ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

IMPLEMENTACIÓN DE UN SOFTWARE PARA EL MONITOREO Y OBTENCIÓN DE REPORTES DE UNA RED DE DATOS Y SU APLICACIÓN EN LA AUDITORÍA DE UNA RED OPERANTE.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

SANTIAGO JAVIER OÑATE CHÁVEZ

DIRECTOR: ING. PABLO HIDALGO

Quito, marzo 2007

DECLARACIÓN

Yo, Santiago Javier Oñate Chávez, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Santiago Javier Oñate Chávez

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Santiago Javier Oñate Chávez, bajo mi supervisión.

_____

Ing. Pablo Hidalgo
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco especialmente a Dios por darme la vida, a mis padres que por su esfuerzo y perseverancia me han guiado por el camino correcto y me han dado el regalo mas valioso, una educación de calidad conllevando a culminar mi carrera universitaria, a mis hermanos que han sido un ejemplo para mí en todas las cosas que realizan, a todos mis amigos que me han ayudado en las buenas y en las malas para solucionar los problemas que se presentan.

Agradezco a mi director de Proyecto de Titulación, Ing. Pablo Hidalgo ya que con su conocimiento y paciencia me ha sabido guiar en la realización de todo este proyecto.

A todos y cada uno de mis profesores que han sabido brindarme sus conocimientos a lo largo del tiempo de estudios que pase en esta prestigiosa universidad, especialmente al Ing. Xavier Armendáriz, quien me colaboró en la revisión de la parte correspondiente a su especialidad, en este Proyecto de Titulación.

A todos mis familiares que me han ayudado a llevar los estudios con la vida diaria, dándome consejos, alegrías y compañía, algo muy importante en el camino diario.

Santiago.

DEDICATORIA

Dedico este trabajo especialmente a mis padres, a mis hermanos y mis pequeños sobrinos.

A ellos.

CONTENIDO

DECLARACIÓN I
CERTIFICACIÓN II
AGRADECIMIENTOS III
DEDICATORIA IV
CONTENIDO V
RESUMEN IX
PRESENTACIÓN X

INTRODUCCIÓN A LA SEGURIDAD Y AUDITORÍA DE REDES
1.1 INTRODUCCIÓN A LAS REDES DE DATOS 1
1.1.1 ESTRUCTURA DEL MODELO OSI DE ISO 2
1.1.1.1 Capas del Modelo OSI 3
1.1.2 REDES DE DATOS 4
1.1.2.1 Características de las Redes de Área Local 5
1.2 INTRODUCCIÓN A LA SEGURIDAD DE REDES 6
1.2.1 ANTECEDENTES HISTÓRICOS DE SEGURIDADES DE REDES 9
1.2.1.1 Ataques más comunes 11
1.2.1.1.1 Negación del Servicio 11
1.2.1.1.2 Puertas Traseras 13
1.2.1.1.3 Escaneo 13
1.2.1.1.4 Spoofing 14
1.2.2 ESTADÍSTICAS DE SEGURIDAD 15
1.3 AUDITORÍAS DE SEGURIDAD 16
1.3.1 AUDITORÍAS TÉCNICAS 20
1.3.1.1 Pruebas de Caja Negra 21
1.3.1.2 Pruebas de Caja Blanca 22
1.4 SISTEMA DE DETECCIÓN DE INTRUSOS 23
1.4.1 CLASIFICACIÓN DE LOS NIDS 25
1.4.2 SENSORES DE UN IDS 26
1.4.2.1 Sensores Push 26
1.4.2.2 Sensores Pull 27

