	/usr/kerberos/bin/gzip	2
370	/usr/bin/kshell	2
371	/usr/bin/reboot	2
372	/usr/bin/gcj	2
373	/usr/X11R6/lib/xscreensaver/loop	2
374	/usr/X11R6/lib/xscreensaver/lavalite	2
375	/usr/X11R6/lib/xscreensaver/ant	2
376	/usr/bin/redhat-config-rootpassword	2
377	/usr/X11R6/lib/xscreensaver/flag	2
378	/bin/ln	2
379	/usr/bin/gnome-search-tool	2
	Total de ejecuciones	12887

Apéndice B

Grupos Estándar del Sistema

Tabla B.1: Grupos Estándar del Sistema Operativo Linux

Grupo	GID	Miembros
uucp	14	uucp
man	15	
games	20	
gopher	30	
dip	40	
ftp	50	
lock	54	
nobody	99	
users	100	
rpm	37	rpm
utmp	22	
floppy	19	
vcsa	69	
ntp	38	
canna	39	
nscd	28	
rpc	32	
postdrop	90	
postfix	89	
named	25	
postgres	26	
sshd	74	
rpcuser	29	
nfsnobody	65534	
pvm	24	
apache	48	
xfs	43	
desktop	80	
gdm	42	
mysql	27	
webalizer	67	
mailman	41	
mailnull	47	
smmsp	51	
squid	23	
ldap	55	
netdump	34	
pcap	77	
ident	98	
privoxy	101	
radvd	75	
fax	78	
slocate	21	
wnn	49	

Apéndice C

Niveles de Seguridad

El Criterio de Evaluación de Sistemas de Computadoras Fiables (*Trusted Computer System Evaluation Criteria -TCSEC-*), es una normativa también conocida como libro naranja [DOD85], por el color de su portada. Esta normativa suministra los requisitos de seguridad para sistemas de procesamiento automático de datos. Este libro se ve complementado con el llamado libro rojo, que extiende la interpretación de este para la evaluación de sistemas fiables a redes de computadoras. Estos libros se desarrollaron para que los usuarios tuvieran una métrica para que determinar el grado de fiabilidad que pueden poner en un sistema.

El criterio de evaluación TSEC se divide en cuatro clases principales, que van desde la D (la menos segura) hasta la A (la más segura).

Clase D: Protección mínima

Esta división contiene sólo una clasificación. Está reservado para los sistemas que han sido evaluados pero que no cumplen con los requisitos para una clase de evaluación más alta.

I. Nivel D1 Protección mínima de seguridad. En esta clasificación no existe ningún tipo de control, ya sea de acceso o para garantizar la protección de la información.

Clase C: Protección discreta

Esta división proporciona protección discreta mediante de la capacidad de auditoría.

I. Nivel C1 Protección de seguridad discreta. Describe un sistema donde el usuario se identifica con un nombre y contraseña. Existe cierta protección en archivos y directorios para que solo el usuario propietario pueda utilizarlos. En este nivel el administrador tiene acceso y control ilimitado del sistema.

II. Nivel C2 Protección de acceso controlado. En este nivel se tienen todas las características del nivel C1 y además se agrega una auditoría para procesos críticos. En este nivel se auditan las actividades de los usuarios.

Clase B: Protección obligatoria

En esta división se requiere de todos las características requeridas por el nivel C2, además de un modelo informal de política de seguridad, para etiquetar los datos y el acceso a ellos.

I. Nivel B1 Protección de seguridad etiquetada. En este nivel se define que un objeto bajo control de acceso, no puede sufrir cambios en los permisos definidos por el propietario del archivo.

II. Nivel B2 Protección estructurada. Este nivel exige que cada objeto del sistema asignado, sencillo o múltiple, tenga su propio sistema de protección.

III. Nivel B3 Dominios de seguridad. En este nivel se requiere de la instalación de hardware para acceso a dominios. El usuario tiene que conectarse al sistema mediante de una ruta de acceso.

Clase A: Protección verificada

Esta división se caracteriza por el uso de métodos de comprobaciones de seguridad formal, para asegurar que los controles de seguridad, obligatorios y discrecionales, empleados en el sistema puedan proteger eficazmente la información procesada por el sistema. Se exige una documentación extensa para demostrar que la base segura de planeación, desarrollo y aplicación.

I. Nivel A1 Diseño verificado. En este nivel se exige un proceso de diseño, control y verificación de todos los niveles de seguridad. El hardware y software deben estar protegidos contra violaciones en los sistemas de seguridad.

