UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

DISEÑO E IMPLEMENTACION DE UN SISTEMA DE SEGURIDAD EN AMBIENTES LINUX/UNIX

T E S I S

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

PRESENTA

JAVIER IBARRA GARCÍA

ENSENADA, BAJA CALIFORNIA **Septiembre 2006**

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE CIENCIAS

DISEÑO E IMPLEMENTACION DE UN SISTEMA DE SEGURIDAD EN AMBIENTES LINUX/UNIX

T E S I S P R O F E S I O N A L

QUE PRESENTA

JAVIER IBARRA GARCÍA

APROBADO POR:

M.C. CARLOS GONZÁLEZ SÁNCHEZ
Presidente del Jurado y Director de Tesis

M.C. Evelio Martínez Martínez
Co-Director de Tesis

Adrián Enciso Almanza

L.C.C. ADRIAN ENCISO ALMANZA
SECRETARIO

M.C. ADÁN HIRALES CARBAJAL
1ER. VOCAL

Resumen de la Tesis de Javier Ibarra García presentada como requisito parcial para la obtención de la Licenciatura en ciencias Computacionales. Ensenada, Baja California, México. Septiembre de 2006.

DISEÑO E IMPLEMENTACION DE UN SISTEMA DE SEGURIDAD EN AMBIENTES LINUX/UNIX

Resumen aprobado por:

M.C. Carlos González Sánchez
Director de Tesis

M.C. Evelio Martínez Martínez
Co-Director de Tesis

El internet es hoy un medio de comunicación e intercambio de datos privados entre sistemas y personas alrededor del mundo, dado tal hecho existen personas que conocen el poder de la información y por ese motivo buscan la manera de obtenerla, aunque eso signifique un plagio o robo y en algunos países un delito.

El presente trabajo presenta los riesgos que hay en la red internacional y la manera en que los usuarios maliciosos operan para robar la información y las acciones que se pueden tomar cuando el sistema es comprometido. Por tal motivo se creo un sistema desarrollado sobre la plataforma Linux/Unix como sistema operativo, con el fin de proteger la información de los usuarios gracias a la detección de posibles intentos de intrusión.

También se muestra el análisis, diseño y desarrollo del sistema que previene posibles intrusiones (IDK). El lenguaje de programación utilizado es Java (Java2) por la arquitectura que permite la ejecución del sistema en diversas plataformas de hardware y software.

El modelo de desarrollo fue orientado a objetos (POO), el cual permite que el sistema pueda crecer y ser de mayor utilidad para los usuarios.

Palabras clave: internet, Linux/Unix, POO, intrusión.

Agradecimientos

A mis padres por la ayuda incondicional que siempre me han brindado, a mis familiares que han contribuido en mi formación personal y en especial a mis maestros y ex compañeros, los cuales contribuyeron a mi formación profesional.

Índice de contenido

Índice de figuras

Índice de imágenes

Índice de tablas

Capítulo 1: Introducción

La comunicación ha sido algo fundamental en la vida de los seres humanos, al inicio y hasta nuestros días ésta es de vital importancia, solamente que en la actualidad es mucho más compleja dado que podemos comunicarnos con otras personas ubicadas en puntos geográficos distintos al nuestro.

La primera comunicación eléctrica a través de una red fue la telegráfica, en 1844 por Samuel Morse. Comunicando así a finales del mismo año las ciudades de Washington, D.C. y Baltimore, MA. En 1876 Alexander Graham. Bell patenta lo que hoy en nuestros días conocemos como teléfono. Ambos inventos no se quedaron en una comunicación punto a punto, si no que se formo una red telefónica a lo largo de Estados Unidos (EUA), posteriormente a otros confines de la tierra.

Durante la segunda guerra mundial aparecen las primeras computadoras. Pero no fue hasta 1969 cuando se establece la primera red de computadoras entre 4 universidades de los EUA. Esta primera red, recibió el nombre de ARPANET (Advanced Research Projects Agency Network). Esta red es la antecesora de lo que hoy conocemos como Internet. [EVEL1996].

ARPANET no poseía gran seguridad. Los usuarios estaban restringidos y eran todos ellos de confianza. Al igual la información que pasaba por la red no era de carácter confidencial. [DGON2003].

Como se menciona anteriormente, la red ARPA evolucionó a lo que hoy es Internet, una red de redes, como algunos autores la definen, que comunica a casi todo el mundo. Pero es en su tamaño y complejidad donde inicia un problema. Internet es de acceso *público*, así que cualquier persona con una computadora y una conexión a internet puede ingresar a la supercarretera de la información. Las personas que hacen uso de internet, pueden usarlo para fines educativos, comerciales, entretenimiento, entre otras, pero lamentablemente existen personas que con el fin de obtener un beneficio propio (económico, de conocimientos o de otra índole). Usan la internet de una forma que podríamos llamar como *no ético*, por lo que es de suma importancia definir reglas de seguridad para evitar que éstos usuarios maliciosos provoquen estragos en los sistemas de cómputo, tales como computadoras personales o grandes servidores de empresas o instituciones educativas, entre otras.

Las redes proveen gran flexibilidad para el acceso de información o recursos compartidos. Sin embargo incrementa la exposición de información muy sensible y esto hace necesario implementar estrategias de control, para que esta información se mantenga integra, confidencial y disponible para los usuarios autorizados. [KILL2001].

1.1 Historia de los Sistemas de Detección de Intrusos

El solo hecho de escuchar *Sistema de Detección de Intrusos* nos puede traer a la mente un dispositivo de alarma, lo cual es totalmente cierto, pero este tipo de alarmas son para detectar intrusiones en nuestros sistemas de cómputo y evitar que estas intrusiones causen estragos en las redes de información.

1.1.1 Auditores

Los sistemas de auditorias según el "Libro Marrón" (llamado así por el color de su portada), titulado "A Guide to Understandig Audit in Trusted Systems" [NCSC1988]. Deben cumplir cinco mecanismos de auditoria, descritos a continuación:

- Permitir la revisión de patrones de acceso y el uso de mecanismos de protección de sistema.
- Permitir el descubrimiento de intentos internos y externos de burlar un mecanismo de protección.
- Permitir el descubrimiento de la transición de usuario cuando pasa por un nivel menor de privilegios a otro mayor.
- Permitir el bloqueo de los intentos de usuarios por saltarse los mecanismos de protección del sistema.

- Deberá funcionar como una garantía ante los usuarios de que toda la información que se recoja sobre ataques e intrusiones será suficiente para controlar los posibles daños ocasionados en el sistema.

A continuación se muestra el funcionamiento básico de un sistema de auditorias.

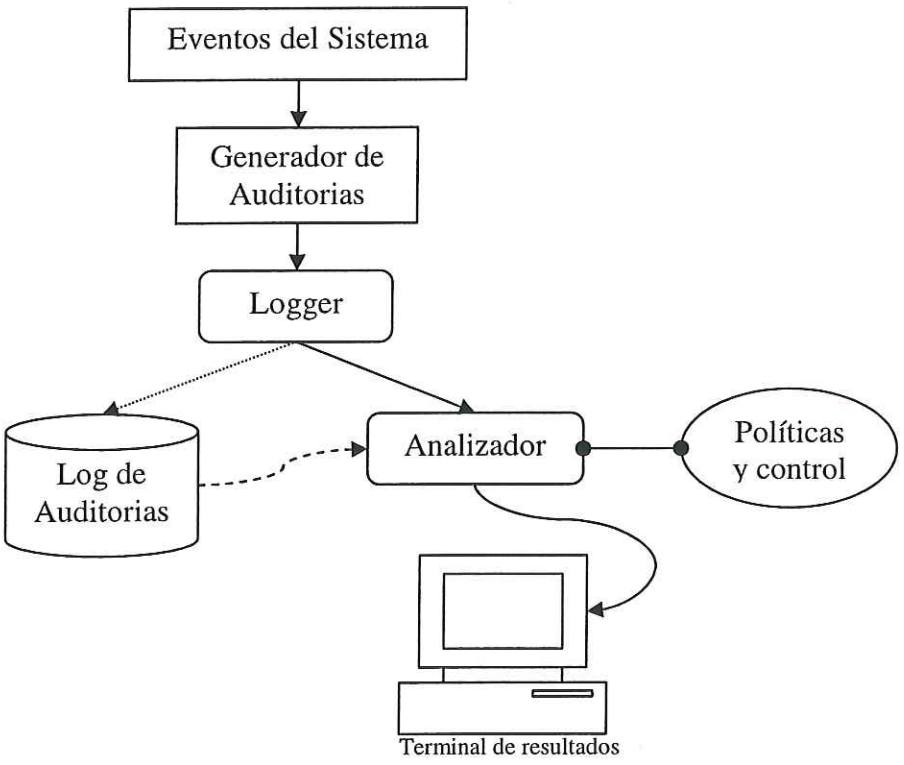

Figura 1-1. Sistema de Auditorias Básico [DGON2003].

1.1.2 Primeros Sistemas Detectores de Intrusos

Existe una gran variedad de Sistemas de Detección de Intrusos. Las diferencias de estos sistemas inician desde su manera de operar, así como el objetivo puede ser solo la protección o además de eso encontrar al responsable de los actos ilícitos para tomar acciones legales en contra de quienes resulten responsables de dichos actos.

Todo empezó en la década de los cincuenta, cuando la empresa telefónica estadounidense "Bell Telephone System" decidió automatizar el proceso de registro de eventos en sus conmutadores telefónicos. La computadora era capaz de almacenar cronológicamente los eventos más importantes que iban ocurriendo en sus entrañas. A esto se le llamó EDP (Electronic Data Process, o proceso electrónico de datos) y el cual inicio una pequeña revolución en el cada vez más pujante mundo de la informática. Antes de esto, todos los informes sobre los eventos acaecidos en una computadora, eran cuidadosamente registrados y entregados por esforzados técnicos en papel físico. Desde entonces todo sistema operativo cuenta con un sistema de almacenamiento de bitácoras (logs) que registra los eventos más importantes ocurridos en el sistema. Pero no fue hasta 1980 cuando James P. Anderson estudió el problema del análisis de bitácoras e ideó un mecanismo que automatizara la ardua tarea de revisión de registros. Su sistema se basaba en la comparación con estadísticas generadas a partir de la observación. Durante un período de desarrollo, se estudiaba el uso, por ejemplo, que el usuario hacía de una cuenta. A partir de esa observación, se creaban patrones estadísticos que definían

el comportamiento tipo del usuario. El "detector de intrusos" podía entonces detectar si se había "raptado" una cuenta a partir de la comparación de comportamientos cada vez que "alguien" tenía acceso a esa cuenta. Esta idea marcaría el desarrollo de los futuros sistemas de detección de intrusos [SANT2005].

James P. Anderson propuso un sistema de clasificación que distinguía entre ataques internos y externos, basado en si los usuarios tenían o no permisos de acceso a la computadora. Los objetivos del sistema eran los siguientes: [ANDE1972]

- Proporcionar suficiente información para que los encargados de seguridad localizaran el problema.
- Capaz de obtener datos de diferentes recursos del sistema.
- Para evitar los ataques internos tenía que detectar usos indebidos o fuera de lo normal por parte de los usuarios.
- Debía obtener la estrategia usada por el atacante para entrar en las cuentas.

IDES, El "Intrusion Detection Expert System" desarrollado por Dorothy Denning y Peter Neumann entre los años de 1984 y 1986, fue un modelo que definía un sistema de detección de intrusiones en tiempo real. [DENN1986]. Este sistema proponía una correspondencia entre actividad anómala o uso indebido, donde una actividad anómala es aquella rara o inusual dado un contexto estadístico. Usaba perfiles, así podía definir las acciones o eventos en el sistema. Este conjunto de elementos permitía crear

pautas de comportamiento de los usuarios para poder detectar posibles anomalías. IDES era un sistema híbrido ya que agregaba un nivel de seguridad mediante el uso de un sistema experto, el cual se basaba en reglas de seguridad que minimizaban los efectos de un intruso que intentara eludir el detector de anomalías.

El SRI International's Computer Science Lab se baso en IDES para crear NIDES (Next-Generation Intrusion Detection Expert System), El subsistema estadístico de NIDES emplea una amplia gama de medidas estadísticas multivariante para perfilar el comportamiento de usuarios o de otras entidades de cómputo. El análisis se basaba en perfiles, donde cada perfil es una cuenta de usuario con un valor estadístico establecido por el comportamiento del usuario en el sistema. NIDES produce un perfil separado de uso para cada usuario o entidad, también incluyeron un componente del análisis de la firma para identificar una actividad intrusa, a NIDES se le agregó un recurso gráfico basado en X/Motif para proporcionar la configuración y vigilar la operación de NIDES y para aumentar su utilidad. [SRII1990].

MIDAS "Multics Intrusion Detection and Alerting System", fue otro proyecto importante creado por el National Computer Security Center (NCSC). Se utilizó para monitorear el sistema NCSC Dockmaster; un Honeywall que corría uno de los sistemas operativos más seguros de aquel entonces, un Multics. Igualmente que IDES, MIDAS usaba un sistema híbrido para combinar la estadística de anomalías como reglas de seguridad de un sistema experto [SMAH1988]. Fue publicado en internet en 1989 y

monitoreo el Mainframe Dockmaster en el año de 1990. Ayudó a fortalecer la autenticación de usuarios, mejoró la seguridad de ataques externos y seguía bloqueando las intrusiones internas.

NADIR "Network Audit Director and Intrusion Reporter", fue uno de los sistemas con más éxito en los años ochenta, usaba técnicas muy similares a los sistemas de esa época, tales como IDES o MIDAS. Monitoreaba una red de aproximadamente 9,000 usuarios. La persona responsable del proyecto fue Kathleen Jackson. [KATH1991].

NSM "Network System Monitor", desarrollado en la Universidad de California para trabajar en una estación UNIX de Sun. NSM fue el primer Sistema de Detección de Intrusos que monitoreaba el tráfico de la red, usaba esos datos como principal fuente de información. Los sistemas anteriores se basaban en los eventos del sistema o las pulsaciones del teclado, el funcionamiento básico de NSM lo usan muchos sistemas de detección de intrusos basados en red, los pasos de NSM son los siguientes:

- Ponía el dispositivo de red en modo promiscuo, tal que monitoreaba todo el tráfico de la red, incluso el que no estaba dirigido a él.
- Capturaba los paquetes de la red.
- Identificaba el protocolo utilizado para poder extraer los datos necesarios (IP, ICMP, etc.).

- Usaba un enfoque basado en matrices para archivar y analizar las características de los datos, para hacer la búsqueda de variaciones estadísticas que permitieran descubrir un comportamiento anómalo como violación de reglas.

Los responsables del NSM fueron Karl Levitt, Todd Heberlein y Biswanath Mukherjee de la Universidad de California. [HEBR1995].

El esquema usado por NSM y el rápido crecimiento de las redes de computadoras provocó que se crearan nuevos modelos de Sistemas de Detección de Intrusos ahora no sólo basados en la Máquina si no también en Red.

DIDS "Distributed Intrusion Detection System", fue creado por grandes instituciones como el Centro de Soporte Criptológico de las Fuerzas Aéreas de EUA, El Laboratorio Nacional de Lawrence Livermore, la Universidad de California y los Laboratorios Haystack. Fue el primer sistema capaz de monitorear las violaciones e intrusiones a través de las redes. El principal responsable fue Steve Smaha.

DIDS fue el primer sistema capaz de relacionar los eventos que recibía para poder detectar una posible intrusión. Además, reunía la información de forma que era posible hacer un seguimiento del intruso y así poder identificarlo y perseguirlo para que pudiera ser castigado por la ley.

Por los años 90's aparecieron los primeros productos comerciales de Sistemas de Detección de Intrusos, uno de los productos más famosos de la época fue el "Computer Wath" creado por la empresa AT&T, el "Information Security Officer's Assistant" (ISOA) de Planning Research Corp. y el "Clyde VAX Audit" por Clyde Digital, entre otros.

1.2 Definiciones:

Es necesario definir una serie de conceptos, los cuales son de gran importancia para el propósito de este documento así como la comprensión del mismo.

1.2.1 Seguridad: Consiste en la protección de *activos*, por lo que se debe conocer esos activos y su valor. Esta es una observación muy general y por supuesto muy cierta en seguridad de computadoras. Pero no siempre se puede mantener seguros esos activos, por lo que es necesario tener en cuenta lo siguiente:

1.2.1.1 Prevención: Tomar medidas para anticipar que nuestros activos (equipo de computo) o recursos sean dañados.

1.2.1.2 Detección: Tomar medidas que permitan anticipar cuándo los activos han sido dañados, cómo fueron dañados y quién causo el daño.

1.2.1.3 Reacción: Tomar medidas que permitan recuperar los activos para recuperarse de un daño.

1.2.2 Seguridad en cómputo: Tomando en cuenta como los activos de la información pueden ser comprometidos. La definición mas frecuente propuesta propone tres aspectos. [GOLL1999].

1.2.2.1 Confidencialidad: Prevención de *accesos* no autorizados a la información.

1.2.2.2 Integridad: Prevención de *cambios* no autorizados de la información.

1.2.2.3 Disponibilidad: Prevención de *negación* no autorizada de información o de recursos.

1.2.3 Ataques: Los ataques son cualquier intento de poner en riesgo la confidencialidad, integridad o disponibilidad de la información y/o recursos.

1.2.3.1 Ataques a la confidencialidad: El propósito es obtener acceso a información confidencial a la que no se esta autorizado. Por ejemplo contraseñas, correos electrónicos, cuentas bancarias, información contable, etc.

1.2.3.2 Ataques a la integridad: No solo consiste en el acceso no autorizado a archivos de configuración o confidenciales, si no la modificación de los mismos para obtener algún beneficio de ellos.

1.2.3.3 Ataques a la disponibilidad: También conocidos como ataques de negación de servicio (DoS, por sus siglas en ingles). Por ejemplo negación al acceso a internet, correo electrónico, apagado de la computadora, negar que el usuario imprima documentos, etc.

1.2.4 Sistema Detector de Intrusos: Es una herramienta encargada de monitorear los eventos catalogados como intentos de intrusión.

Un intento de intrusión se puede definir como aquel que pone en riesgo la seguridad del sistema computacional, es decir, la *confidencialidad*, *integridad* e *integridad* de la información y los recursos. [BACE2001].

1.2.5 Intruso: Vamos a definir a un intruso como una amenaza, la cual puede afectar los aspectos de seguridad que ya hemos visto tales como confidencialidad, integridad y disponibilidad y va a estar representado en la siguiente tabla.