1.4.3 CONSOLA DE IDS…… 27
1.4.4 UBICACIÓN DE LOS IDS……27
1.4.5 ANÁLISIS DE LOS DATOS OBTENIDOS……..28
1.4.6 FALSOS POSITIVOS Y FALSOS NEGATIVOS……30
1.4.6.1 Falso positivo (false positive)……30
1.4.6.2 Falso negativo (false negative)……..30
1.5 ESPECIFICACIÓN DEL SOFTWARE…… 32
1.5.1 OBJETIVOS…… 32
1.5.2 RESULTADOS ESPERADOS…….. 32
1.5.3 DATOS DE ENTRADA PARA LA OPERACIÓN DE LOS MÓDULOS…...33
1.5.4 PROCESAMIENTO DE LOS DATOS DE ENTRADA……..33
1.6 ANÁLISIS COMPARATIVO CON SOLUCIONES COMERCIALES……..35

DISEÑO, DIAGRAMACIÓN Y CODIFICACIÓN DEL PROGRAMA
2.1 DIAGRAMACIÓN UML DE LOS MÓDULOS DEL SOFTWARE……39
2.1.1 FASE DE INICIO……40
2.1.1.1 Objetivo…… 40
2.1.1.2 Alcance……..41
2.1.1.3 Descripción……..41
2.1.1.4 Requerimientos específicos…….. 42
2.1.2 MODELO DE CASOS DE USO…… 44
2.1.2.1 Descripción de casos de uso……44
2.1.3 PAQUETE DE ANÁLISIS…….. 52
2.1.4 MODELO DE DISEÑO……..53
2.1.5 DIAGRAMA DE CLASES……..53
2.1.6 DIAGRAMA DE INTERACCIÓN……..55
2.1.7 DIAGRAMAS DE COMPONENTES…… 61
2.1.8 DIAGRAMA DE COLABORACIÓN…….. 63
2.1.9 DIAGRAMA DE NAVEGACIÓN……..65
2.1.10 DESCRIPCIÓN INTERFAZ DE USUARIO……..66
2.2 CODIFICACIÓN Y PROGRAMACIÓN DEL MÓDULO DE INTERFAZ DE CONFIGURACIÓN Y EJECUCIÓN DEL SISTEMA DE DETECCIÓN

DE INTRUSOS……..67
2.3 CODIFICACIÓN Y PROGRAMACIÓN DEL MÓDULO DE ESCANEO DE PUERTOS……71
2.4 REQUERIMIENTOS MÍNIMOS DE SOFTWARE Y HARDWARE DE LOS EQUIPOS……...74
2.4.1 REQUERIMIENTOS DE HARDWARE…… 74
2.4.2 REQUERIMIENTOS DE SOFTWARE……76
2.5 DOCUMENTACIÓN DEL SOFTWARE……..76

AUDITORÍAS, PRUEBAS DE FUNCIONAMIENTO Y ANÁLISIS DE RESULTADOS
3.1 AUDITORÍA DE LA RED OPERANTE……...78
3.1.1 ANÁLISIS DE LA RED DE PRUEBA……..79
3.1.1.1 Configuración de los sistemas operativos implantados: usuarios, ficheros……...81
3.1.1.2 La fiabilidad de las contraseñas utilizadas……...81
3.1.1.3 La configuración del software y hardware……82
3.1.1.4 Determinación de las vulnerabilidades presentes en los sistemas debido a la desactualización en la aplicación de “parches” de seguridad……82
3.1.2 ANÁLISIS DE LAS MÁQUINAS QUE CONFORMAN LA RED DE DATOS……83
3.1.2.1 Resumen de Máquinas de la red de datos…… 84
3.2 PRUEBAS DE FUNCIONAMIENTO DEL SOFTWARE DESARROLLADO EN LA RED……..87
3.2.1 ESTADÍSTICAS RELACIONADAS A LA UTILIZACIÓN DE PROGRAMAS P2P……90
3.2.2 RESULTADOS DE PRUEBAS DE MÓDULO ESCÁNER DE PUERTOS EN LAS MÁQUINAS DE LA RED……92
3.2.3 RESULTADOS DE PRUEBAS DE MÓDULO SISTEMA DE DETECCIÓN DE INTRUSOS EN LAS MÁQUINAS DE LA RED……94
3.3 ESTUDIO Y ANÁLISIS DE LOS RESULTADOS…….....109