Apéndice D

Histogramas de los Usuarios Participantes

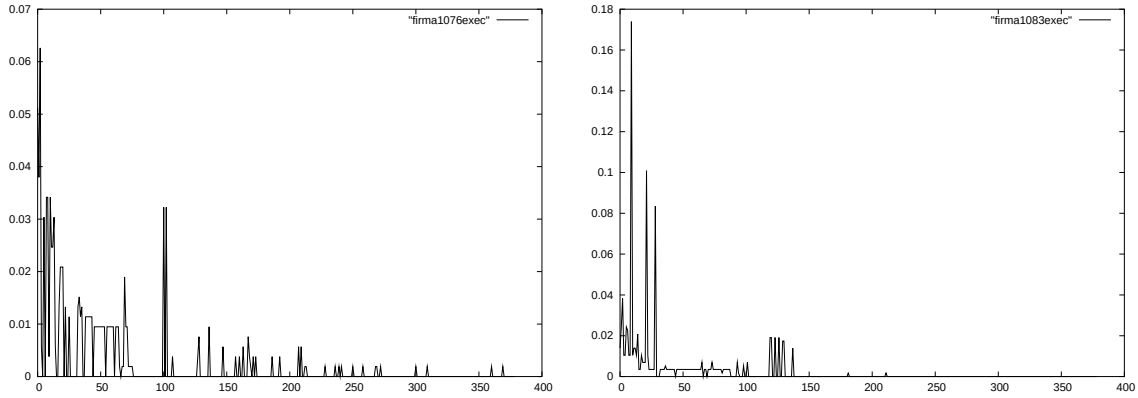


Figura D.1: Histogramas de ejecuciones de los usuarios 0 y 1

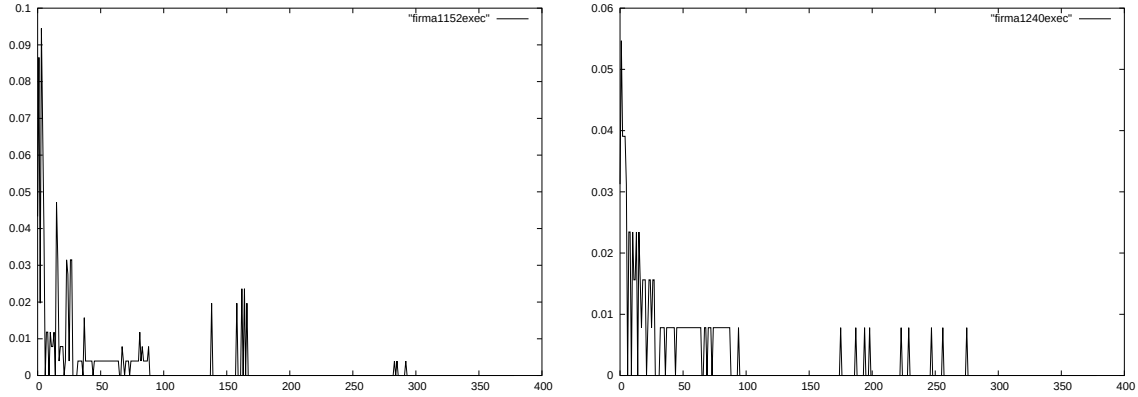


Figura D.2: Histogramas de ejecuciones de los usuarios 2 y 3

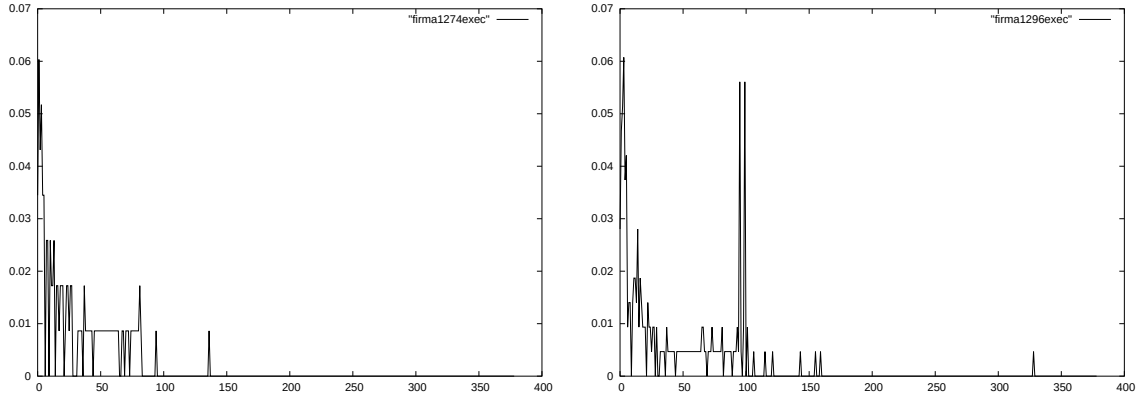


Figura D.3: Histogramas de ejecuciones de los usuarios 4 y 5

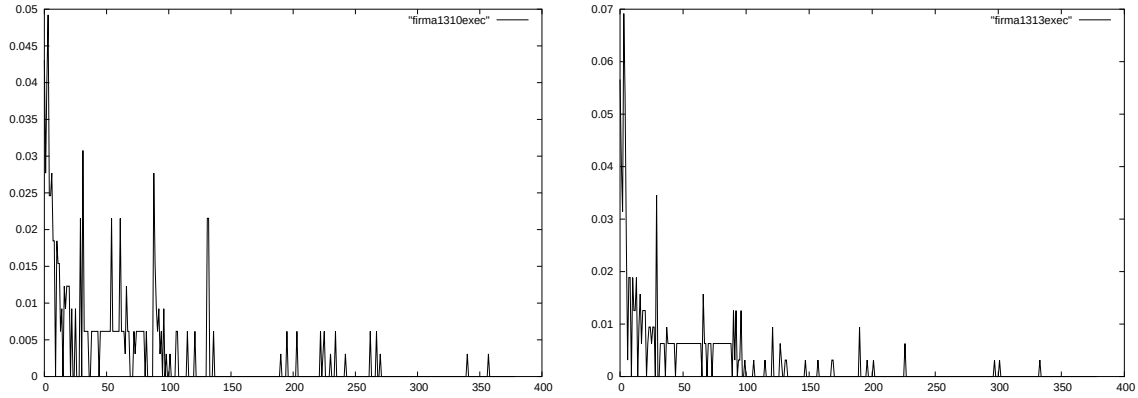


Figura D.4: Histogramas de ejecuciones de los usuarios 6 y 7

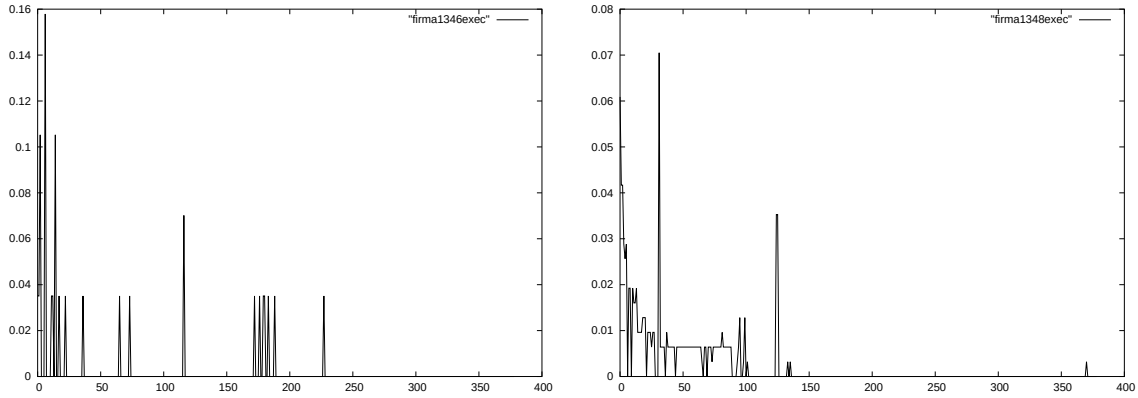


Figura D.5: Histogramas de ejecuciones de los usuarios 8 y 9

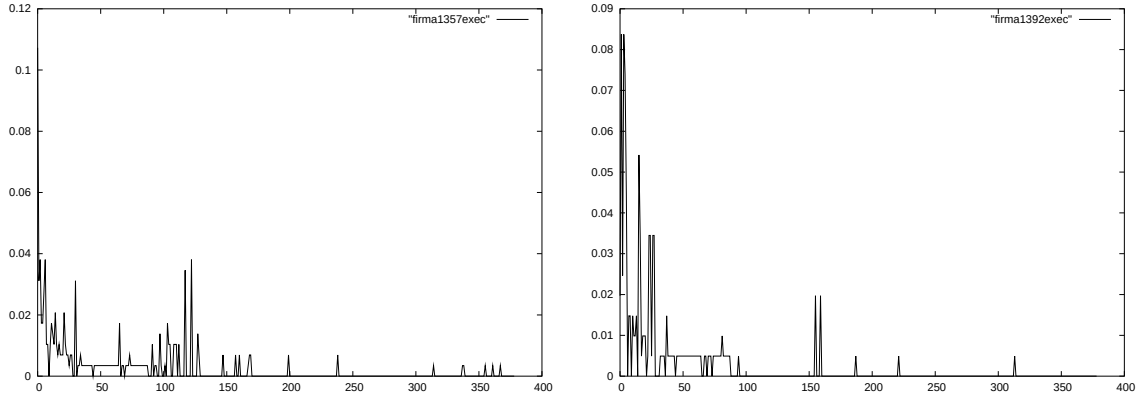


Figura D.6: Histogramas de ejecuciones de los usuarios 10 y 11

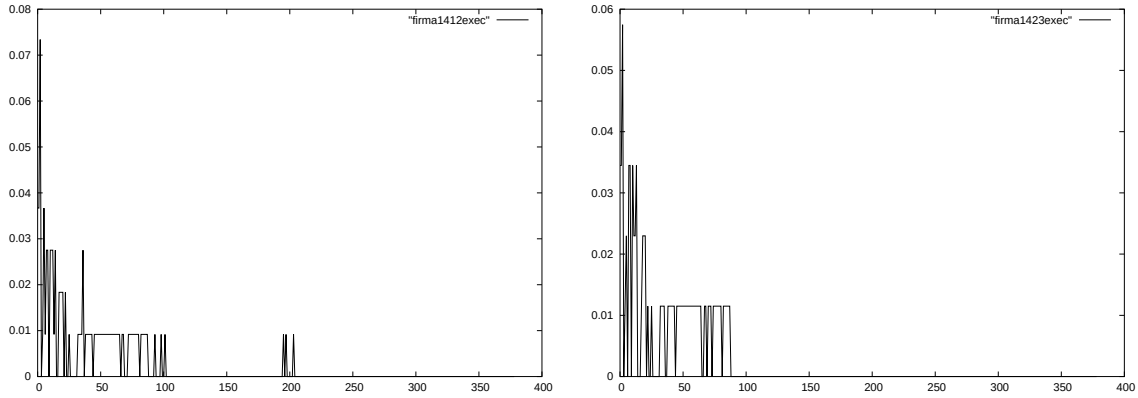


Figura D.7: Histogramas de ejecuciones de los usuarios 12 y 13

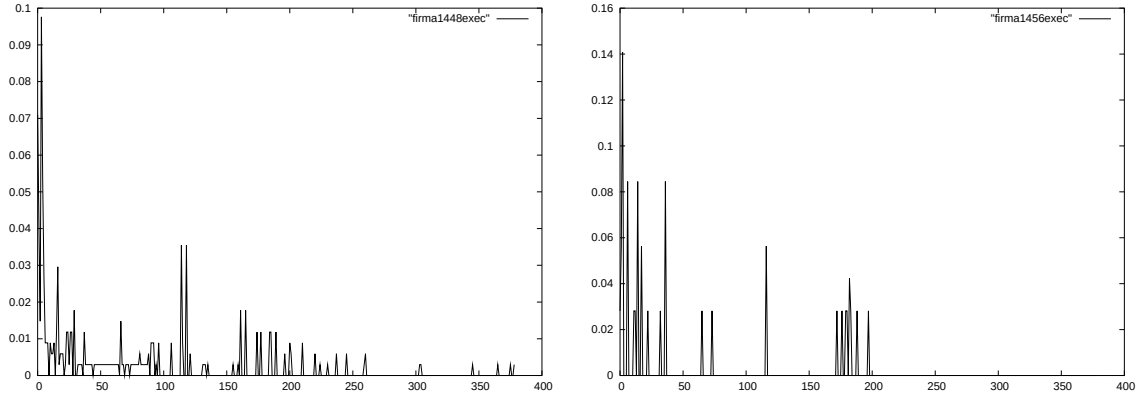


Figura D.8: Histogramas de ejecuciones de los usuarios 14 y 15

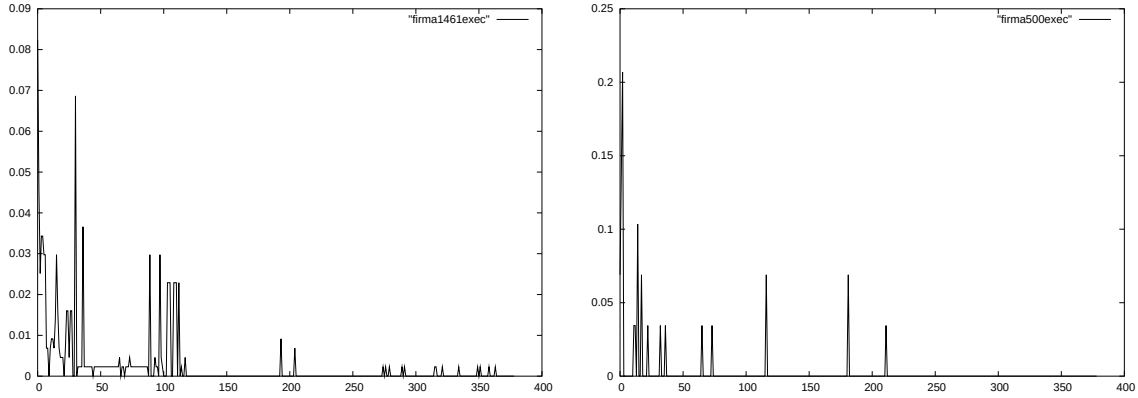