Amenaza	Preocupación	Riesgo
Usuario NO Autorizado **(curioso)** →	Confidencialidad → Integridad	Acceso no autorizado Observación no autorizada Monitoreo no autorizado Copias no autorizadas Robo
Usuario NO Autorizado **(malicioso)** →	Confidencialidad Integridad → Disponibilidad	Introducir, usar o producir datos falsos Modificar o reemplazar datos validos Mala representación o negación Uso erróneo de datos Falla al utilizar datos/recursos cuando son requeridos
Usuario Autorizado →	Integridad Disponibilidad →	Destruir, dañar o contaminar Negar, prolongar o retrasar uso o acceso
Desastre Natural →	Disponibilidad	

Tabla I. Amenaza/Preocupación/Riesgo [KILL2001].

Ya que conocemos un poco más acerca de la seguridad y los intrusos, es necesario que empecemos a ver como podemos evitar una intrusión o al menos detectarla para así poder tomar las medidas necesarias para que el daño no sea mayor al ya causado.

1.2.6 Diferencias entre los Sistemas de Detección de Intrusos

Anteriormente se mencionaron algunos Sistemas de Detección de Intrusos, los cuales eran distintos entre sí por su forma de evaluar y decidir si el sistema ha sido comprometido. Todos los sistemas de detección deben guardar los registros en lugares distintos a los del sistema para que el intruso no pueda borrar los registros y así el Administrador pueda detectar lo sucedido. Esto lo cumple cualquier detector de intrusos con un mínimo de calidad.

La información de las bitácoras nos ayuda a darnos cuenta de las intrusiones, no se debe perder, por que es la que nos ayuda a saber que sucedió en nuestro sistema cuando éste fue atacado por un intruso.

La clasificación de los sistemas de detección será de acuerdo a la localización de la información que se analiza para saber si hay un intento de intrusion.

- **Basados en máquinas**: Recogen los datos generados por una computadora. Los registros de eventos y las colas de auditorias pertenecen a este grupo.
- **Basados en múltiples máquinas:** Como el nombre lo dice, recoge información de dos o más máquinas. La lógica es la misma que la de los basados en máquina, pero es más complejo por que debe manejar la información de múltiples computadoras.
- **Basados en redes:** Capturan paquetes de la red. Por lo general los dispositivos de red se usan en modo promiscuo, por lo que el sistema se convierte en lo que se conoce como *sniffer* o rastreador.
- **Basados en aplicación:** Registran la actividad de una determinada aplicación. Por ejemplo los registros de un servidor ftp.
- **Basados en objetivos:** Estos difieren a los demás por que crean sus propios registros. Utilizan funciones de cifrado para detectar posibles alteraciones de sus objetivos y comparan sus resultados con las políticas. Este método es útil cuando la información no puede ser analizada de otra forma. Por ejemplo contraseñas.
- **Híbridos:** Combinan dos o más fuentes de distinto tipo, así se amplían las posibilidades de detección. Estas soluciones también se les llama integradas.

Pero esta es solo la localización de la información, todavía necesitamos analizarla y se va a clasificar de acuerdo al tipo de análisis. Los dos tipos de análisis son:

- **Detección de uso indebido:** Se comparan patrones de ataques conocidos con la información recogida buscando que existan coincidencias.
- **Detección de anomalías:** Se usan técnicas estadísticas que definen de forma aproximada lo que es el comportamiento usual o normal.

La Figura 1-2 muestra un esquema general de detección de anomalías y de usos indebidos:

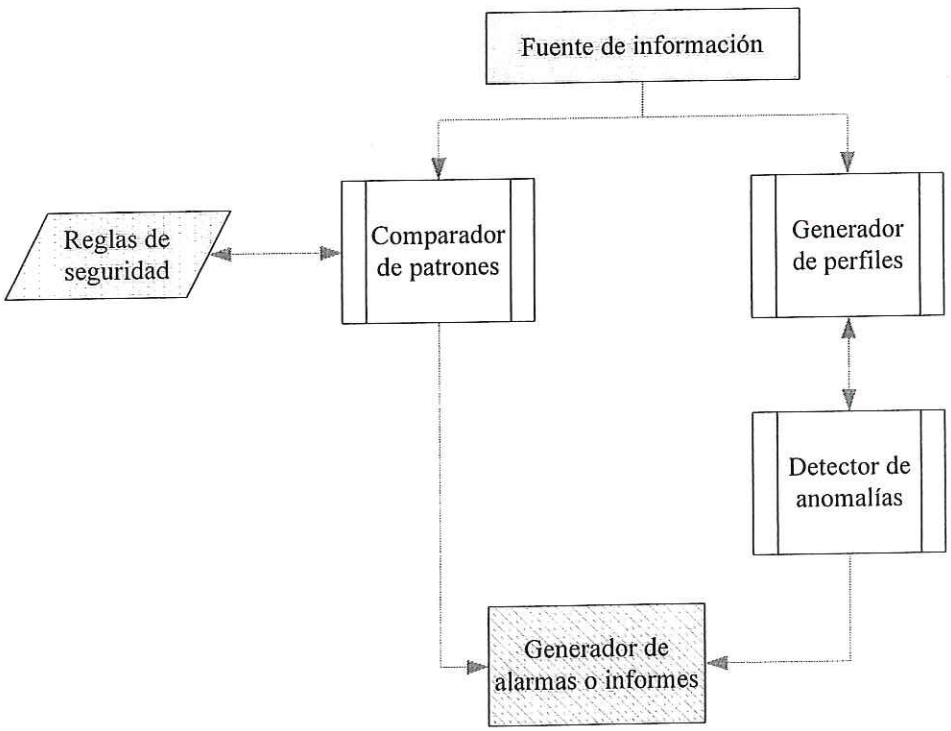

Figura 1-2. Esquema general de un sistema de Detección de Intrusiones [DGON2003].

La mayoría de los detectores son para detectar usos indebidos, anomalías o una mezcla de ambos. Aparte de estos, existen otros que son de *análisis de integridad*. Este

es el método que utilizan herramientas que revisan la integridad de los archivos, complementando a los sistemas de detección. Estas herramientas revisan si hay cambios en los archivos usando mecanismos de encriptación como funciones de hash.

Otro enfoque de los detectores toman en cuenta el tiempo, por lo que los tipos de análisis en *tiempo* son:

- **Por lotes:** Cada intervalo de tiempo se procesa una parte de los datos recibidos, enviando las posibles alarmas de intrusiones *después* de que hayan sucedido.
- **Tiempo real:** Los datos los examina en el tiempo en que son recibidos (o con un retardo mínimo). La aparición de estos análisis hizo posible las respuestas automáticas.

1.2.7 Tipos de respuestas de los Sistemas de Detección de Intrusiones

Existen dos tipos de respuestas de los sistemas de detección, estas son:

- **Respuestas Pasivas:** El detector no toma decisiones o acciones que cambien el curso del ataque. Solo envía o registra la alarma correspondiente.

- **Respuestas Activas:** Estos no sólo generan una alarma, si no que reacciona tomando una acción para terminar el ataque. Por ejemplo terminando la sesión del posible intruso o bloquear sus acciones.

1.2.8 Clasificación general de los Sistemas de Detección de Intrusiones

Con las diferencias entre los sistemas de detección de intrusos anteriormente mencionados en los puntos 1.2.6 y 1.2.7 podemos clasificarlos de la siguiente manera para una mejor comprensión de los mismos:

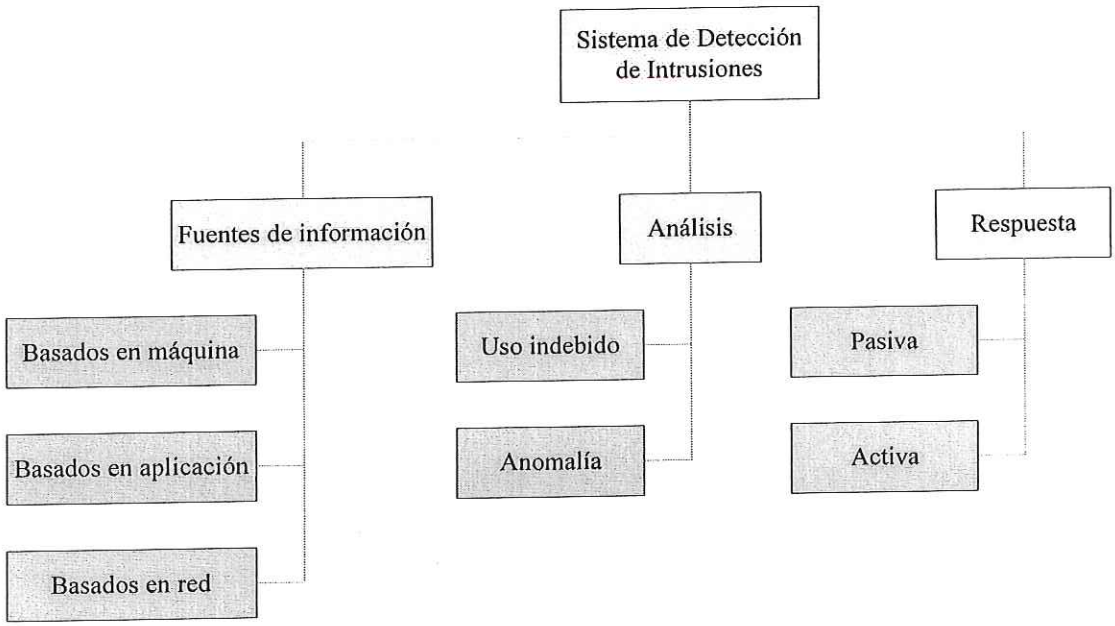

Figura 1-3. Tipos principales de Sistemas de Detección de Intrusiones [DGON2003].

1.2.9 Administración de los Sistemas de Detección de Intrusos

La manera en que se administra la detección de intrusos es otro elemento importante que se debe tomar en cuenta. Existen dos enfoques según el monitoreo de múltiples computadoras o redes.

La centralización y la integración con herramientas para la gestión de redes, la centralización trata de concentrar las funciones del control en un nodo, el cual dirige a los demás elementos de detección de intrusiones. Otra forma de solucionar la administración centralizada es hacer que forme parte de las funciones de gestión de redes. Muchos productos comerciales de detección de intrusos generan mensajes SNMP (Simple Network Management Protocol) para la herramienta de captura de gestión de redes.

1.3 Objetivo

El objetivo de este trabajo es desarrollar una herramienta que ayude a los administradores de sistemas de cómputo a fortalecer la seguridad de los mismos. Para lograr el objetivo es necesario realizar un análisis, diseño y finalmente en base a los anteriores realizar la implementación de la herramienta. Este sistema protegerá, en medida de lo posible, de posibles ataques por usuarios maliciosos de la red y detectará accesos inusuales en el sistema para alertar al administrador. El sistema será

desarrollado en el lenguaje de programación Java, un lenguaje multiplataformas que permite ejecutar aplicaciones en diferentes sistemas operativos, es un lenguaje gratuito y se puede adquirir fácilmente en el internet. Se utilizará la metodología de desarrollo en cascada, la cual es una metodología de desarrollo de software, la cual se explica con más detalle en el capítulo 4 de este documento.

1.4 Justificación

Dado que muchos de los sistemas computacionales se conectan a la red publica de internet, por la cual se transmite información privada, debemos hacer lo posible como usuarios responsables para que nuestra información y la de las personas que nos la confían para protegerla, es justo y necesario hacer el mayor esfuerzo para lograr ese objetivo, es por ello por lo que este trabajo ha sido diseñado.

Existe una gran variedad de programas utilizados para atacar los sistemas de cómputo. También existen en el mercado muchas aplicaciones para la protección de la información pero los precios pueden llegar a ser muy elevados, eso dependiendo del proveedor y probablemente de la calidad del sistema.

La finalidad es crear una herramienta de uso libre (gratuita), capaz de proteger la los sistemas de posibles ataques y a su vez la información.

1.5 Organización del trabajo

Este documento consta de 8 capítulos cuyo contenido se organiza de la siguiente manera:

En el capítulo 2 se presenta información sobre cortafuegos, protocolos y una revisión a otras herramientas de seguridad. El capítulo 3 nos enseña como es que los hackers entran y hacen mal uso de los sistemas de cómputo y como podemos protegernos de ellos. Posteriormente en el capítulo 4 nos enfocaremos la metodología que se siguió para el desarrollo del sistema IntrudEns Detection Killer (IDK). El capítulo 5 esta dedicado al diseño de IDK, donde muestran los módulos del sistema y sus respectivas soluciones. Se describe sobre el lenguaje de programación utilizado.

El capítulo 6 presenta los algoritmos del sistema y las interfaces de usuario y su funcionamiento. En el capítulo 7 se presentan las pruebas de usabilidad que se desarrollaron para demostrar el funcionamiento del sistema. Finalmente en el capítulo 8 se presentan las conclusiones del trabajo, las recomendaciones y el trabajo futuro.

Capítulo 2: Cortafuegos y filtrado de paquetes

En la figura 2-1 se aprecia una muralla que divide el internet de una red de área local. Esa muralla es una computadora conectada al internet por medio de cualquier tipo de conexión (E1, VPN, ADSL o MODEM), y funciona como dispositivo de seguridad. A esta computadora se le llama o denomina como *cortafuegos*.

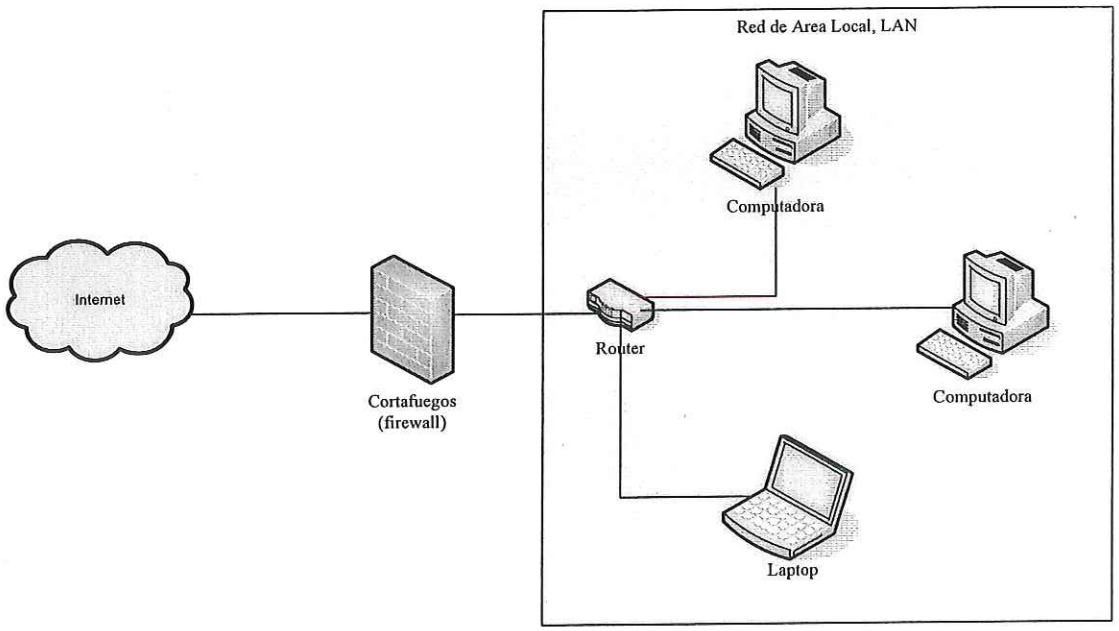

Figura 2-1. Cortafuegos entre internet y una red de área local.

La definición de *cortafuegos* depende de su implementación y su propósito. Pero se definirá cortafuegos como aquél dispositivo de hardware o software que tiene conexión directa al exterior y su propósito es evitar que se pueda afectar y/o comprometer el funcionamiento, información, recursos, etc. de la red de área local. La

red podría ser por ejemplo de tipo gubernamental, universitaria, empresarial o simplemente una computadora personal. [ZIEG2000].

2.1 Servicios

Los servicios basados en red son programas que pueden ser accedidos por otros equipos de la red. Los puertos de los servicios identifican los procesos a los que corresponden, por ejemplo el puerto 80 corresponde al nombre del servicio WWW del protocolo HTTP (HyperText Transfer Protocol, Protocolo de transferencia de hipertexto), el cual es el método más común de intercambio de información en el internet, mediante el cual se transfieren las páginas de un servidor a un cliente (navegador). Los puertos son números que identifican los diferentes servicios y están numerados desde 0 hasta el 65,535. (2^{16}).

Existe una numeración reservada para los procesos que solo puede ejecutar el súper usuario (root), los puertos van de 1 a 1,023. Estas asignaciones las coordina la Autoridad de asignación de números de internet (IANA, Internet Assigned Numbers Authoriry), para establecer estándares y así evitar conflictos.

Como ya se mencionó, estos puertos son las puertas de comunicación entre las computadoras, para comprenderlo mejor, puede ser ejemplificado como una casa con 65,536 puertas por las cuales puede entrar o salir, entonces es de mucha importancia que

se conozca quien entra o sale, cuando y como lo hace, etc. A su vez, por seguridad de nuestros sistemas, debemos cerrar las puertas que no se necesitan y por las cuales podría entrar algún intruso, virus, gusano, troyano o cualquier evento que ponga en riesgo el sistema.

2.2 Mensajes de red IP

El protocolo de internet IP define la estructura del mensaje que las computadoras intercambian durante su comunicación. El mecanismo de cortafuegos IP (IPFW) que se incluye en Linux es compatible con tres tipos de mensajes IP: ICMP, UDP y TCP. Los paquetes ICMP (Protocolo de mensajes de control de Internet) contienen información sobre la comunicación entre dos equipos finales. Los paquetes UDP transportan datos en la capa de transporte entre dos programas basados en red, sin ninguna garantía de éxito en cuanto a la entrega o el orden de la entrega de paquetes. Los paquetes TCP (Protocolo de Control de Transmisión) se transmiten sobre la capa de transporte TCP entre dos programas basados en red, este protocolo contiene información adicional para mantener una conexión confiable y activa.

2.2.1 Mensajes IP: ICMP

El encabezado de estos paquetes contiene las direcciones IP origen y destino, un identificador del protocolo ICMP y un tipo de mensaje ICMP. Los mensajes ICMP

indican si el paquete es: comando, respuesta a un comando, información de estado o una condición de error. Estos paquetes no contienen información sobre los puertos origen y destino, a su vez, no se envían entre programas, si no entre los equipos origen y destino.

2.2.2 Mensajes IP: UDP

El encabezado de los paquetes UDP contiene las direcciones IP origen y destino, el tipo de protocolo UDP y los números de puerto de servicio origen y destino. Es un protocolo sin estado, en otras palabras no es confiable, un programa envía un paquete UDP y este puede ser recibido o no y el receptor puede contestarlo o no. Es decir, no hay "acuse de recibido" del paquete.