3.4 ESTABLECIMIENTO DE NORMAS, PROCEDIMIENTOS Y POLÍTICAS DE SEGURIDAD……..111

CONCLUSIONES Y RECOMENDACIONES
4.1 CONCLUSIONES…….....119
4.2 RECOMENDACIONES…….. 122
REFERENCIAS BIBLIOGRÁFICAS……...125
ANEXOS……..…126

RESUMEN

El presente proyecto busca concientizar a los coordinadores, supervisores o encargados de las empresas dependientes de los recursos informáticos a precautelar el área de redes de datos, ya que se pueden tener problemas de incursión y ataques a las vulnerabilidades de los sistemas, causando grandes pérdidas económicas. Los problemas que pueden presentarse no solamente pueden ser externos sino también internos, esto último debido a la desinformación de los usuarios o empleados y una falta de definición de políticas de seguridad. Todo esto queda demostrado en la serie de pruebas realizadas y los resultados obtenidos.

El capítulo 1 presenta la introducción a las redes de datos, la estructura del modelo OSI y los principales parámetros de seguridades de redes; también se destacan los posibles ataques a que un sistema está expuesto, para por último presentar una serie de programas comerciales que se pudiesen ocupar para evaluar las seguridades existentes, resaltando las ventajas y desventajas de cada una de ellos.

El capítulo 2 muestra los pasos a seguirse para la elaboración del software de auditoría de red según el módulo (escáner de puertos o interfaz gráfica de configuración de los principales parámetros de *Snort*). Se realizan los diagramas *UML* necesarios, la codificación de los módulos (resaltando las principales funciones a utilizarse), para por último determinar los requerimientos mínimos de software y hardware que necesitará la aplicación.

El capítulo 3 indica los resultados de las pruebas del software desarrollado en la red, la auditoría hecha a la red operante y el análisis de los resultados obtenidos, para culminar con la definición de las políticas y normas de seguridad a seguirse en la organización.

PRESENTACIÓN

La dependencia empresarial de hoy en día de un sistema informático interconectado y la conexión de éste a una red insegura como es el Internet, hace importante la auditoría de vulnerabilidades de la red de datos y la definición de políticas de seguridad, ya que si existieran fallos en esta área de la empresa y no se tiene reglas claras de uso de los recursos, se podría tener problemas de virus informáticos o intrusiones no deseadas, conllevando a que se puedan perder datos que en algunos casos como entidades financieras o instituciones públicas, serían catastróficos.

Cada red sin importar la ubicación o tamaño, tiene vulnerabilidades y amenazas que pueden ser utilizadas por atacantes internos o externos a la organización, para dejar fuera de servicio a los equipos principales.

Las estadísticas determinan que diariamente se suscitan intentos de acceso no autorizados a equipos de interconectividad como *routers*; estos ataques son detectados básicamente por mensajes de autenticación fallida enviados por el dispositivo.

Por lo que es importante la utilización de un software para evaluar los posibles problemas y con esto poder definir las políticas de seguridad a seguirse, disminuyendo el riesgo de cualquier posible ataque interno o externo contra la operabilidad del sistema; todo lo dicho es aplicable a cualquier tipo de redes de datos, sean organizaciones pequeñas o grandes como centros de operación de red en donde al interconectar varios clientes son más propensos a experimentar ataques. De cumplirse esto se brindará a los usuarios y clientes confiabilidad, disponibilidad e integridad de la información y del sistema.

INTRODUCCIÓN A LA SEGURIDAD Y AUDITORÍA DE REDES

1.1 INTRODUCCIÓN A LAS REDES DE DATOS [1][2][4]

Para poder realizar una comunicación total entre las computadoras y elementos de red, se debe trabajar en un estándar de conectividad, para esto se genera una técnica de estructuración de la comunicación entre elementos de red.