Figura D.9: Histogramas de ejecuciones de los usuarios 16 y 17

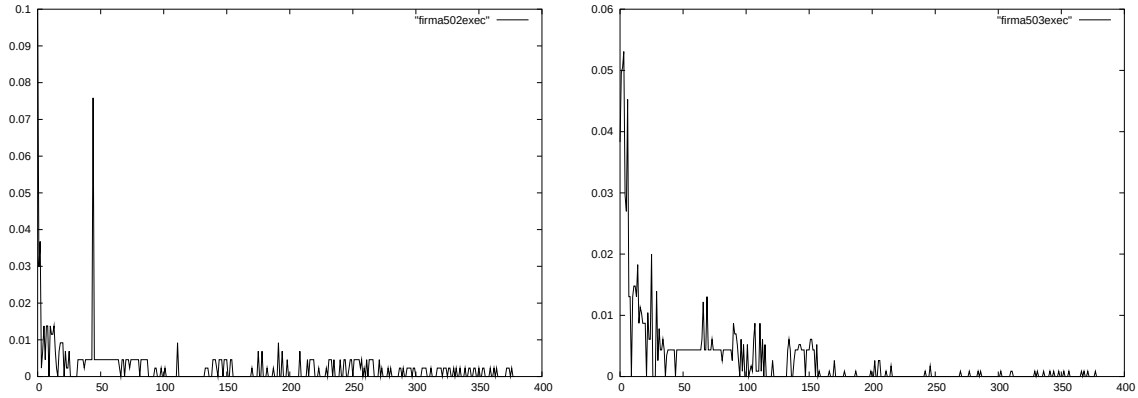


Figura D.10: Histogramas de ejecuciones de los usuarios 18 y 19

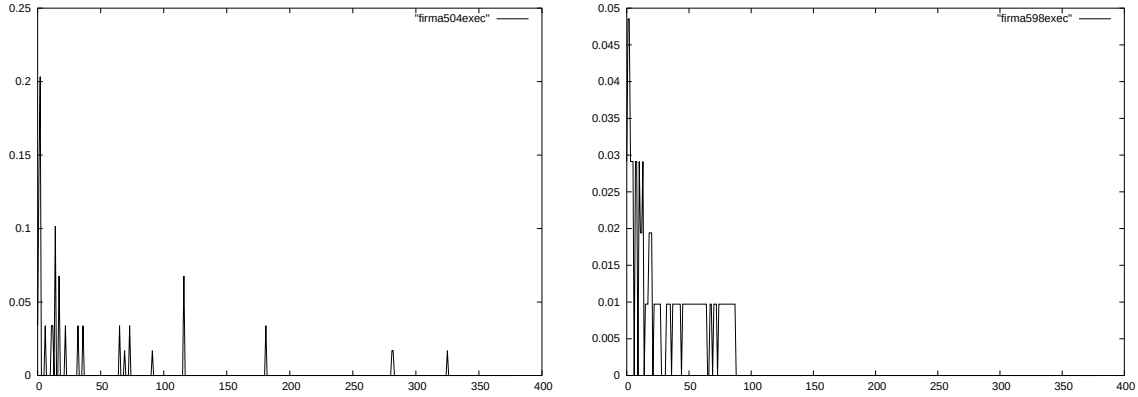


Figura D.11: Histogramas de ejecuciones de los usuarios 20 y 21

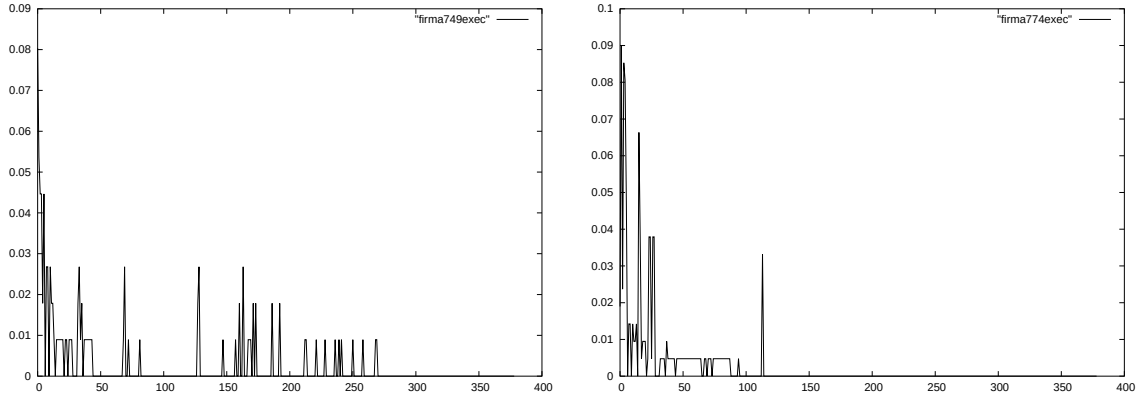


Figura D.12: Histogramas de ejecuciones de los usuarios 22 y 23

Referencias

[AIX01] AIX. *Communications Programming Concepts*. AIX, 2001.

[Allen B. Tucker00] Allen B. Tucker, J. *The Computer Science and Engineering Handbook*. CRC Press, 2000.

[Allen00] Allen, J. y Christie, A. State of the practice of intrusion detection technologies. Inf. téc., Carnegie Mellon, Software Engineering Institute, January 2000.

[Allenb.Tucker02] Allen b. Tucker, J. *The Computer Science and Engineering Handbook*. CRC, 2002.

[Anderson80] Anderson, J. P. Computer security threat monitoring and surveillance. 1980. James P. Anderson Co.