Los paquetes que utilizan UDP como protocolo de transporte, se dice que son orientados a no conexión.

2.2.3 Mensajes IP: TCP

El encabezado de los paquetes TCP contiene las direcciones IP origen y destino, el tipo de mensaje de protocolo TCP, los puertos de servicio destino y los números de secuencia y de confirmación, así como los indicadores de control que se usan para que la comunicación sea confiable y activa en ambos sentidos.

Los paquetes que utilizan TCP como protocolo de transporte, se dice que son orientados a conexión.

2.3 Filtrado de paquetes

Los protocolos con filtrado de paquetes operan sobre la capa de transporte y red del modelo de referencia OSI. Protegen al sistema con decisiones de enrutamiento después de haber filtrado los paquetes en base a la información del encabezado de dicho paquete.

2.3.1 Cortafuegos de filtrado de paquetes

El cortafuegos con filtrado de paquetes posee una lista de reglas, las cuales aceptan o niegan el paquete. Dichas reglas se basan en la tarjeta de red, en los puertos de servicio UDP y TCP, los indicadores de conexión TCP, los mensaje ICMP a nivel de red y si el paquete es entrante o saliente de la máquina.

Las reglas son definidas como cadenas, cada paquete se compara con las reglas definidas y dependiendo de las políticas establecidas se definirá el destino de los paquetes.

Cada cadena del cortafuegos tiene una instrucción y una serie de acciones a realizar de acuerdo al mensaje especifico. Existen dos perspectivas para la configuración de un cortafuegos, dependiendo el nivel de paranoia del administrador del sistema.

- Negar todos los paquetes y permitir que pasen los paquetes seleccionados y establecidos de forma explicita.
- Aceptar todos los paquetes y negar que pasen los paquetes seleccionados y establecidos de forma explicita.

La primera configuración permite que sólo se permita la comunicación a lo que se necesita en cada momento y cerrando la mayoría de las posibilidades para un intento de intrusion o ataque. La segunda configuración en cambio es más sencilla, pero puede ser más propensa a errores por descuidos de configuración del administrador, por lo que el sistema está más expuesto a recibir un ataque. A continuación se muestra la lógica en el filtrado de paquetes para ambas políticas.

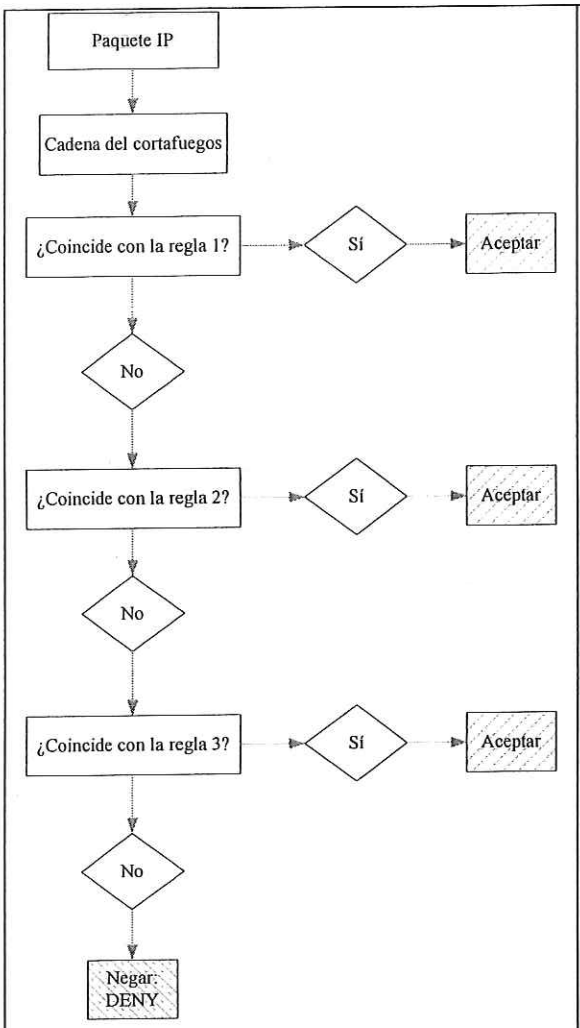

Figura 2-2. Política de negación de paquetes

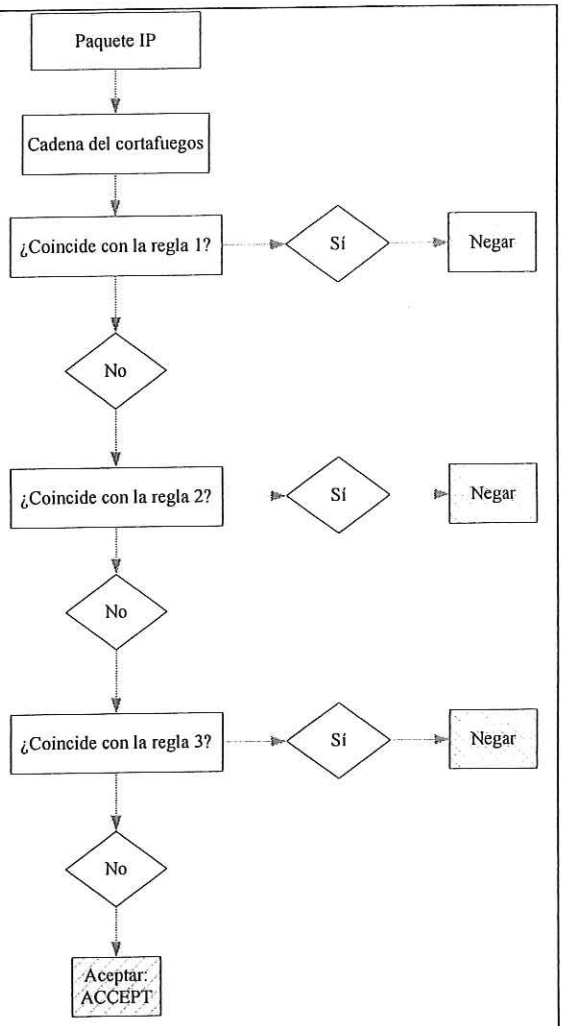

Figura 2-3. Política de Aceptación de paquetes

Otra posibilidad es el rechazo de los paquetes, suena muy parecido a la negación de los mismos, pero no lo es, el rechazo consiste en negar el paquete y responderle al remitente un error ICMP. Por el contrario cuando se niega un paquete, simplemente se descarta y no se le da una notificación al remitente. Pero la mejor elección es la negación de paquetes, por las siguientes razones:

- Enviar una respuesta de error duplica el tráfico de red.
- La mayoría de esos paquetes son malévolos.
- Cualquier paquete al que se responda se puede usar en un ataque de negación de servicio.
- Cualquier respuesta, como un mensaje de error, ofrece información potencialmente útil para el atacante.

El siguiente es un diagrama que ejemplifica el funcionamiento general sobre el rechazo o negación de paquetes:

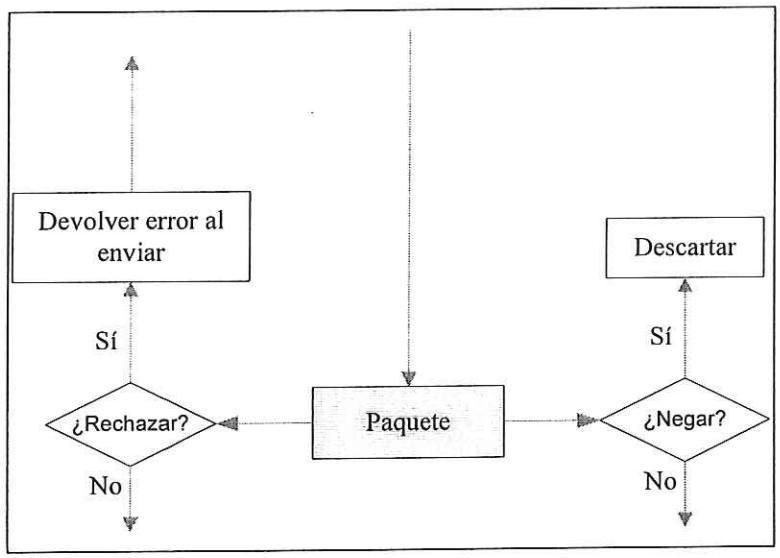

Figura 2-4. Diagrama de flujo del rechazo o negación de un paquete.

2.4 Aplicaciones de cortafuegos

2.4.1 TCPWrapper

Es una aplicación incluida en los sistemas Unix, ofrece un control de acceso a los servicios arrancados por el demonio *inetd*, el cuál inicia los servicios del sistema, escuchando y creando conexiones. Pero para los sistemas Linux como SuSe, Red Hat, entre otros, el demonio es una versión extendida del *inetd* llamado *xinetd*, en este último se configuran los servicios uno por uno dentro de la trayectoria */etc/xinet.d/* ahí donde se encuentra la configuración de cada servicio, como por ejemplo *telnet*, *ftp*, *secure shell* entre otros.

Ventajas:

- Guarda el historial de las conexiones.
- Control de acceso a la red

Archivos:

- /etc/hosts.allow Reglas que permiten acceso.
- /etc/hosts.deny Reglas que niegan el acceso.

Si estos archivos están vacíos, permitirá las conexiones por dicho servicio, siempre y cuando este servicio este activo en el sistema.

2.4.2 ipfwadm/ipchains/iptables

Al inicio ***ipfwadm*** era la herramienta proporcionada por Linux para la implementación de políticas de filtrado de paquetes y debido a sus limitaciones (por ejemplo, sólo puede manejar los protocolos TCP, UDP o ICMP) ***ipfwadm*** fue reescrito para convertirse en ***ipchains*** en 1998. Esta nueva herramienta introdujo bastantes mejoras con respecto a la anterior, pero seguía careciendo de algo fundamental, de ésta forma, a mediados de 1999 ***ipchains*** fue sustituido por ***iptables***, que de nuevo introducía importantes mejoras con respecto a su predecesor. Ahora ***iptables*** ofrece un sistema de NAT (*Network Address Translation*) mucho más avanzado, incorpora mejoras en el filtrado e inspección de paquetes, y presenta un subsistema de *log* mucho más depurado que ***ipchains***. Por tanto, ***iptables*** es en la actualidad el software de cortafuegos en Linux.

Históricamente, todos los sistemas de cortafuegos nativos de Linux han sido orientados a comando: esto significa, muy por encima, que no leen su configuración de un determinado fichero, por ejemplo durante el arranque de la máquina, sino que ese archivo de arranque ha de ser un *script* donde, línea a línea, se definan los comandos a ejecutar para implantar la política de seguridad deseada; esta es una importante diferencia con respecto a otros cortafuegos orientados a archivo: en éstos la política se define en un archivo ASCII con una cierta sintaxis, que la aplicación interpreta y se carga en el sistema.

La sintaxis de *iptables* (o la de ipchains, bastante similar) puede llegar a resultar muy compleja si se invoca al sistema de filtrado desde línea de órdenes; pero existen diferentes interfaces para el administrador, algunos tan cómodos e intuitivos capaces de presentar las políticas de una forma gráfica basada en objetos y de transformar después esas políticas en *scripts* con las órdenes de iptables o ipchains equivalentes; iptables o ipchains son herramientas flexibles, potentes e, igual de importante, gratuitas, que funcionan sobre un sistema operativo también gratuito. [VILL2002].

2.5 Ventajas del filtrado de paquetes

El filtrado de paquetes tiene un gran número de ventajas. [ZWIC2000]. Estas ventajas, así como sus desventajas se describen a continuación.

2.5.1 Un enrutador como blindaje puede proteger una red entera

Una de las ventajas del filtrado de paquetes por medio de un enrutador como blindaje entre la red interna y el exterior es, que si se ubica estratégicamente, puede proteger una red entera, así se gana una gran seguridad en la red, haciendo que el paquete se filtre por medio del enrutador.

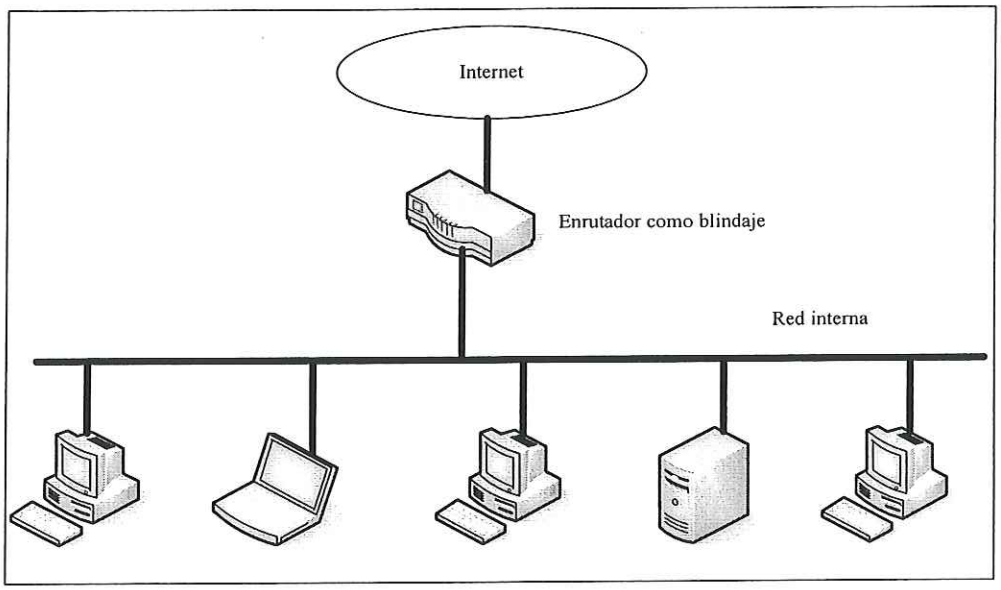

Figura 2-5. Usando el enrutador como blindaje para filtrar paquetes.

2.5.2 La filtración simple del paquete es extremadamente eficiente

La filtración de paquetes pone atención en algunas cabeceras del paquete, por lo que puede hacerse con gastos de procesamiento muy bajos. En cambio el usar un Proxy Server como cortafuegos es más lento por que también implica conexiones con otros programas.

2.5.3 La filtración del paquete está extensivamente disponible

Las capacidades de filtración del paquete están disponibles en la red internet ya sean en hardware o software. Muchos productos comerciales de enrutadores incluyen capacidades para la filtración de paquetes.

2.6 Desventajas de la filtración de paquetes

Aunque la filtración de paquetes proporciona muchas ventajas, hay algunas desventajas al usar el paquete que se filtra también. [ZWIC2000].

2.6.1 Las herramientas de filtración actuales no son perfectas

A pesar de la extensa disponibilidad de fuentes de hardware o software para el filtrado de paquetes, aún no es una herramienta perfecta, las principales limitaciones, son las siguientes:

- Las reglas de filtrado tienden a ser duras de configurar, por lo que requieren de un buen conocimiento del tema para poder hacer una correcta administración.
- Una vez configuradas, tienden a ser difíciles de probar.
- Las capacidades de filtrado de muchos productos son incompletas, por lo que algunas opciones de configuración podrían ser complejas o imposibles.
- Los conjuntos de filtración del paquete puede tener fallos de funcionamiento.

2.6.2 La filtración del paquete reduce el desempeño del enrutador

La filtración de paquetes en el ruteador agrega una carga significativa en él, debido a que el filtrado es un proceso que toma tiempo, hablando en términos de

milisegundos. Éste, agregado al tiempo de direccionamiento, ocasiona que se aumente el tiempo de procesamiento general del enrutador, ocasionando retardos del paquete al pasar por el éste.

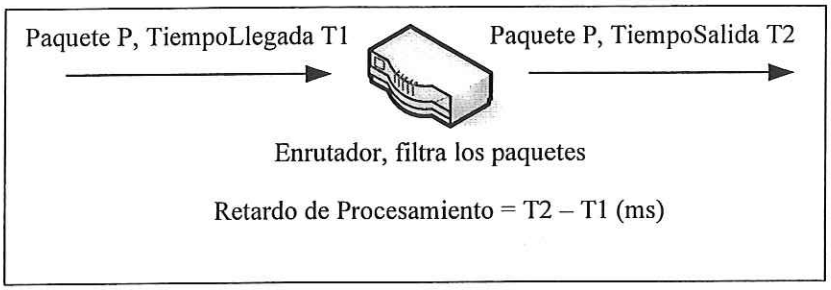

Figura 2-6. Muestra el retardo ocasionado en el enrutador por el procesamiento de filtrado.

2.6.3 Algunas políticas no se pueden cumplir con la filtración de paquetes por medio de enrutadores

La información disponible en el enrutador no permite que el usuario especifique algunas reglas que pudiera querer agregar. Por ejemplo, los paquetes dicen quien es el que lo envía, pero no el programa que necesita ese paquete, por lo que si se crean reglas de muy alto nivel podría bloquear paquetes que se necesitan para alguna aplicación. Entonces se deben crear reglas mas específicas y algunos ruteadores no lo permitirían, de acuerdo a las limitaciones que este tuviera.

Capítulo 3: Ataques y procedimientos de protección

En el capítulo anterior se mencionaron conceptos tales como *servicios* y los tipos de mensajes por IP: ICMP, UDP y TCP. De tal manera serán utilizados para la mejor comprensión del este capítulo.

Los hackers habitualmente explotan las vulnerabilidades de los servicios, principalmente los protocolos TCP/IP. Los servicios basados en este protocolo más comúnmente explotados son Telnet, FTP, TFTP, SMTP, HTTP, SSH, etc. Cada servicio tiene su "Talón de Aquiles" muy particular, los cuales atentan contra la seguridad de nuestro sistema. Cabe mencionar que en cada actualización de estos servicios se intenta arreglar dichos defectos para minimizar los riesgos de intrusion al sistema. [RIVA2003].

3.1 Ataques

3.1.1 Ataques basados en detalles de bajo nivel de los servicios

Al inicio del capítulo 2 se hace una descripción de los servicios y/o protocolos. Estos poseen errores de programación y provocan agujeros en la seguridad de los sistemas. Cada ataque realizado a esos protocolos suele tener el nombre del autor que logró penetrar por esos agujeros. Este tipo de ataques son cada día mayor más sin embargo no es una tarea sencilla. [ZWIC2000].

3.1.2 Escaneo de puertos

El escaneo de puertos es una estrategia para buscar servicios o puertos abiertos en una máquina. El escaneo de puertos es muy sencillo de detectar, por lo que los atacantes buscan métodos para disfrazar los escaneos. Por ejemplo, muchas máquinas no registran conexiones hasta que se hacen completamente, así que un atacante puede enviar un paquete inicial, enviando un SYN (paquete de sincronización) pero ningún ACK (acuse de recibido), entonces recibe la respuesta, otro SYN si el acceso está abierto o un RST (reset, restauración). Aún que este evento no quede registrado, puede tener otros efectos desafortunados como negación de servicios. [ZWIC2000].