Se pueden definir algunos aspectos importantes en la técnica de estructuración de la comunicación de los elementos de redes de datos.

Jerarquía de protocolos

Las redes de datos se deben organizar en capas, el número de capas, nombre, contenido y función de cada una varía en las diferentes redes; los servicios brindados entre las capas son dependientes unas de otras.

Protocolos

Son reglas y convenciones utilizadas en la comunicación de las capas de la máquina transmisora con las correspondientes capas de la máquina receptora de la información.

Procesos pares

Se definen como procesos pares cuando la operación realizada en una determinada capa es inversa a la realizada por su correspondiente en el lado de recepción o transmisión según sea el caso.

Interfaces

Es donde se definirán los servicios y operaciones primitivas entre capas adyacentes. Se definirán 4 tipos de primitivas:

- **Solicitud:** Una entidad desea que el servicio realice un trabajo
- **Indicación:** Una entidad es informada acerca de un evento.
- **Respuesta:** Una entidad desea responder a un evento.
- **Confirmación**: Una entidad va a ser informada acerca de su solicitud

No necesariamente tienen que estar presente las 4 primitivas ya que en algunos casos pueden omitirse algunas de ellas.

Arquitectura de red

Es el conjunto de capas y una jerarquía de protocolos, donde se tiene información suficiente para escribir un programa o desarrollar un hardware correspondiente a la capa en la cual va a operar.

1.1.1 ESTRUCTURA DEL MODELO OSI DE ISO

El objetivo perseguido por *OSI (Open System Interconnection)* es establecer una estructura que presenta las siguientes particularidades:

Se tiene una estructura multicapa con la idea de que cada capa se dedique a resolver una parte del problema de comunicación; esto significa que cada capa ejecuta funciones específicas, que servirán a la capa superior.

La capa superior utiliza los servicios de la capa inferior; cada capa se comunica con su similar en otras computadoras, pero debe hacerlo enviando un mensaje a través de las capas inferiores en la misma computadora. La comunicación entre capas está bien definida. La capa N utiliza los servicios de la capa N-1 y proporciona servicios a la capa N+1.

Tiene puntos de acceso entre las diferentes capas para que se pueda acceder a los servicios brindados por la capa correspondiente; cada capa es dependiente de la capa inferior y también de la superior, como se explicó anteriormente.

En cada capa, se incorpora al mensaje original una cabecera, esto se define como un elemento de control, que permite que una capa en la computadora receptora se entere de que su similar en la computadora emisora está enviándole información.

Por esta razón, se considera que un mensaje está constituido de dos partes: cabecera e información. Entonces, la incorporación de cabeceras es necesaria aunque representa datos de información extra, lo que implica que un mensaje corto pueda ser voluminoso y el rendimiento de la red baje ya que aumenta la latencia por el procesamiento; sin embargo, como la computadora destino retira las cabeceras en orden inverso a como fueron incorporados en la computadora origen, finalmente el usuario sólo recibe el mensaje original.

1.1.1.1 Capas del Modelo OSI

El modelo OSI tiene 7 capas, los principios que se aplicaron para llegar a siete capas son los siguientes:

1. Se debe crear una capa siempre y cuando se necesite un nivel diferente de abstracción.

2. Cada capa realiza una función bien definida.

3. La función de cada capa se debe definir en relación de los protocolos estandarizados internacionalmente.

4. Los límites de las capas deben elegirse de modo de minimizar el flujo de información a través de las interfaces.

5. La cantidad de capas debe ser suficiente para no tener que agrupar funciones distintas en la misma capa y lo bastante pequeña para que la arquitectura no se vuelva inmanejable.