[Anderson95] Anderson, D., Frivold, T., y a Valdes. Nides. 1995. SRI.

[Bace00] Bace, R. y Mell, P. Intrusion detection systems. Inf. téc., National Institute of Standards and Technology, 2000.

[Casas02] Casas, J. G. y Magidin, A. *Introducción a la Criptología*. Universidad Nacional Autónoma de México, 2002.

[CERT03] CERT. Cert/cc overview. Inf. téc., Carnegie Mellon University, 2003.

[CERT05] CERT. Computer emergency responce team, 2005. Http://www.cert.org.

[Cisco05] Cisco. Cisco systems, inc., 2005. Http://www.cisco.com.

[Crosbie95] Crosbie, M. y Spafford, G. Active defense of a computer system using autonomous agents. 1995. COAST Laboratory Purdue University.

[Daniel83] Daniel y Terrell. *Business Statistics*. Houghton, 1983.

[Debar92] Debar, H., Becker, M., y Siboni, D. A neural network component for an intrusion detection system. 1992. IEEE.

[Denning86] Denning, D. E. An intrusion detection model. 1986. SRI International.

[DOD85] DOD. Departament of defense trsuted computer system evaluation criteria. Inf. téc., Departament of Defense, 1985.

[Dowel90] Dowel, C. y Ramstedt, P. The computer watch data reduction tool. 1990. NIST.

[Fink94] Fink, C. K. G. y Levitt, K. Automated detection of vulnerabilities in privileged programas by execution monitoring. 1994. IEEE.

[Flegel00] Flegel, U. Pseudonymizing unix log files. 2000. University of Dortmund Germany.

[Golberg96] Golberg, I., Wagner, D., Thomas, R., y Brewer, E. A secure enviroment for untrusted helper applications. 1996. USENIX.

[Group05] Group, L. N. R. Tcpdump/libcap, 2005. Http://www.tcpdump.org.

[Habra92] Habra, J., charlier, B. L., Mounji, A., y Mathieu, I. Asax:software architecture and rule based language for universal audit trail analisys. 1992. Springer Verlag.

[Heberlein90] Heberlein, T., Dias, G., Levitt, K., biswanath Mukherjee, Wood, J., y Wolber, D. A network security monitor. 1990. Computer Scociety Press.

[IEEE04] IEEE, T. y Group, T. O. *Definitions for ndbm database operations*. Portable Operating System Interface (POSIX), 2004. Http://www.opengroup.org/onlinepubs/009695399/basedefs/ndbm.h.html.

[Ilgun93] Ilgun, K. Ustat: A real time intrusion detection system for unix. 1993. IEEE.

[Int04] InterSect Alliance, PO Box 120, Belconnen, Camberra ACT 2617, Australia. *Guide to SNARE for LINUX*, 2004. Http://www.intersectalliance.com.

[Jackson91] Jackson, K. A., Dubois, D. H., y Stallings, C. A. An expert system application for network intrusion detection. 1991. NIST.

[Jou97] Jou, Y. F., Gong, F., candru Sargor, Wu, S. F., y Rance, C. W. Architecture design of a scalable intrusion detection system for the emerging network infrastructure. 1997. North Carolina State University.

[Kevin D. Mitnick02] Kevin D. Mitnick, S. W., William L. Simon. *The Art of Deception:Controlling the Human Element of Security.* Robert Ipsen, 2002.