3.1.3 IP Spoofing

En el IP Spoofing un atacante envía los paquetes con un direccionamiento incorrecto de la fuente u origen, cuando esto pasa, la contestación será enviada obviamente al origen incorrecto, no al atacante. Existen tres tipos de IP Spoofing [ZWIC2000]:

- El atacante puede interceptar la contestación
- El atacante necesita ver la contestación
- El atacante no desea la contestación

3.1.3.1 El atacante puede interceptar la contestación

Si un atacante está en alguna parte de la red, entre la victima y la fuente falsa, el atacante puede ver la contestación y continuar una conversación indefinidamente. Ésta es la base de los ataques de secuestro (Hijacking).

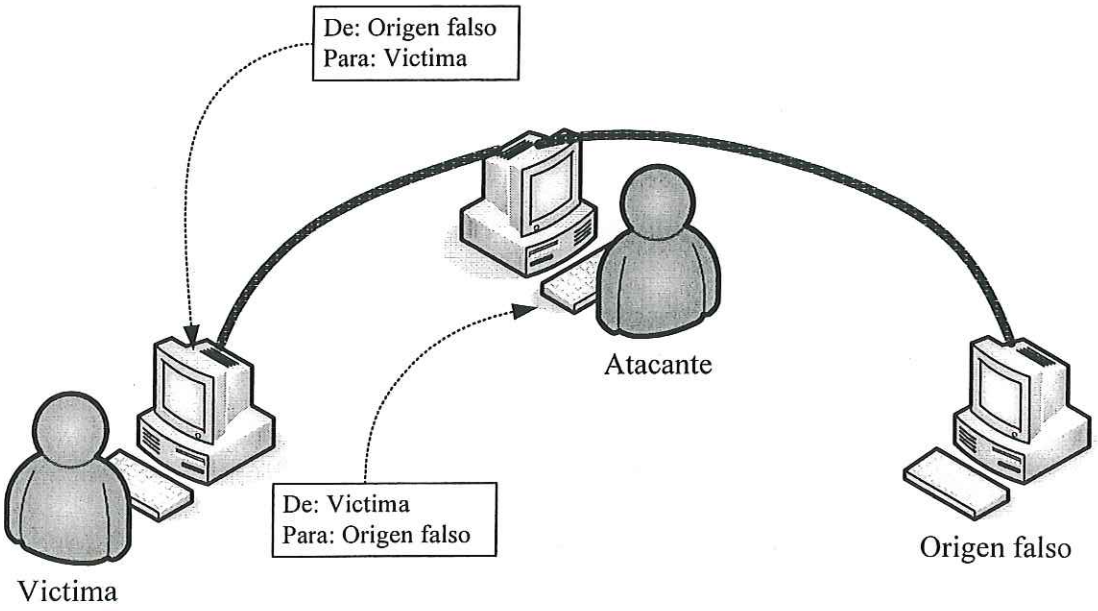

Figura 3-1. El atacante intercepta las respuestas a la maquina falsa.

3.1.3.2 El atacante no necesita ver la contestación

Un atacante no cuida cuál es la contestación. Si el atacante hace una negación de servicio, la máquina atacada no va probablemente a contestar de todos modos. Incluso si no es de este tipo, el atacante puede poder realizar un cambio deseado sin necesitar ver la respuesta.

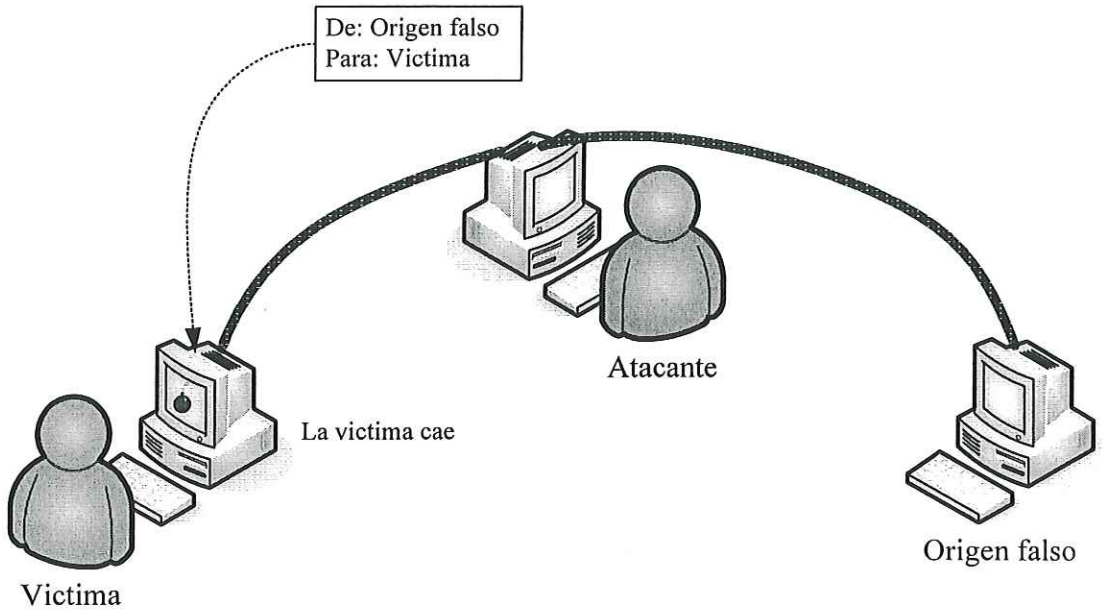

Figura 3-2. El atacante usa los paquetes del origen falso para negar el servicio.

3.1.3.3 El atacante no desea la contestación

En este caso, se confía que la contestación va en alguna parte. Este ataque tiene múltiples variantes usando diferentes protocolos y métodos para multiplicar las contestaciones. Los protocolos más comunes son la generación de eco del ICMP y el servicio de la generación de eco basado en UDP. El método más común de multiplicar las contestaciones es utilizar un direccionamiento de la difusión como el direccionamiento de la fuente. A continuación un par de escenarios para la mejor comprensión de esto.

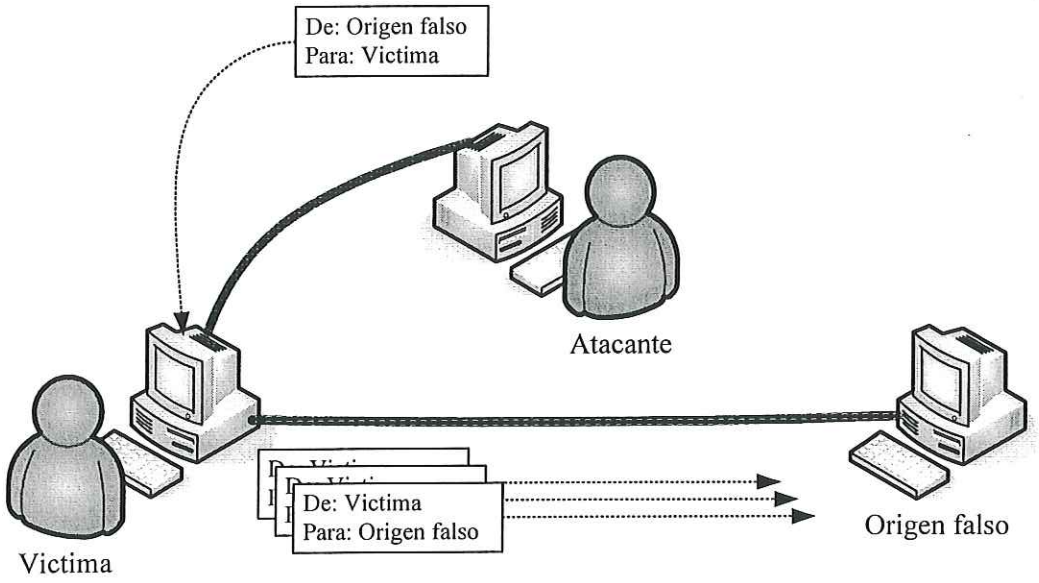

Figura 3-3. Atacante usa paquetes para atacar a un tercero.

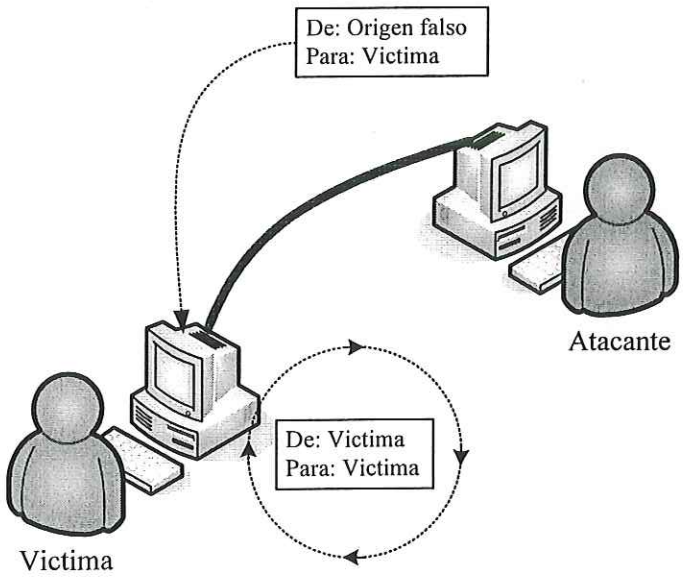

Figura 3-4. Atacante usa un ciclo para enviar los paquetes a la propia victima.

3.2 Cómo actúa un Hacker

A continuación se muestra una serie de pasos para lograr la intrusión a un sistema computacional. [RIVA2003].

- **Obtención de la información del equipo a atacar:** Se necesita recopilar cierta información del sistema que se pretende atacar, para elegir la mejor estrategia de intrusión.
- **Intrusión al sistema:** Existen dos estrategias para introducirse al sistema, la primera consiste en entrar directamente al sistema sin poseer una cuenta de usuario, la segunda consiste en extraer el archivo que posee las contraseñas del sistema y extraer las claves con algún programa.
- **Obtención de la cuenta de administrador:** Consiste en conseguir los privilegios más altos en el sistema para poder hacer lo que sea dentro de él. Se pueden ejecutar programas que monitoreen la red interna y así poder conseguir cuentas en otros sistemas del dominio al que se ha penetrado.
- **Mantener los privilegios de administrador:** La manera habitual de mantener los privilegios consiste en copiar un shell (consola de comandos) que posea los privilegios que se buscan en la cuenta de un usuario normal.
- **Borrar las huellas:** Toda la actividad realizada anteriormente queda registrada en lo que se conoce como bitácoras, por lo que se borran los registros de lo hecho para que el administrador no se de cuenta de la intrusion.

3.3 Aplicaciones para detección de intrusos

Existe un gran número de aplicaciones diseñadas para la detección de intrusos, a continuación una pequeña lista de aplicaciones que ayudan a los administradores para la detección de intrusos. [RIVA2003].

Permiten la detección de intrusos	**AAFID**, la información se ubica en: http://www.cerias.purdue.edu/research/aafid/.
Permiten la detección de intrusos en la red	**Hummer Project**, http://www.mrc.uidaho.edu/~hummer/index.htm. **Snort**, http://www.snort.org.
Detecta escaneos en los puertos	**tcplogd**, http://www.tigerteam.net/linuxgroup/tcplogd/. **scanlogd**, http://www.openwall.com/scanlogd/

Tabla II. Aplicaciones para detección de intrusos.

3.4 Que hacer cuando se detecta a un intruso

Cuando se detecta un intruso se debe realizar una búsqueda de software instalado con un fin malicioso en la red o en los sistemas afectados, cuado se localizan los equipos afectados, se debe hacer lo siguiente. [RIVA2003]:

a) Desconectarlos de la red.

b) Realizar una inspección detallada.

c) Realizar un informe sobre las fallas de la seguridad y sus posibles soluciones.

d) Realizar una inspección detallada sin la desconexión de los sistemas, esto con el propósito de espiar al intruso.

Se debe buscar que nivel de confidencialidad ha afectado, que contraseñas ha conseguido y analizar que pretende el intruso.

Existe en el mercado algunos programas que buscan la ruta que ha tomado el intruso para entrar a nuestros sistemas, por lo que es posible saber si se busca lo suficiente a quien ilegalmente penetro los sistemas, pero eso depende mucho de que tipo de sistemas ha penetrado, ciertamente si se trata de una organización gubernamental o empresa con los suficientes recursos, ganas y tiempo para encontrar a los culpables de la intrusion, lo harán para poder tomar acciones legales en contra de ellos siempre y cuando las leyes que los respaldan tengan castigos severos en contra de los que resulten responsables. Sin embargo, si se tratase de una institución de tipo educativa u otra sin los suficientes recursos para la búsqueda de los culpables, estos actos quedarían impunes.

Capítulo 4: Metodología de desarrollo

En capítulos anteriores se ven los fundamentos sobre este trabajo, en este capítulo se presenta la metodología utilizada para el diseño y desarrollo del sistema IntrudEns Detection Killer (IDK).

4.1 Metodología de desarrollo del sistema IDK

La metodología utilizada fue desarrollo en cascada, esta metodología ordena rigurosamente las etapas del desarrollo de software. Esta metodología es exitosa si todas las etapas se llevan a cabo sin errores, en el caso contrario obliga a retroceder en las etapas para corregirlos.

4.2 Desarrollo en cascada

El desarrollo en cascada parte de no tener absolutamente nada hasta la creación de un programa de software que cumple con lo necesario y soluciona el problema por el cual fue creado. El ciclo del desarrollo de software con esta metodología se ve en la figura 4-1.

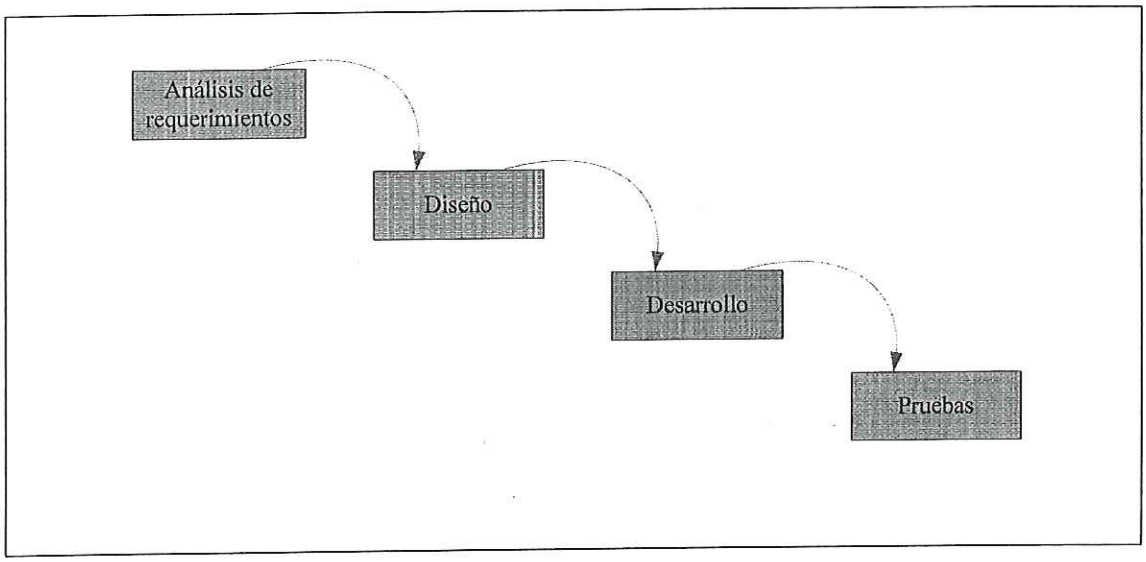

Figura 4-1 Ciclo de vida del desarrollo de software.

- **Análisis de requerimientos:** Esta etapa consta de la recopilación de información para definir el sistema a desarrollar.
- **Diseño:** Aquí se realizan esquemas que facilitan la comprensión de cómo funcionará el sistema que se pretende desarrollar.
- **Desarrollo:** Consta de la creación de código en el lenguaje de programación definido, la programación se basa en la etapa anterior que es el diseño.
- **Pruebas:** Las pruebas se encargan de evaluar el comportamiento del sistema para verificar que cumpla todos los requerimientos establecidos al inicio.

4.2.1 Análisis

En base a lo estudiado en capítulos anteriores y teniendo en cuenta que ésta es la primera etapa del desarrollo de software, éste se llevó a cabo de la siguiente manera:

a) Identificar características de sistemas existentes de seguridad en cómputo:

Para empezar, el primer paso fue la búsqueda de herramientas de seguridad existentes en el mercado, encontrar sus características y deficiencias de los mismos, esta información se utiliza en la etapa de diseño.

b) Evaluación de factibilidad del sistema:

Se debe evaluar que tan factible es la realización del sistema, se deben contemplar las implicaciones tanto económicas, humanas, tecnológicas, etc. Así como la dificultad que implica la realización de dicho sistema y el tiempo que llevará su construcción.

c) Análisis técnico:

Aquí se crea un modelo a partir de las observaciones ya realizadas sobre los sistemas comerciales existentes para crear un sistema que sea distinto de ellos.

Las etapas de diseño, desarrollo y pruebas se explicaran con detalle en los siguientes capítulos.

4.3 Lenguaje Unificado de Modelado (UML)

La notación UML se deriva y unifica las tres metodologías de análisis y diseño orientados a objetos más extendidas.

- Metodología de **Grady Booch** para la descripción de conjuntos de objetos y sus relaciones.
- Técnica de modelado orientada a objetos de **James Rumbaugh** (OMT: Object-Modeling Technique).
- Aproximación de **Ivar Jacobson** (OOSE: Object Oriented Software Engineering) mediante la metodología de casos de uso.

El desarrollo de UML comenzó a finales de 1994 cuando *Grady Booch* y *Jim Rumbaugh* de *Rational Software Corporation* empezaron a unificar sus métodos. A finales de 1995, *Ivar Jacobson* y su compañía *Objectory* se incorporaron a *Rational* en su unificación, aportando el método OOSE.

De las tres metodologías de partida, las de *Booch* y *Rumbaugh* pueden ser descritas como orientadas a objetos, ya que sus aproximaciones se enfocan hacia el modelado de los objetos que componen el sistema, su relación y colaboración. Por otro lado, la metodología de *Jacobson* es más centrada a usuario, ya que todo en su método se deriva de los escenarios de uso. UML se ha ido fomentando y aceptando como estándar desde el OMG, que es también el origen de CORBA, el estándar líder en la

industria para la programación de objetos distribuidos. En 1997 UML 1.1 fue aprobada por la OMG convirtiéndose en la notación estándar de facto para el análisis y el diseño orientado a objetos.