[Kossakowski99] Kossakowski, K.-P. Glossary of computer security incident handling terms and abbreviations. Inf. téc., DFN-CERT, 1999. Http://www.dfn-cert.de/eng/pre99papers/certterm.html.

[Kumar94] Kumar, S. y Spafford, E. H. A pattern matching model for misuse intrusion detection. 1994. Purdue University.

[lid05] Lids, 2005. Http://www.lids.org/.

[Lincold Laboratory99] Lincold Laboratory, M. Intrusion detection attacks database, 1999. Http://www.ll.mit.edu/IST/ideval/docs/1999/attackDB.html.

[Lippmann98] Lippmann, R. P. y Cunningham, R. K. Improving intrusion detection performance using keyword selection and neural networks. 1998. MIT.

[lpc05] Lpacp, 2005. Http://www.tcpdump.org/.

[Lunt88] Lunt, T. F., Jagannathan, R., Lee, R., Listgarten, S., Edwards, D. L., y Neumann, P. G. Ides: The enhanced prototype, a real time intrusion detection system. 1988. SRI.

[Mark Mitchell01] Mark Mitchell, J. O. y Samuel, A. *Advanced Linux Programming*. New Riders, 2001.

[mas05] Mastercard warns of massive credit card-breach, 2005. Http://www.securityfocus.com/news/11219.

[Me01] Me, L. y Michel, C. Intrusion detection: A bibliography. 2001. Supelec.

[Mukkamala02] Mukkamala, S., Janoski, G., y Sung, A. Intrusion detection using neural networks and support vector machines. 2002. Institute of Mining and Technology Socorro New Mexico.

[NCSA98] NCSA. The common gateway interface, 1998. Http://hoohoo.ncsa.uiuc.edu/cgi/.

[nem05] Nemesis, 2005. Http://nemesis.sourceforge.net/.

[nma05] Nmap, 2005. Http://www.insecure.org/nmap/.

[ofDefense USA83] of Defense USA, D. *Departament of Defense Trusted Computer System Evaluation Criteria*. DoD, 1983.

[ofDefense05] of Defense, D. The defense advanced research projects agency, 2005. Http://www.darpa.mil.

[org.05] org., S. The open source network intrusion detection system, 2005. Http://www.snort.org.

[Paxon98] Paxon, V. Bro: A system for detecting network intruders in real time. 1998. USENIX.

[Red03a] Red Hat, Inc., PO Box 13588 Research Triangle Park NC 27709 USA. *Red Hat Linux 9: Red Hat Security Guide*, 2003. Http://www.redhat.com.

[Red03b] Red Hat, Inc., PO Box 13588 Research Triangle Park NC 27709 USA. *Red Hat Linux 9: Red Hat System Administration Primer*, 2003. Http://www.redhat.com.

[roo05] Rootkit, 2005. Http://searchsecurity.techtarget.com/.

[Ryan98] Ryan, J., Li, M.-J., y Miikkulainen, R. Intrusion detection with neural networks. 1998. Departament of Computer Science The university of Texas at Austin.

[Schiffman00] Schiffman, M. D. Libnet 101, part1: The primer. Inf. téc., Guardent, 2000. Http://www.insecure.org/tools.html.

[Sebring88] Sebring, M. M., Shellhouse, E., Hanna, M. E., y Whitehurst, R. A. Expert systems in intrusion detection: A case study. 1988. NIST.

[sel05] Selinux, 2005. Http://www.nsa.gov/selinux.

[Smaha88] Smaha, S. Haystack: An intrusion detection system. 1988. Computer Society Press.

[Snapp92] Snapp, S. R., Smaha, S. E., Teal, D. M., y Grance, T. The dids prototype. 1992. USENIX.

[sri05] Sri, 2005. Http://www.csl.sri.com/.

[Stallman01] Stallman, R. The GNU project, 2001. Http://www.gnu.org/. Updated: 2004/05/26.

[Sundar98] Sundar, J., Fernandez, J. O. G., Isacoff, D., Spafford, E., y Zamboni, D. An architecture for intrusion detection using autonomous agents. 1998. COAST Laboratory Purdue University.

[Systems99] Systems, I. S. Intrusion detection, take two, 1999. Http://www.nwc.com/1023/1023f12.html.

[Tanenbaum97] Tanenbaum, A. y Woodhull, A. *Operating Systems Deasign and Implementation*. Prentice Hall, 1997.

[Toxen00] Toxen, B. *Real World Linux Security*. PTR, 2000.

[Tri05] Tripwire, 2005. Http://www.tripwire.com/downloads/.

[Vaccaro89] Vaccaro, H. S. y Liepins, G. E. Detection of anomalous computer session activity. 1989.