UML es el primer método en publicar un modelo en su propia notación, incluyendo la notación para la mayoría de la información de requisitos, análisis y diseño. Se trata pues de un modelo auto-referencial (cualquier lenguaje de modelado de propósito general debería ser capaz de modelarse a sí mismo).

UML es un estándar de modelado, este lenguaje ayuda a describir los procesos, cualquiera que ellos sean y no necesariamente solo para el proceso de desarrollo de software.

4.3.1 Reglas de UML

Los bloques de construcción de UML no pueden combinarse de cualquier manera. Como cualquier lenguaje UML tiene unas reglas que especifican a qué debe parecerse un modelo bien formado. Un modelo bien formado es aquel que es semánticamente auto consistente y está en armonía con todos sus modelos relacionados.

UML tiene reglas semánticas para:

- Nombres: Cómo llamar a los elementos, relaciones y diagramas.
- Alcance: El contexto que da significado específico a un nombre.

- Visibilidad: Cómo se pueden ver y utilizar esos nombres por otros.
- Integridad: Cómo se relacionan apropiada y consistentemente unos elementos con otros.
- Ejecución: Qué significa ejecutar o simular un modelo dinámico.

Los modelos que construidos durante el proceso software de un sistema con gran cantidad de software tienden a evolucionar y pueden ser vistos por diferentes usuarios de formas diferentes y en momentos diferentes. Por esta razón, es común en el equipo de desarrollo no sólo construir modelos bien formados, sino también construir modelos que sean:

- Abreviados: Ciertos elementos se ocultan para simplificar la vista.
- Incompletos: Pueden estar ausentes ciertos elementos.
- Inconsistentes: No se garantiza la integridad del modelo.

Estos modelos que no llegan a ser bien formados son inevitables conforme los detalles de un sistema van apareciendo y mezclándose durante el proceso software. Las reglas de UML estimulan (pero no obligan) a considerar las cuestiones más importantes de análisis, diseño e implementación que llevan a tales sistemas a convertirse en bien formados con el paso del tiempo.

Capítulo 5: Diseño

En este capítulo se muestra la solución propuesta del Sistema de Seguridad que involucra todos los fundamentos que en capítulos anteriores se han mostrado, tales como detección de intrusos, cortafuegos, ataques, etc.

En la **Figura 5-1** se muestra la propuesta del algoritmo general del sistema, y posteriormente se describen los elementos más relevantes del mismo.

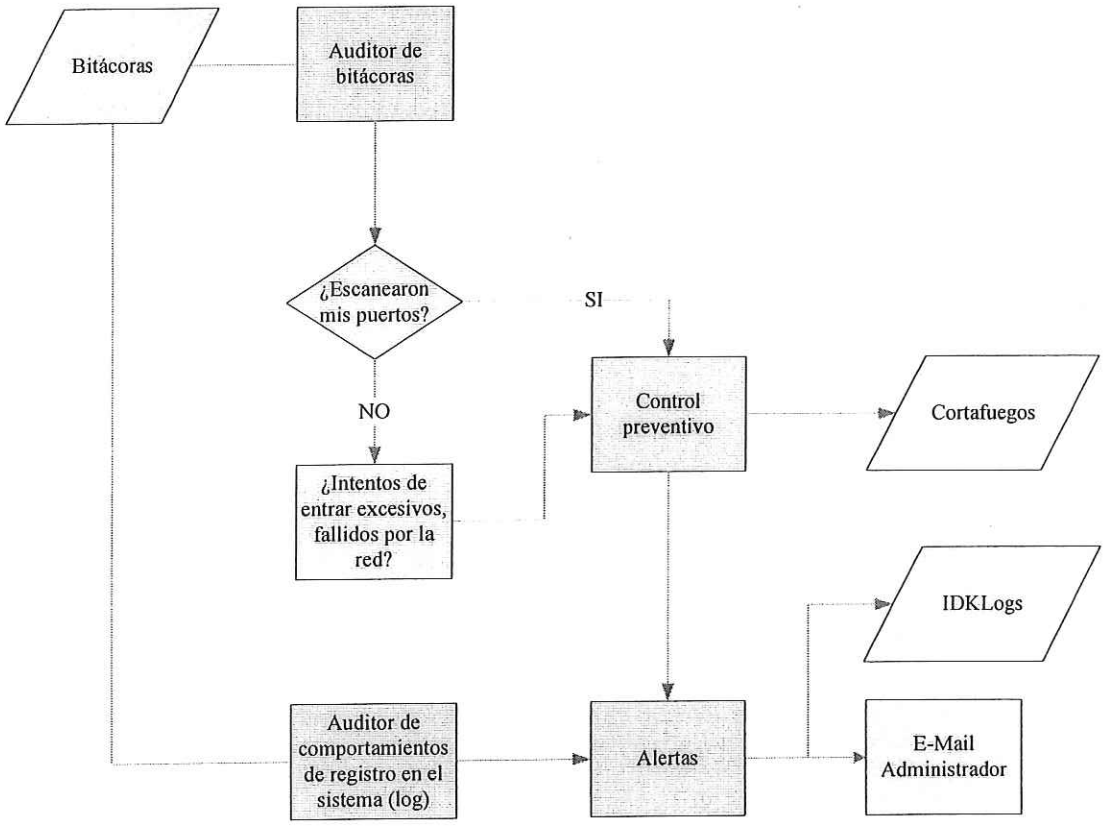

Figura 5-1. Diagrama general del sistema

5.1 Especificación del sistema

El sistema IDK deberá ser:

a) Amigable: Crear interfaces que causen este efecto de sencillez al usuario, proporcionando una rápida comprensión de lo que el sistema es capaz de hacer.

b) Portable: IDK podrá ser usado en los diferentes sistemas Linux que existen por ejemplo: SuSe, Mandrake, Fedora, Mandriva, Red Hat, etc. Para cumplir con este requisito se debe elegir el lenguaje de programación más adecuado.

El sistema IDK tiene cuatro módulos principales, los cuales llenan lo presentado en la figura 5-1. A continuación se explican brevemente cada módulo del sistema que se deberá desarrollar.

- **Control preventivo:** es el que ejecuta las reglas de cortafuegos.
- **Alertas:** Notifica lo sucedido en el sistema al administrador.
- **Auditor de comportamientos de registro en el sistema:** El que evalúa el comportamiento de los usuarios y emite una alerta si algún usuario se ha comportado fuera de lo común.
- **Auditor de bitácoras:** El que revisa las bitácoras en tiempo real para activar el control de prevención de intrusion en el sistema.

A continuación se describen los algoritmos de forma genérica y se ilustran los pasos mediante los diagramas de secuencia establecidos con la notación usada por el lenguaje unificado de modelado UML, el cual se menciona en el capítulo anterior.

5.1.1 Registro al sistema

El Administrador intenta ingresar al sistema IDK para usarlo como protección de su computadora de posibles ataques de usuarios maliciosos de la red.

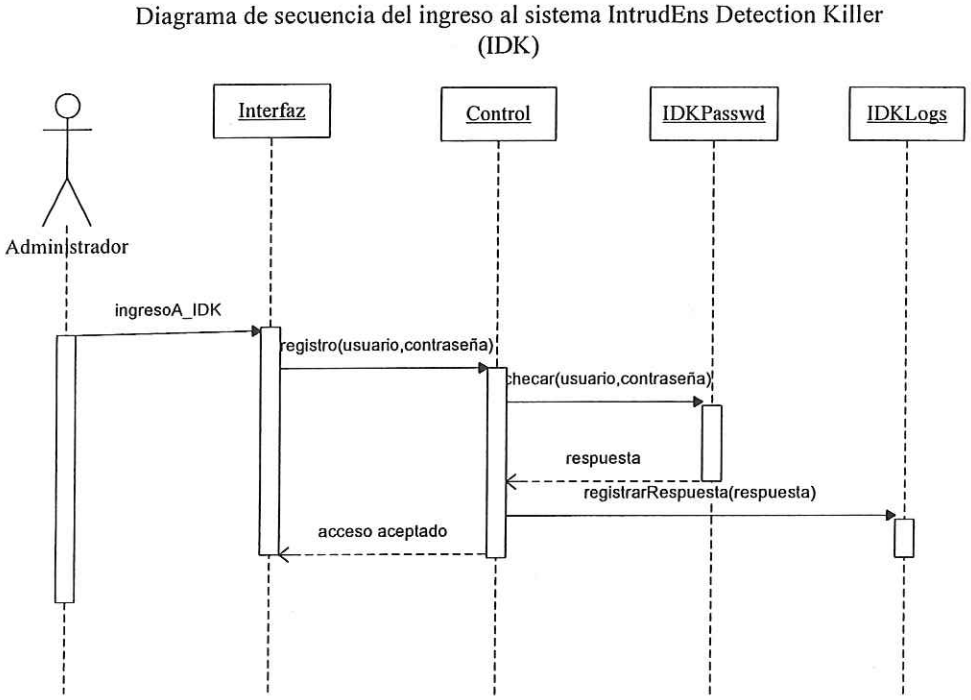

Figura 5-2. Diagrama de registro correcto al sistema

Diagrama de secuencia del ingreso al sistema IntrudEns Detection Killer (IDK)

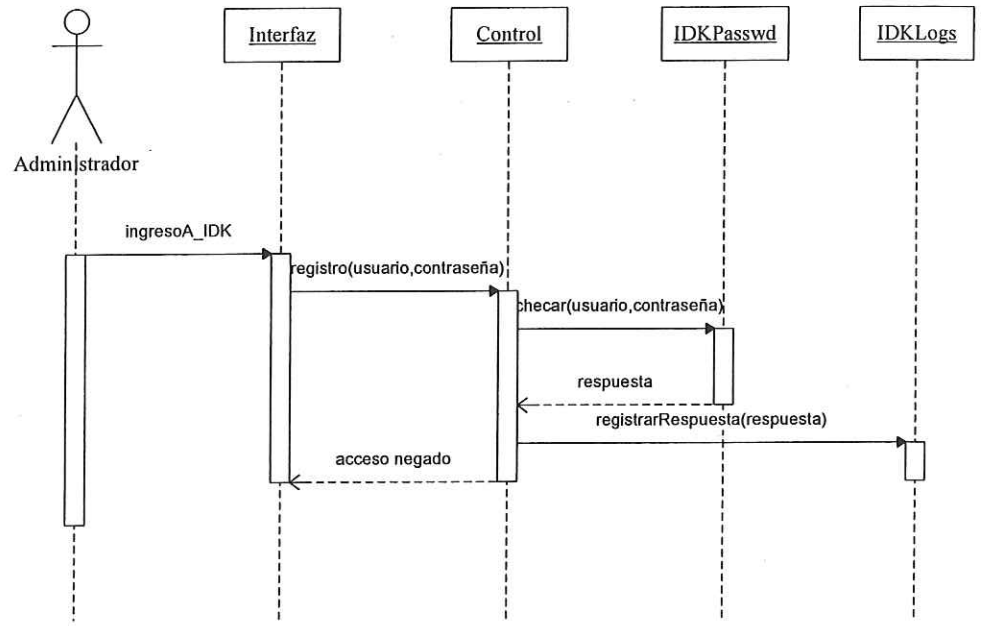

Figura 5-3. Diagrama de registro incorrecto al sistema

En este diagrama de secuencia se ven los pasos que el sistema tiene internamente y que son invisibles ante el usuario, en primera instancia el Administrador solicita el ingreso al sistema, por lo que el sistema recibe el nombre de usuario y su contraseña. El sistema posee su propia configuración de usuario y contraseña independiente al sistema operativo, entonces el propio IDK valida los datos proporcionados por el usuario. Si el usuario y contraseña son válidos se le proporciona el acceso a quien hizo la petición como se muestra en la **Figura 5-2**, por el contrario si alguno de los datos es incorrecto el acceso al sistema será negado por IDK al usuario enviándole una señal de error como se muestra en la **Figura 5-3**.

5.1.2 Agregar reglas al cortafuegos

En esta opción del sistema IDK, el Administrador puede agregar las reglas que desee en el cortafuegos para poder tener un mejor control. Esta opción provee flexibilidad al sistema pero cabe mencionar que la eficacia de las reglas en el cortafuegos dependerá de la experiencia que el administrador tenga en cortafuegos específicamente en la herramienta implementada en los sistemas Linux para filtrado de paquetes **iptables** que ya se mencionó en el Capítulo 2 sección 2.4.2.

Diagrama de secuencia de administración de reglas del cortafuegos con el sistema IntrudEns Detection Killer (IDK)

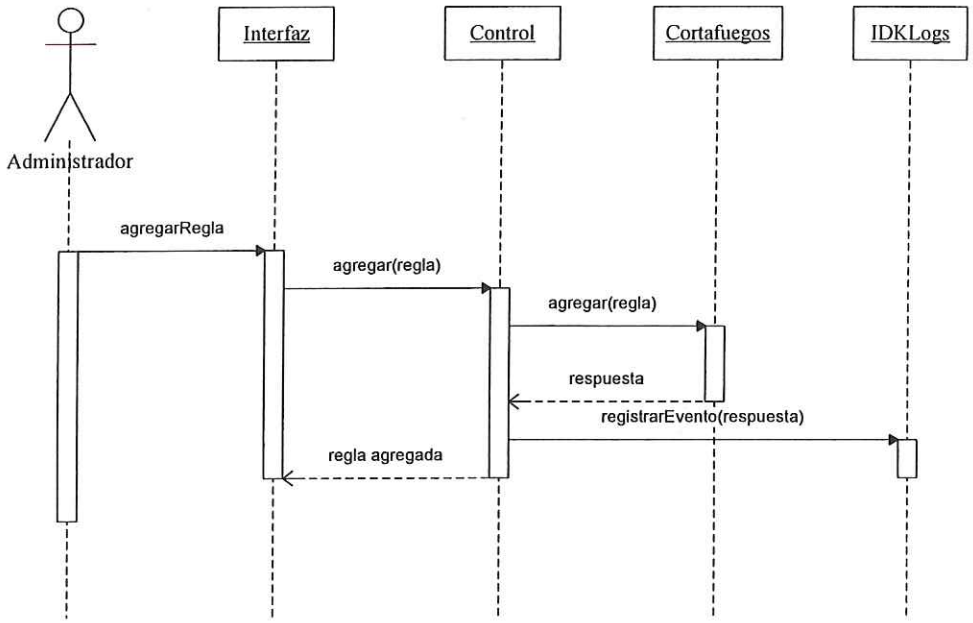

Figura 5-4. Diagrama de regla agregada en el cortafuegos

Diagrama de secuencia de administración de reglas del cortafuegos con el sistema IntrudEns Detection Killer (IDK)

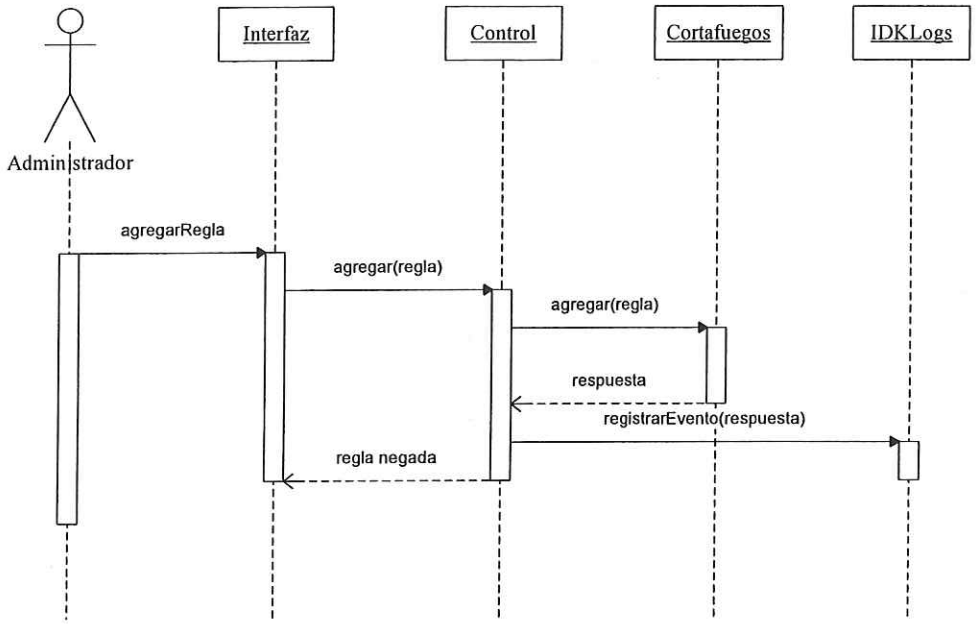

Figura 5-5. Diagrama de regla agregada negada en el cortafuegos

En la **Figura 5-4** se describen los pasos que se realizan al interior del sistema. En principio el administrador elegirá la opción de agregar regla en el cortafuegos que el sistema muestra gráficamente, una vez ingresada la nueva regla el sistema intentará que el cambio en el cortafuegos tome efecto en el sistema operativo, lo cual arrogará una respuesta positiva al cambio como se muestra en la **Figura 5-4** o en su defecto negativa al cambio en el sistema operativo como se muestra en la **Figura 5-5**. La respuesta arrojada por el sistema operativo se guardará en los registros de IDK para que esos mensajes puedan ser vistos por el mismo administrador por medio del sistema IDK.

5.1.3 Eliminar reglas al cortafuegos

Tratando de hacer que el sistema IDK sea lo más amigable y flexible para el usuario, también será posible eliminar las reglas del cortafuegos que el administrador determine como innecesarias las secuencias de las posibilidades de esta opción se ven a continuación.