[Wack91] Wack, J. P. Establishing a computer security incident response capability (csirc). 1991. NIST.

[Webster94] Webster, M. *Merriam Webster Dictionary*. Merriam Webster, 1994.

[Yañez00] Yañez, D. J. R. R. *Correlación*. StadHelp, 2000. Http://www.medal.org.ar/stadhelp/Std00014.htm.

[Zirkle05] Zirkle, L. What is host-based intrusion detection? 2005. Virginia Tech CNS.

Glosario

AID del Inglés *Anomaly Intrusion Detection*
(Detección de Intrusión por Anomalía).

Anomalía [**Bace00**] Anormalidad en el comportamiento.

ARPA del Inglés *Advanced Research Agency*.

ASCII del Inglés American Standar Code for Interchange Information
(Código Americano para Intercambio de Información).

Ataque [**Bace00**] Acción conducida por un adversario a una victima.

Auditoría [**Allenb.Tucker02**] Revisión de la *información* correspondiente a los eventos efectuados en un sistema para determinar cuando y como se realizaron o intentaron violaciones de seguridad.

CERT del Inglés Computer Emergency Response Team.

CGI del Inglés *Common Gateway Interface*.

CIA ConfidentialityIntegrityAvailability.

CID ConfidencialidadIntegridadDisponibilidad.

CID *Confidencialidad Integridad Disponibilidad*.

Confidencialidad [**Red03a**] Implica que los recursos no pueden ser leidos o adquiridos por personas no autorizadas.

DARPA [**ofDefense05**] del Inglés *Defense Advanced Research Projects Agency*.

DDOS del Inglés *Distributed Denial of Service*.

Disponibilidad [Red03a] Implica que los recursos deben estar disponibles a las personas autorizadas en el momento que se requiera.

DoD del Inglés *Department of Defense*.

DOS del Inglés *Denial of Service*.

Explotación [Bace00] Proceso de usar una vulnerabilidad para violar una política de seguridad.

Falso negativo [Bace00] Evento que un IDS determina que no es una intrusión y verdaderamente es si lo es.

Falso positivo [Bace00] Evento que un IDS determina como intrusión cuando verdaderamente no lo es.

gid [Mark Mitchell01] del Inglés *group identifier*.

gusano [roo05] Software malicioso que reproduce copias de si mismo.

HIDS del Inglés *Host Intrusion Detection System*
(Sistema Detecctor de Intrusión basado en Host).

Host Computadora que trabaja como anfitrión.

ID del Inglés *Intrusion Detection*
(Detector de Intrusión).

IDS del Inglés *Intrusion Detection System*
(Sistema Detector de Intrusión).

Incidente de seguridad de cómputo [Wack91] Evento adverso en el cual algún aspecto de la seguridad de un equipo de cómputo puede estar amenazado.

Integridad [Red03a] Implica que los recursos no pueden ser alterados por personas no autorizadas.

Intrusión [**Webster94**] Acción de introducirse sin derecho a un sitio.

Intruso [**Webster94**] Persona que realiza una intrusión.

Kernel [**Webster94**] Parte principal del sistema operativo.

LIDS [**lid05**] del Inglés *Linux Intrusion Detection System*.

MD5 [**Toxen00**] Algoritmo que obtine una llave de un archivo basada en sus caraterísticas.

MID del Inglés *del Inglés Misuse Intrusion Detection*
(Detección de Intrusión por mal uso).

MIT del Inglés *Massachusetts Instute of Technology*.

NDBM [**AIX01**] del Inglés *New Database Manager*.

NIDS del Inglés *Network Intrusion Detection System*
(Sistema Detecctor de Intrusión basado en Red).

PDP-1 Computadora marca Digital.

pid [**Mark Mitchell01**] del Inglés *process identifier*.

ppid [**Mark Mitchell01**] del Inglés *parent process identifier*.

Recursos [**Allenb.Tucker02**] datos o programas o parametros para control de acceso o llaves criptograficas o recursos computacionales.

rootkit [**roo05**] Colección de herramientas que usa un intruso para ocultar la intrusión.

Seguridad [**Webster94**] Libre de daño.

SELinux [**sel05**] del Inglés *Security Enhanced Linux*.

SNARE System iNtrusion Analysis & Reporting Enviroment.

Sondeo proceso de busqueda de vulnerabilidades y servicios disponibles.

TMRC del Inglés *Tech Model Railroad Club*.

U2L del Inglés *User to Local*.

U2R del Inglés *User to Root*.

uid [**Mark Mitchell01**] del Inglés *user identifier*.

Vulnerabilidad [**Kossakowski99**] Una característica o falla en un sistema o programa que permite a un atacante pasar las medidas de seguridad.

Índice