Diagrama de secuencia de administración de reglas del cortafuegos con el sistema IntrudEns Detection Killer (IDK)

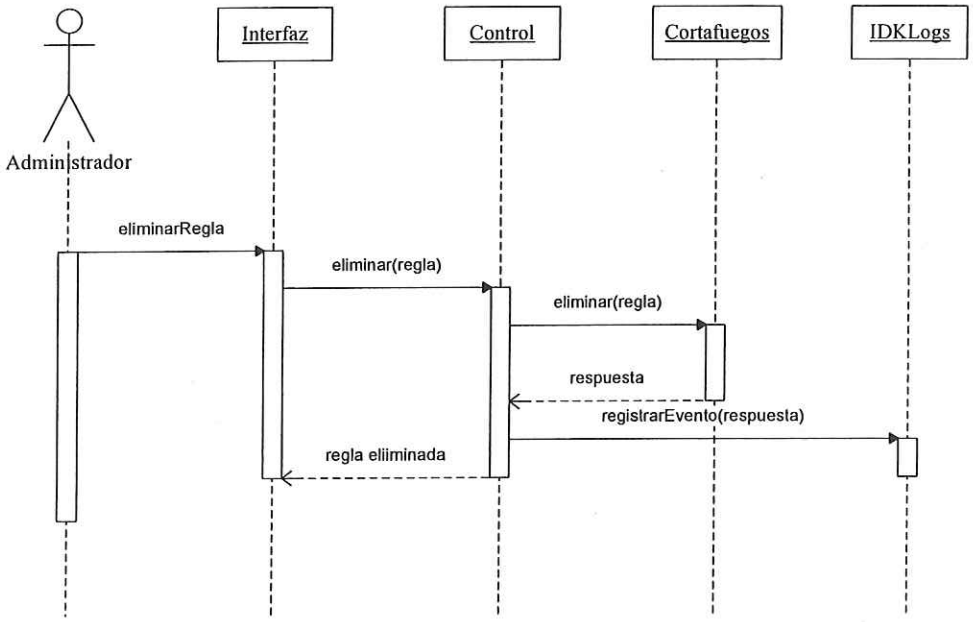

Figura 5-6 Diagrama de regla eliminada en el cortafuegos

Diagrama de secuencia de administración de reglas del cortafuegos con el sistema IntrudEns Detection Killer (IDK)

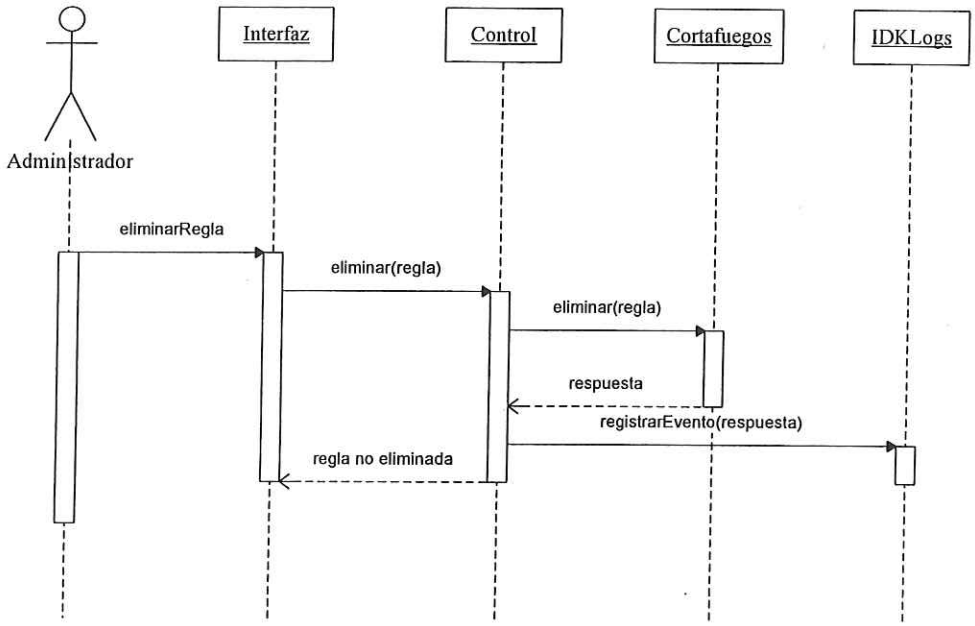

Figura 5-7. Diagrama de regla no eliminada en el cortafuegos

Anteriormente se describen los pasos que se llevan a cabo para agregar reglas al cortafuegos implementado por el sistema IDK y el hecho de eliminar las reglas de dicho cortafuegos no es muy diferente en su secuencia básica al de agregarlas. En principio el usuario deberá seleccionar la regla que desee eliminar del cortafuegos para así elegir la opción de eliminar regla, lo cual el sistema IDK intentará aplicar en el cortafuegos del sistema operativo por lo que se obtendrá una respuesta satisfactoria que es la que se ve en la **Figura 5-6** o por el contrario una respuesta negativa a dicha petición como se muestra en la **Figura 5-7**. Cualquiera que sea la respuesta del sistema operativo se almacenará en los registros de IDK.

5.1.4 Detección del escaneo de puertos

Esta no es una opción que el sistema dará al usuario, por el contrario es simplemente una alarma al administrador y que el mismo sistema IDK pondrá una solución al problema detectado en tiempo real.

Diagrama de secuencia de detección de escaneo de puertos con el sistema IntrudEns Detection Killer (IDK)

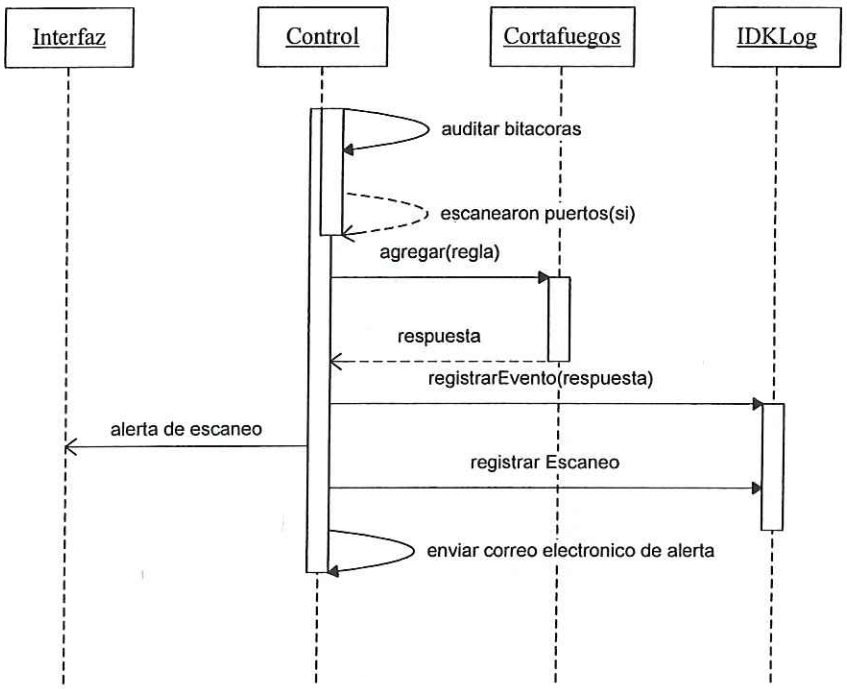

Figura 5-8. Diagrama de detección de escaneo de puertos

El nombre de este módulo del sistema IDK es muy claro, detecta los escaneos en los puertos que puedan existir en el sistema operativo y ante esta situación tomará las

medidas pertinentes para tratar de evitar una potencial intrusión en nuestro sistema por algún usuario que al parecer no tiene buenas intenciones y que es una amenaza a la seguridad.

Este módulo estará monitoreando constantemente las bitácoras del sistema, si se cumplen las condiciones y se detecta un escaneo en los puertos el sistema IDK crea una regla y la agrega al cortafuegos para evitar la intrusión, también se registrará la respuesta emitida por el cortafuegos en los registros de IDK, se le mostrará una alerta gráfica al administrador, se registrará el escaneo en la bitácora de IDK y por último se le enviará un correo electrónico al administrador para que por cualquiera de las alertas anteriores se percate de lo sucedido. **Figura 5-8**.

5.1.5 Detección de registros fallidos

El propósito de este módulo es bloquear la entrada a una dirección IP que ha intentado vulnerar sin éxito, Este comportamiento parecería el de una persona que trata de ingresar a *fuerza bruta* al sistema.

Diagrama de secuencia de detección de ingreso por algún puerto con el sistema IntrudEns Detection Killer (IDK)

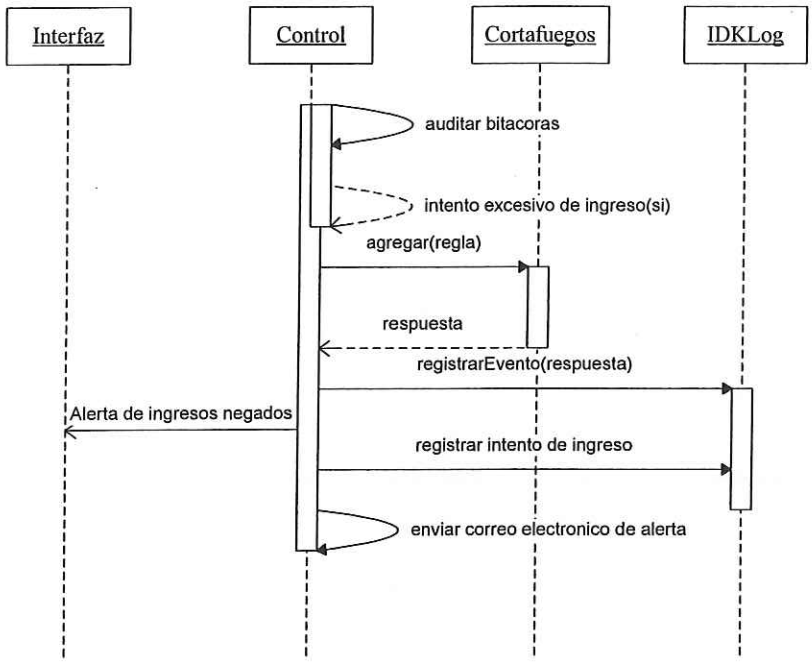

Figura 5-9. Diagrama de detección de ingresos fallidos

Este módulo revisará las bitácoras con el objetivo de ver si alguien esta tratando de ingresar por alguno de los puertos al sistema y no puede conseguirlo, una vez detectado ese evento el sistema IDK creará una regla que se colocará en el cortafuegos para negarle la entrada al posible atacante. El sistema registrará la respuesta emitida por el cortafuegos, muestra una alerta al administrador en forma gráfica, también registra el mensaje en los propios registros de IDK y por último enviará un correo electrónico al administrador. **Figura 5-9**.

5.1.6 Comportamientos extraños

Esta opción se ejecuta de modo invisible o transparente al administrador. Se encarga de evaluar las bitácoras buscando un comportamiento extraño en los usuarios de acuerdo a patrones que cada usuario vaya generando con el tiempo que usa el sistema.

Diagrama de secuencia de detección de comportamientos extraños con el sistema IntrudEns Detection Killer (IDK)

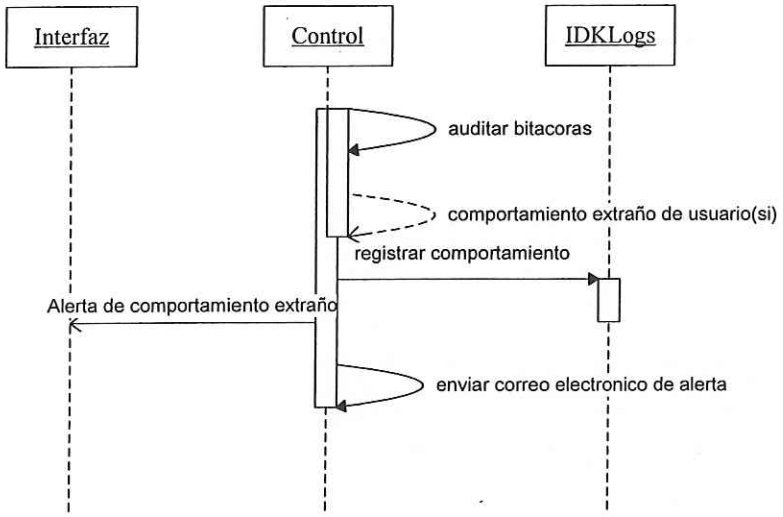

Figura 5-10. Diagrama de detección de comportamientos extraños

El la **Figura 5-10** se describen los pasos del módulo que se encarga de la detección de anomalías de los usuarios, se auditarán las bitácoras y si se encuentra una anomalía se registrará en la bitácora de IDK, se le enviará una alerta gráfica al administrador y también un correo electrónico.

5.2 Lenguaje de programación

Dada la necesidad del sistema IDK de ser portable y amigable al usuario, se opto por el lenguaje de programación Java (creado por Sun en 1995). Este lenguaje proporciona portabilidad entre diferentes plataformas de sistemas operativos así como elementos gráficos que producen un efecto amigable al usuario.

Java es el primer lenguaje que es compilado e interpretado en forma simultánea. Cuando un programador realiza una aplicación o un applet en Java y lo compila, el compilador de Java crea los archivos conocidos como ByteCode, estos archivos son a penas la mitad del proceso que lleva la ejecución de lo que se pretende y solo pueden ser interpretados por lo que se conoce como Máquina Virtual de Java, la cuál es la que permite la portabilidad sobre el hardware y el sistema operativo, existen diversas máquinas virtuales para las diferentes plataformas (Unix, Linux, Windows 95/98/NT/XP, Macintosh, etc.) y cada una de ellas la máquina virtual interpreta el ByteCode para después enviarlo al procesador donde se esté trabajando, de modo que sin importar el origen de creación del ByteCode este podrá ser ejecutado en forma correcta por la máquina sobre la que se esté trabajando. Dado este mecanismo podemos hacer que la aplicación sea multiplataformas, es decir, portable.

Java es un lenguaje gratuito y además está en constante evolución, de esta manera el lenguaje crece cada vez más y toca puntos muy importantes como la

programación hacia bases de datos, multi-hilos, programación en redes y programación distribuida. Las características de Java son: simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portable, de alto desempeño, multi-hilos (multithreaded) y dinámico.

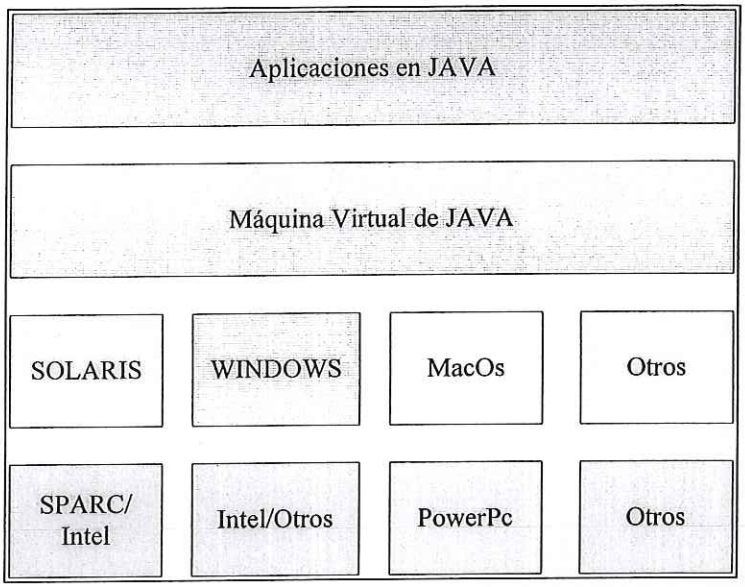

Figura 5-11 Máquina Virtual de Java (JVM) sobre las plataformas y sus procesadores, permitiendo que una aplicación se ejecute sobre varias plataformas.

Por desgracia, el compilador de Java es bastante lento si lo comparamos con otros lenguajes como C++ y eso se debe a que es interpretado y no ejecutado.

5.3 Interfaz de usuario

La interfaz de usuario es algo muy delicado que el programador trata de adecuar a lo que el sistema puede hacer, de tal manera que sea sencillo aun que a veces eso no es

una tarea fácil de lograr, se deben usar imágenes que ayuden a la comprensión de las opciones del sistema así como los colores adecuados para la mejor comprensión para el usuario.

Las características de la interfaz son:

- **Consistente**, Se debe utilizar un formato consistente para la elección de menús, la entrada de ordenes, visualización de datos, respaldo y recuperación de datos.
- **Ofrecer una retroalimentación significativa**, Se debe proporcionar al usuario una realimentación visual y auditiva para asegurar que se establece una comunicación bidireccional (entre el usuario y la interfaz).
- **Buscar la eficiencia en el diálogo, el movimiento y el pensamiento**, El número de pulsaciones debe ser minimizado, la distancia que un ratón debe recorrer entre dos pulsaciones debe ser tenida en cuenta al diseñar el formato de presentación, y el usuario debe encontrar pocas veces una situación en la que tenga que preguntar, "¿Qué significa esto?".
- **Perdonar los errores**, El sistema debe protegerse de los errores del usuario que pudiesen afectarlo causándole un fallo.
- **Mostrar sólo aquella información que sea relevante en el contexto actual**, El usuario no debe tener que buscar a través de datos extraños, menús y gráficos para obtener información relevante a una función concreta del sistema.

- **Utilizar ventanas (si están disponibles) para modularizar los diferentes tipos de información,** Las ventanas mantienen accesibles al usuario muchos tipos de información.

Dada la información presentada anteriormente, esta se consideró para la creación de las interfaces, las cuales se explican con más detalle en el siguiente capítulo.

Capítulo 6: Desarrollo

En base a los diagramas descritos en el capítulo anterior, donde se especifican los módulos del sistema, se desarrollaron con la versión del lenguaje de programación java del paquete jdk-1_5_0_05-linux-i586 y sobre el sistema operativo SuSe 9.1, la herramienta que cumple con lo especificado en dicho diseño.

La utilización de los diagramas de secuencia que se realizaron en el diseño, provee una gran ventaja al momento del desarrollo de la herramienta, por que prácticamente tenemos el problema resuelto. De tal forma podemos concluir que un buen diseño es la base del éxito en el desarrollo de software y los criterios de éxito varían según lo que se pretenda. En nuestro caso la base del éxito depende cae en el hecho de que la herramienta haga lo que se planteo desde un inicio.

En las siguientes líneas se describirá cada módulo en forma algorítmica y se mostrarán los resultados de las pruebas realizadas al sistema.

Recapitulando, las partes importantes del sistema son:

- **Control preventivo:** es el que ejecuta las reglas de cortafuegos.
- **Alertas:** Notifica lo sucedido en el sistema al administrador.

- **Auditor de comportamientos de registro en el sistema:** El que evalúa el comportamiento de los usuarios y emite una alerta si algún usuario se ha comportado fuera de lo común.
- **Auditor de bitácoras:** El que revisa las bitácoras en tiempo real para activar el control de prevención de intrusion en el sistema.

Estos son los módulos que conforman el sistema. Todos son igual de importantes y necesarios.

6.1 Soluciones

Aquí se muestran los algoritmos de cada módulo. Describen paso a paso la tarea de cada elemento del sistema.

6.1.1 Control preventivo

Este módulo ejecuta las reglas del cortafuegos, dicha configuración se guarda en un *script*, el administrador del sistema IntrudEns Detection Killer puede modificar dicho script de acuerdo a su gusto y conocimiento. Este módulo puede ejecutarse dadas dos condiciones, Primero si el administrador ejecuta la opción en la interfaz gráfica del sistema IDK, Segundo si las bitácoras de escaneo cambian y el IP que hizo el escaneo no se encuentra ya en el script del cortafuegos.

Algoritmo:

Condiciones:

Estado, esperando.

Pasos:

1. Esperando por un evento.
2. Si el evento es la opción de aplicar reglas de cortafuegos.
    a. Obtener ubicación del archivo a ejecutar, script.
    b. Ejecutar script.
3. Si no, si el evento fue que cambiaron las bitácoras de los escaneos.
    a. Obtener las direcciones IP de la bitácora de escaneos.
    b. Eliminar las direcciones IP repetidas.
    c. Para cada dirección IP.
        i. Si dirección no existe en el cortafuegos.
            1. agregar regla para bloquear el acceso al IP.
            2. Generar alerta usando el módulo de Alertas.
4. Fin algoritmo.

6.1.2 Alertas

Este módulo inicia cuando se solicita, puede ser activado por el Control preventivo o el Auditor de comportamientos de registros en el sistema.

Algoritmo:

Condiciones:

Estado, inactivo.

Entrada, cadena de alerta.

Pasos:

1. Obtener Tiempo, Año, Mes, Día, Hora, Minutos, Segundos.
2. Obtener archivo de historial.
3. Concatenar nueva alerta al historial.
4. Guardar historial.
5. Enviar Correo electrónico al administrador.

6.1.3 Auditor de comportamientos de registros en el sistema

La finalidad de este módulo es obtener las irregularidades con respecto a los registros de los usuarios al sistema operativo así como las fallas con los servicios controlados por el demonio *xinetd*, el administrador del sistema IDK podrá ver cuando hay intentos fallidos de ingreso al sistema, tanto locales como externos. Cuando un servicio fallé, también podrá ver que proceso fallo y donde se origino la falla, ya sea local o remotamente. Para la mejor comprensión y estética en la estructura del módulo se subdividió en dos, Registros y Anomalías.

Algoritmo de Registros:

Condiciones:

Ninguna.

Pasos:

1. Obtener archivo que contiene los mensajes del sistema operativo.
2. Leer el archivo con los mensajes.
3. Obtener mensajes que indican error al registrarse al sistema.
4. Mostrar mensajes en la interfaz gráfica.

Algoritmo de Anomalías:

Condiciones:

Ninguna.

Pasos:

1. Obtener archivo que contiene los mensajes del demonio xinetd.
2. Leer archivo con los mensajes.
3. Depurar archivo y obtener los mensajes que indican cuando un servicio falló.
4. Mostrar anomalías en la interfaz gráfica.

6.1.4 Auditor de bitácoras

Esta sección audita las bitácoras que contienen los escaneos que registra el demonio *scanlogd*, se usó un hilo de ejecución que actúa por su cuenta y toma las decisiones de bloquear en tiempo real las direcciones IP que hicieron escaneos a la máquina que ejecuta el sistema IDK.

Algoritmo:

Condiciones:

Ninguna.

Pasos:

1. Mientras el sistema este en ejecución.
    a. Auditar bitácoras.
        i. Contar número de escaneos actuales.
        ii. Comparar número de escaneos actuales con antiguos.
        iii. Si número de escaneos actuales es mayor que los antiguos.
            1. Depurar mensajes de escaneos.
                a. Leer archivo de escaneos.
                b. Obtener lista de direcciones IP.
                c. Eliminar direcciones repetidas.
            2. Para cada dirección IP.
                a. Si dirección IP no existe en cortafuegos.

i. Agregar regla para negar el acceso a la dirección IP.

ii. Generar alerta.

3. Numero de escaneos antiguo será igual al numero de escaneos actuales.

b. Esperar 10 segundos.

6.2 Interfaces del sistema IntrudEns Detection Killer version beta 1.0

6.2.1 Interfaz principal

A continuación se muestra la imagen de la ventana principal del sistema IDK y se describe lo que se hace en cada opción del sistema.

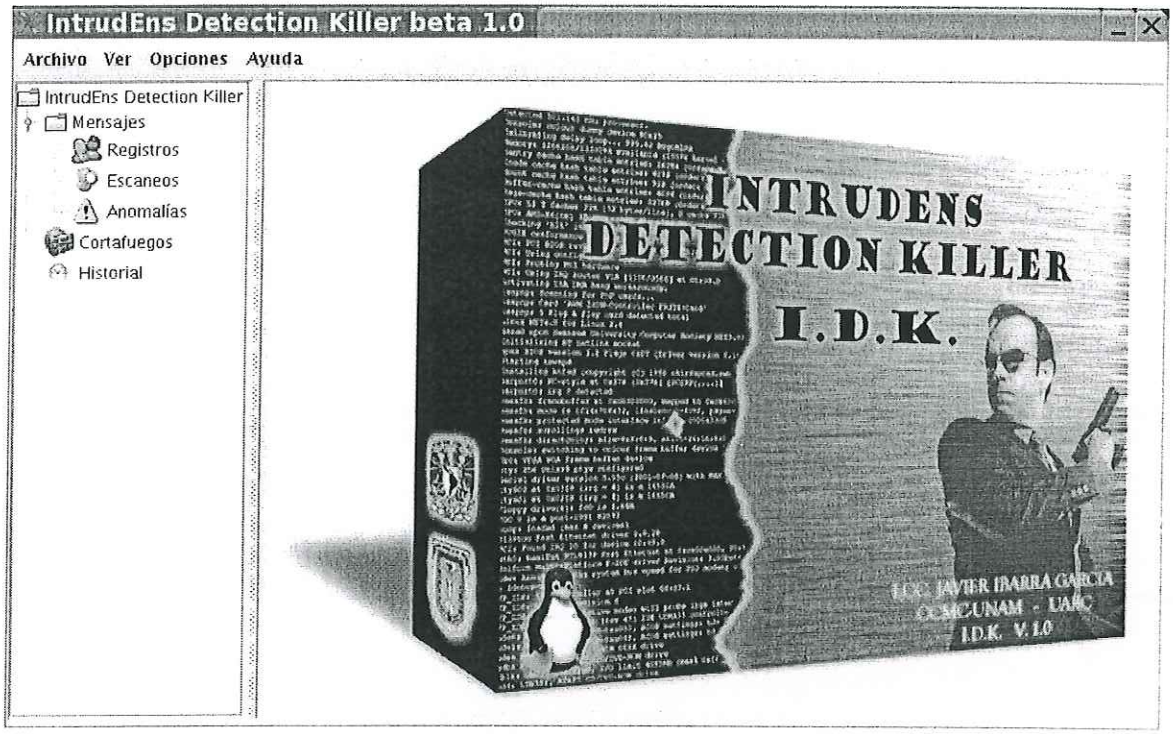

Imagen 6-1 Interfaz principal del sistema IDK.

Para el fácil uso e interacción con el usuario del sistema se colocó un menú de árbol a la izquierda, donde cada vez que el usuario seleccione una de las posibles opciones del menú de árbol, en la parte derecha aparecerá lo deseado. Posteriormente se muestra cada opción y se explica con detalle.

6.2.1.1 Registros

Esta opción permite al usuario ver la lista de registros que han sido rechazados por el sistema operativo, ya sea por error de contraseña o un usuario desconocido, también permite ver el origen del intento de acceso, pudiendo ser local o remoto. Dados

los datos, el administrador podrá analizarlos y buscar si estos intentos fueron de un posible hacker y así tomar las medidas que considere necesarias, la siguiente imagen ejemplifica de una manera mas clara lo que la aplicación muestra gráficamente.

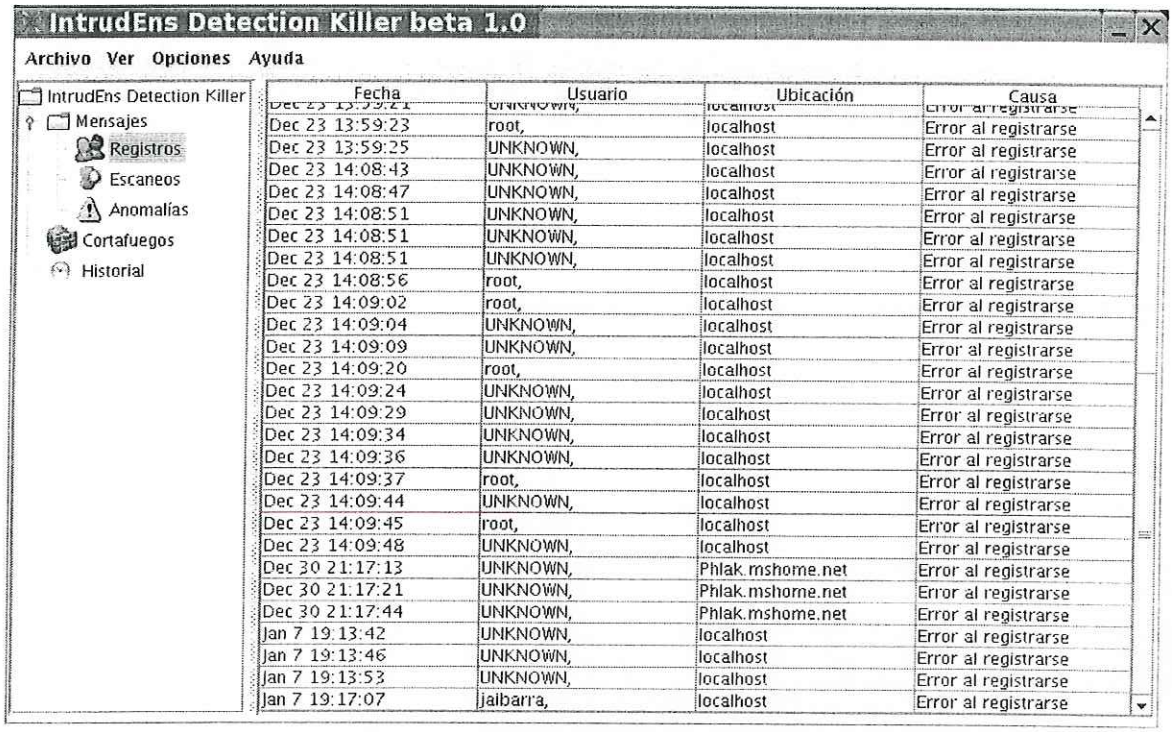

Imagen 6-2 Registro fallido de usuarios al sistema operativo.

6.2.1.2 Escaneos

Aquí se muestran los escaneos que se han realizado en la computadora local, y como se menciono anteriormente, se bloquea automáticamente a las direcciones IP que son desconocidas para la computadora, ello con el objetivo de proteger el sistema de un posible ataque.

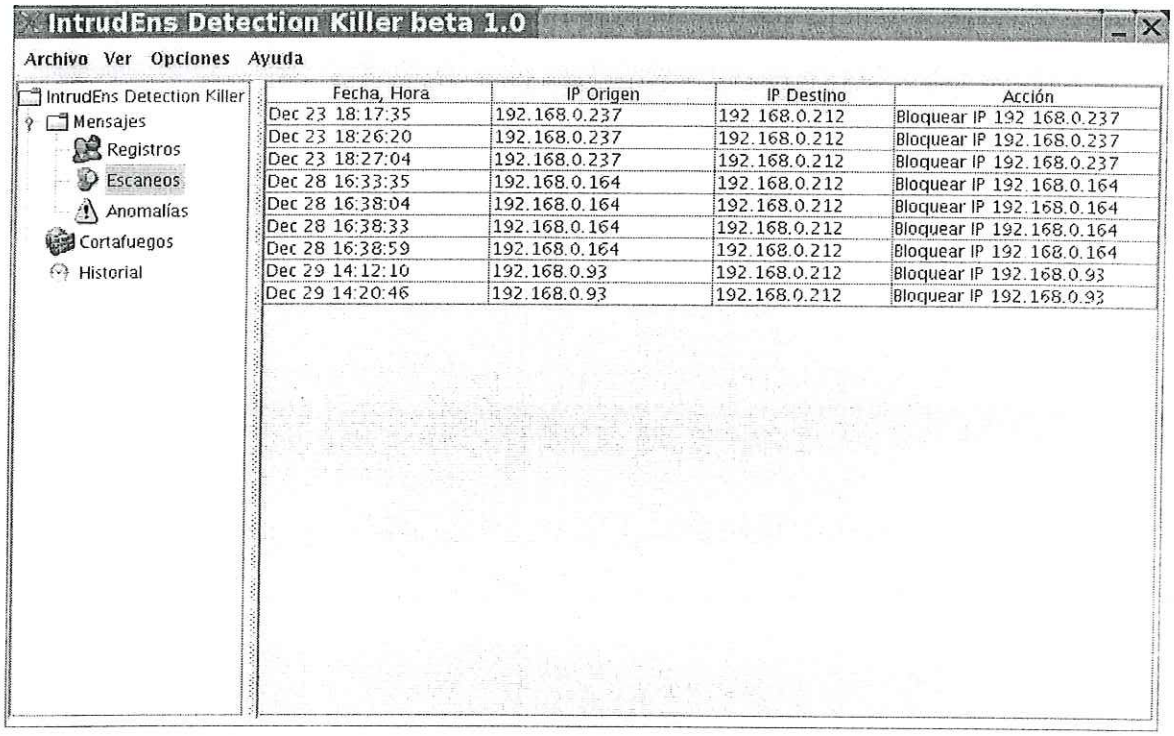

Imagen 6-3 Escaneos detectados por IDK.

6.2.1.3 Anomalías

Las anomalías en el sistema, significan que algún servicio controlado por el demonio *xinetd* falló de alguna manera, podría ser por una mala terminación del servicio de una manera remota, estas alertas avisan al administrador de esos problemas para que los evalúe y tome una decisión al respecto.

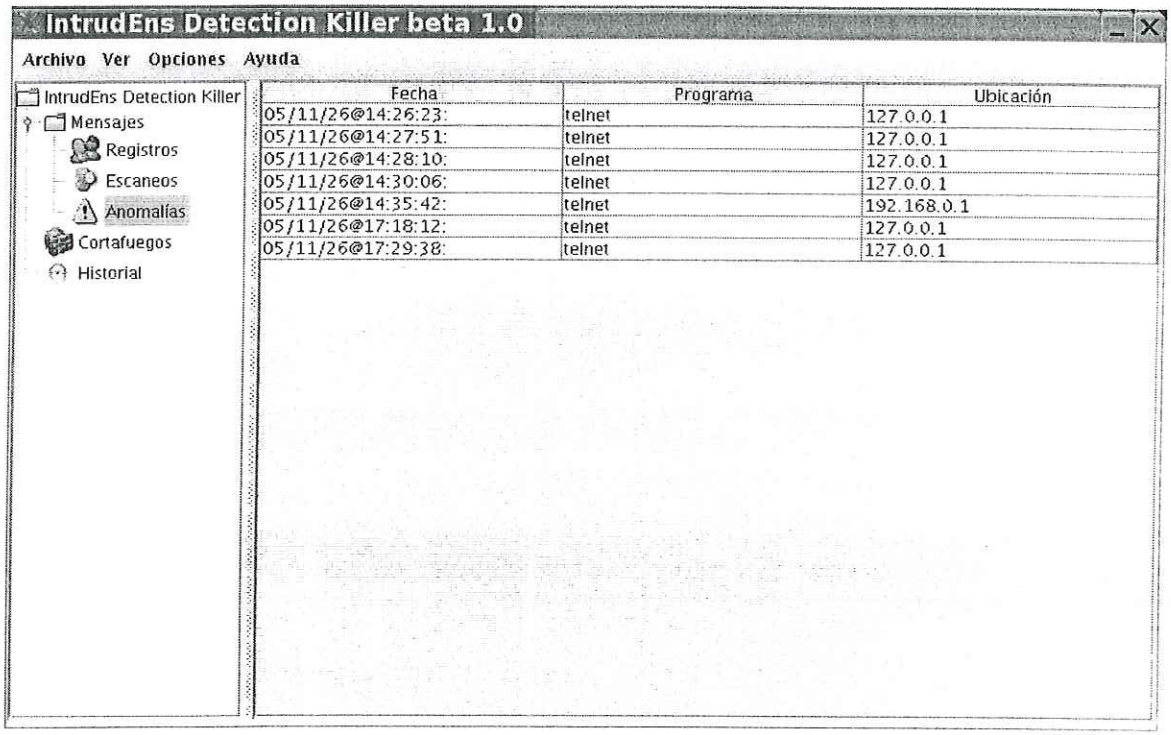

Imagen 6-4 Anomalías detectadas en IDK de los servicios.

6.2.1.4 Cortafuegos

El cortafuegos, es la herramienta que usa IDK para la protección contra los ataques que pudieran presentarse. La administración del cortafuegos queda bajo responsabilidad del encargado de manejar el sistema IDK. El usuario deberá conocer el sistema de seguridad implementado para Linux conocido como **iptables**, dicha herramienta es la que se implementa como cortafuegos en dicho sistema operativo en la actualidad. La **Imagen 6-5** muestra la interfaz del cortafuegos de IDK, donde se puede ver que el administrador podrá realizar una serie de acciones en el mismo, tales como

Agregar, Modificar y Eliminar reglas, Aplicar las reglas en el cortafuegos y ver el Estatus del mismo para ver la actual configuración.

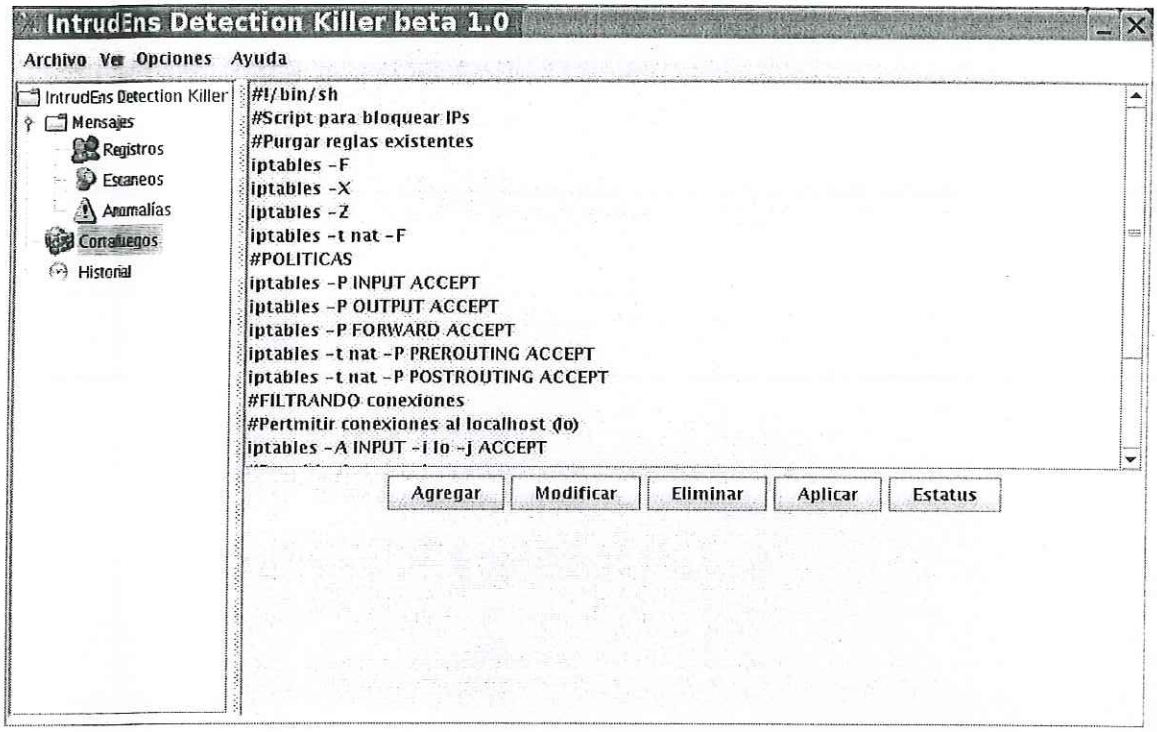

Imagen 6-5 Administración del cortafuegos IDK.

6.2.1.4.1 Agregar regla al cortafuegos

Esta ventana se abre al momento de seleccionar la opción de AGREGAR, en el espacio en blanco se deberá escribir la nueva regla que se desea. El botón de Aceptar agrega la regla y el de Cancelar solo cierra la ventana y no hace nada en el cortafuegos.

Imagen 6-6 Agregar regla en el cortafuegos.

6.2.1.4.2 Modificar regla en el cortafuegos

Cuando se selecciona una regla en el cortafuegos y se elige la opción de modificar, aparece una ventana como en la **Imagen 6-7**, donde se carga la regla seleccionada para que se modifique a criterio del administrador.

Imagen 6-7 Modificar regla en el cortafuegos.

6.2.1.4.3 Estatus del cortafuegos

El estatus es importante, por que así el administrador podrá ver si las reglas en el cortafuegos tomaron efecto y a su vez, ver con un poco de claridad como se comporta el cortafuegos.

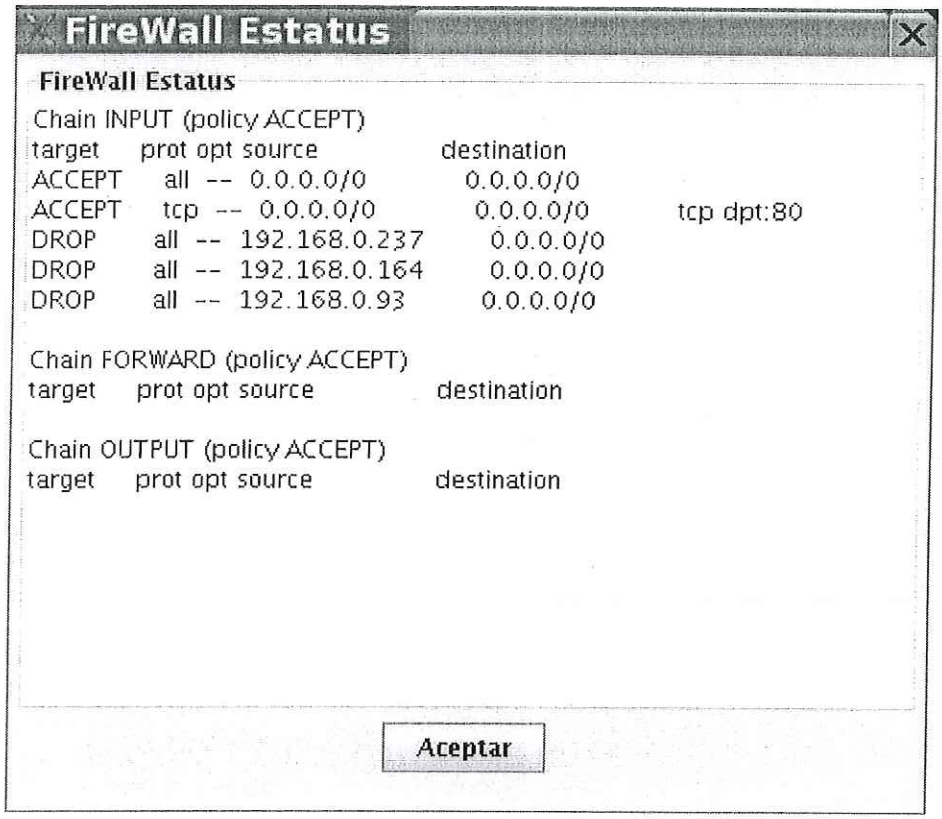

Imagen 6-8 Estatus del cortafuegos.

6.2.1.5 Historial

El historial muestra lo sucedido en el sistema IDK, tal como cambios en el cortafuegos y escaneos, este historial le permitirá al administrador ver lo sucedido en el tiempo que no estuvo presente físicamente en la computadora, lo cual es muy útil por que permite ver lo sucedido. También se verá cuando se toma una acción en contra de un escaneo para proteger el sistema.

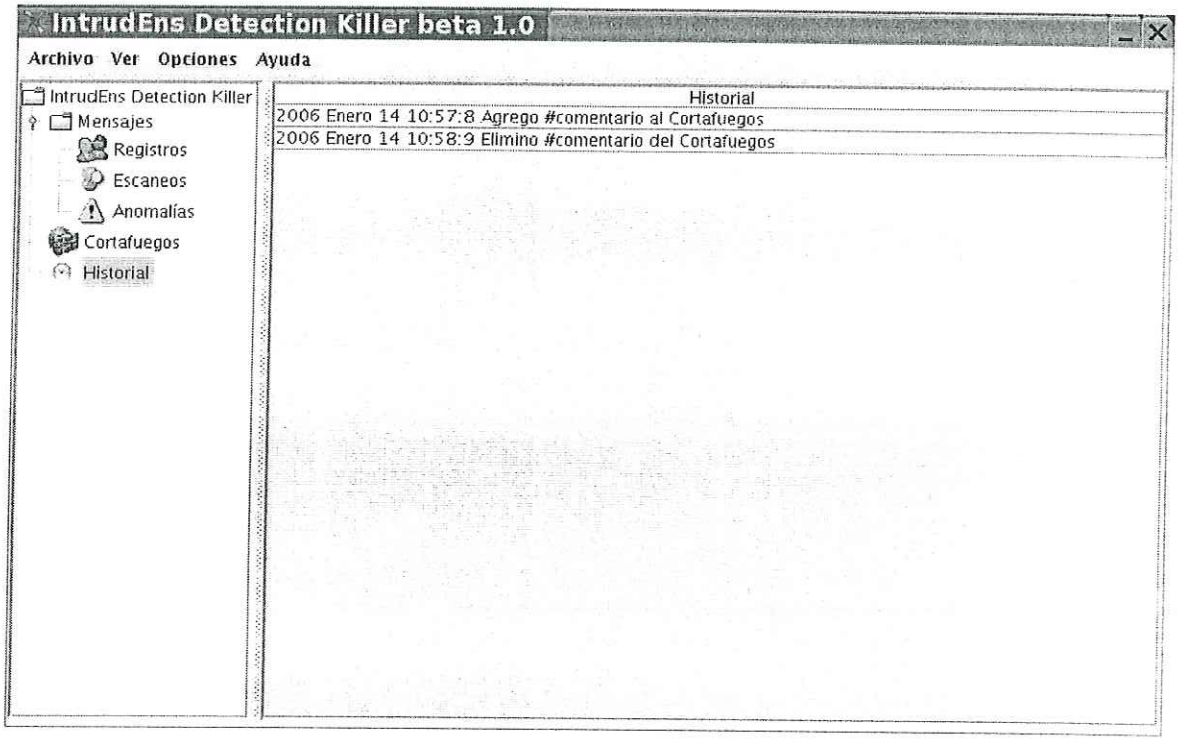

Imagen 6-9 Historial del sistema IDK.

6.2.1.6 Configuración del sistema

La configuración que el sistema IDK posee, le da gran flexibilidad dado que algunos administradores de sistemas tienden a cambiar las bitácoras del sistema operativo que controlan, para engañar al enemigo en caso de un ataque por un hacker. Por tal motivo se le permite que al administrador establecer las rutas de los archivos que son necesarios para el buen funcionamiento del IDK.

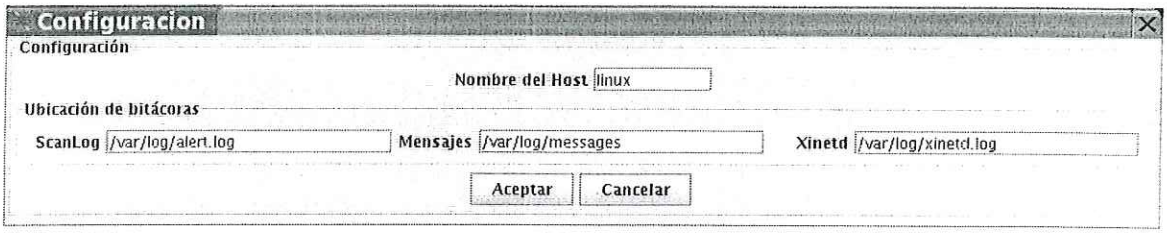

Imagen 6-10 Configuración del sistema IDK.

Todas la imágenes mostradas en este capítulo así como los algoritmos son el **IntrudEns Detection Killer beta 1.0**, esta herramienta provee al usuario un control total sobre el cortafuegos y también lo protege de usuarios maliciosos de la red. El IDK beta 1.0 es una herramienta poderosa que puede ser de gran utilidad para los administradores de sistemas. La herramienta se basó en los aspectos de portabilidad que provee el lenguaje de programación Java y la seguridad inerte que tienen los sistemas UNIX/LINUX para crear un sistema capaz de proteger a los usuarios de los peligros de internet.

En el siguiente capítulo se habla sobre las pruebas realizadas al sistema IDK beta 1.0.

Capítulo 7: Pruebas del IDK

En este capítulo se presentan las pruebas realizadas al sistema IDK, dichas pruebas demuestran la funcionalidad del sistema.

7.1 Infraestructura

El equipo y el contexto bajo el cuál se realizaron las pruebas al sistema IDK se describen a continuación.

7.1.1 Hardware utilizado en las pruebas del IDK

a) PC IBM NetVista, Procesador Celeron 1.7Ghz.

b) Compaq presario, Procesador Pentium III.

7.1.2 Software utilizado en las pruebas del IDK

Sistemas operativos:

a) SuSe 9.1, Linux.

b) PHLAK (Professional Hackers Linux Assault Kit) 0.3, Linux.

c) Microsoft Windows XP, Professional Version 2002, SP2.

Software:

a) NMap (Network Mapper). Herramienta para el escaneo de puertos.

b) NMapFe. Herramienta gráfica para escanear con diferentes programas de escaneos de puertos.

c) scanlogd. Herramienta para la detección de escaneos.

Las herramientas de software utilizadas son las más famosas en su ámbito así como efectivos, por ello se decidió utilizarlas para la evaluación del sistema.

7.2 Evaluación del IDK

Para verificar que el sistema funcionara como se planteo en el diseño, se implemento una pequeña red de área local interconectada por un *switch* como interfaz de comunicación entre las computadoras de la red.

El objetivo general de IDK es proteger mediante un cortafuegos a los posibles intrusos que se encuentran en la red, por lo que se optó por crear el ambiente para llevar a cabo simulaciones y así probar el sistema.

7.3 Diseño de pruebas

Las pruebas que se llevaron a cabo al sistema fueron pruebas de funcionalidad.

Las pruebas de funcionalidad se realizaron con la finalidad de evaluar que el sistema reaccione como se espera, los resultados de las pruebas sugieren cambios en la funcionalidad del sistema.

7.3.1 Pruebas de funcionalidad

Para llevar a cabo estas pruebas se usaron las herramientas de escaneo, tales como Nmap y NMapFe a través de la red de área local implementada.

Se usó una computadora encargada de asignar las direcciones IP a los sistemas de cómputo vía DHCP.

La computadora encargada de realizar los escaneos encendía y cargaba el sistema operativo PHLAK desde un CD, una vez cargado el sistema se procedió a realizar un escaneo de puertos a la computadora con el sistema IDK, con un tiempo de espera de a lo más diez segundos, se debe ver que el sistema fue escaneado e inmediatamente se tomar una acción de protección.

7.3.2 Resultado de las pruebas

Los resultados de las pruebas fueron satisfactorios, el sistema bloqueó las direcciones IP de los sistemas que realizaron los escaneos, el usuario del sistema es capaz de modificar a su elección la configuración del cortafuegos, así como la del sistema IDK para la reubicación de los archivos necesarios para el funcionamiento del mismo.

El siguiente capítulo presenta las conclusiones y recomendaciones para trabajos futuros.

Capítulo 8: Conclusiones y recomendaciones para trabajos futuros

Este capítulo presenta las conclusiones finales de este trabajo, la aportación del mismo y las recomendaciones de importancia que deben considerarse en trabajos futuros relacionados con la seguridad en sistemas de cómputo.

8.1 Conclusiones

El sistema IDK tiene como bases los resultados de algunas herramientas dedicadas a realizar tareas muy especificas, por lo que su buen funcionamiento depende de programas ajenos a el. El sistema cubre una pequeña parte de la seguridad en sistemas de cómputo, provee protección y alertas que permiten al administrador de los sistemas tomar las decisiones importantes pero el IDK también toma decisiones de acuerdo a lo sucedido en su alrededor.

Se puede concluir que el sistema cubre los siguientes puntos:

- Fácil de usar.
- Fácil de entender.
- Fácil de configurar.
- Responde en tiempo real ante las amenazas.
- Alerta al administrador de lo sucedido.

- Registra los eventos importantes del sistema.
- Permite el manejo del cortafuegos.
- Muestra el estado de la configuración del cortafuegos.
- Portable.

8.2 Aportaciones

Se desarrollo una aplicación que funciona como cortafuegos y que reacciona ante estímulos predeterminados de la red. Es una aplicación ligera que usa poderosas herramientas predeterminadas del sistema sobre el que se desarrollo, tales como **iptables**. A su vez es multiplataformas y tiene una gran posibilidad de crecimiento en el área de seguridad en sistemas de cómputo.

Se creo con un modelo orientado a objetos, el cual proporciona grandes posibilidades de implementar otras funcionalidades al sistema.

8.3 Recomendaciones

En este mundo donde la tecnología avanza con una gran rapidez es necesario mantener los sistemas actualizados para poder estar a la vanguardia tecnológica. Es necesario actualizar los sistemas que nos protegen de los programas y usuarios maliciosos que existen en internet, el sistema IDK usó las herramientas que en su tiempo

fueron parte la vanguardia tecnológica, pero es importante que los sistemas evolucionen y crezcan para poder cumplir el objetivo por el cuál fueron creados.

8.4 Trabajo futuro

Las herramientas que el sistema IDK utiliza, tales como iptables, scanlogd, xinetd, pueden ser desarrolladas como parte del mismo sistema, con la finalidad de no depender de programas externos. La interfaz gráfica puede ser sometida a cambios con la finalidad de hacerla más amigable al usuario y para agregar funcionalidad.

El alcance del IntrudEns Detection Killer puede ser tan grande como la imaginación y el conocimiento sean posibles llegar, para ellos es necesario invertir un recurso que es muy importante, tiempo.

Referencias

[ANDE1972] Anderson, James, P. *Computer Security Technology Planning Study.* ESD-TR-73-51, v II. Electronic System Division, Air Force Systems Command, Hanscom Filed, Bedford , MA, Octubre 1972.

[BACE2001] R. Bace, P. Mell. *Intrusion Detection Systems. NIST (National Institute of Standards and Technology),* Special Publication Agosto 2001. [Consultado en Agosto de 2005]. http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf

[DENN1986] Denning, Dorothy. *An Intrusion Detection Model.* Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, CA, Abril 1986.

[DGON2003] Gonzáles Gómez Diego. *Sistemas de Detección de Intrusos.* [Consultado en Agosto de 2005]. http://www.dgonzalez.net/pub/ids/html/

[EVEL1996] Martínez Martínez Evelio. *Telecomunicaciones y Redes, Introducción: Historia de las telecomunicaciones.* [Consultado en Agosto de 2005]. http://www.eveliux.com/fundatel/historia.html

[GOLL1999] Gollman, Dieter. 1999. *Computer Security.* Wiley.

[HEBR1995] Heberlein, Todd. *Network Security Monitor (NSM) – Final Report.* Lawrence Livermore National Laboratory, Davis, CA, Febrero 1995.

[KATH1991] Kathleen, Jackson. D. DuBois. C. Stallings, *An expert system application for network intrusion detection*, Proceedings of the 14th Department of Energy Computer Security Group Conference, 1991.

[KAUF2002] Kaufman, Charlie. Perlman, Radia. Speciner, Mike. 2002. *NETWORK SECURITY - PRIVATE Communication in a PUBLIC World.* Prentice Hall series in computer security and distributed systems.

[KILL2001] Killmeyer Tudor, Jan. 2001. *Information Security Architecture: an integrated approach to security in the organization.* Auerbach.

[NCSC1988] National Computer Security Center. *A Guide to Understanding Audit in Trusted Systems.* 1 Junio 1988. Versión 2. (Tan Book). [Consultado en Agosto de 2005] http://www.radium.ncsc.mil/tpep/library/rainbow/

[RIVA2003] Rivas López, José. Ares Gómez, José. Salgado Seguín, Víctor. 2003. *Linux: Seguridad Técnica y Legal.* Ediciones VirtuaLibro.

[SANT2005] De Los Santos, Sergio. *Sistemas de Detección de Intrusos*. [Consultado en Agosto de 2005] http://www.portalmundos.com/mundoinformatica/seguridad/deteccionintrusos.htm

[SMAH1988] Smaha, Steve E. *An Intrusion Detection System of the Air Force*. Proceedings of the Fourth Aerospace Computer Security Applications Conference, Orlando, FL, Diciembre 1988.

[SRII1990] SRI International. *Next-Generation IDES, History*. [Consultado en Agosto de 2005]. http://www.sdl.sri.com/programs/intrusion/history.html

[VILL2002] Villalón Huerta, Antonio. *Seguridad en Unix y Redes*. Consultado en Septiembre de 2005, en línea desde Agosto 08 del 2003]. http://es.tldp.org/Manuales-LuCAS/doc-unixsec/unixsec-html/node250.html

[ZIEG2000] Ziegler, Robert. 2000. *Guía Avanzada Firewalls Linux*. Prentice Hall.

[ZWIC2000] Zwicky, Elizabeth D. Cooper, Simon. Chapman, Brent. 2000. *Building Internet Firewalls, Second Edition*. O'Reilly.