APLICACIÓN
PRESENTACIÓN
SESIÓN
TRANSPORTE
RED
ENLACE
FÍSICA

Tabla 1.1 Capas del modelo OSI

1.1.2 REDES DE DATOS

En la mayoría de las empresas que recurren a la informática para satisfacer sus crecientes necesidades de información, se puede empezar con unos pocos computadores y unos cuantos periféricos. Poco a poco se van ampliando tanto los recursos de hardware como los recursos de software para la gestión de la información.

Esta ampliación suele llevar asociado un problema de redundancias, tanto de software, información y hardware. Por ejemplo, cada nuevo equipo va a necesitar de su propia impresora para generar informes (redundancia hardware); los datos almacenados en uno de los equipos es muy probable que sean necesarios en otro de los equipos de la empresa, por lo que será necesario copiarlos en éste (redundancia de información); los computadores que trabajen con los mismos

datos tendrán que tener los mismos programas para manejar dichos datos (redundancia software).

Todos estos problemas tienen una fácil solución la cual es una red *LAN (Local Area Network),* la cual va a permitir compartir bases de datos evitando la redundancia de información, programas evitando la redundancia software y periféricos como módem, impresora, escáner, evitando la redundancia hardware; adicionalmente pone a nuestra disposición otros medios de comunicación como pueden ser el correo electrónico y el *IRC (Internet Relay Chat)*.

Además una Red de Área Local conlleva ahorro de dinero ya que no es preciso comprar muchos periféricos, se consume menos papel, y si se tiene una sola conexión a Internet se la puede compartir a todos los computadores conectados a la red.

Las redes locales permiten interconectar computadores que estén dentro de un mismo edificio o en edificios colindantes, pero siempre teniendo en cuenta que el medio físico o inalámbrico que los une no puede tener más de unos miles de metros.

1.1.2.1 Características de las Redes de Área Local

- Altas velocidades de transmisión desde Mbps hasta Gbps
- Áreas pequeñas de cobertura.
- Servicios orientados y no orientados a conexión.
- Multiplexación estadística.
- No sensible al retardo

Como conclusión de lo expuesto anteriormente se pueden mencionar los principales beneficios del uso de las Redes de Área Local:

Compartición de periféricos, como son impresoras, plóters, módems, scanner.

Compartición de grandes cantidades de información mediante el empleo de gestores de bases de datos en red, facilitando el acceso y la actualización de los datos.

La red se convierte en un mecanismo de comunicación entre los usuarios conectados a ella, ya que permite el envío de mensajes mediante el empleo del correo electrónico, ya sea entre usuarios de la red de datos local o entre usuarios de otras redes o sistemas informáticos, programando reuniones o intercambiando ficheros de todo tipo.

Se aumenta la eficiencia de los computadores, poniendo a disposición del usuario todo un sistema que hace que las consultas sean más rápidas y cómodas.

Los sistemas operativos de red intentan dar la sensación de que los recursos remotos a los que accede el usuario son locales al computador desde el cual está trabajando el usuario. Por ejemplo, un usuario puede estar consultando la información de una base de datos, y en ningún momento tiene conocimiento de si la información a la cual está accediendo se encuentra en su propio computador o en otro distinto dentro de su red local o en cualquier otra parte del mundo.

Cabe mencionar que se puede conectar computadores separados por grandes distancias, esto se hace mediante el uso de las redes de área extensa *WAN (Wide Area Network)*, las cuales se sirven de otras redes de comunicaciones como puede ser la red telefónica o una red dedicada de datos para transmitir información entre los computadores.

1.2 INTRODUCCIÓN A LA SEGURIDAD DE REDES

Hoy por hoy las organizaciones son cada vez más dependientes de sus redes informáticas y un problema que las afecte, por mínimo que sea, puede llegar a comprometer la continuidad de las operaciones y causar pérdidas económicas grandes; es por eso que las seguridades informáticas son un pilar fundamental

para mantener en un sistema la confidencialidad, disponibilidad e integridad de la información que es lo mas importante.

Confidencialidad

Se trata de prevenir el acceso no autorizado a la información sensible que es manejada por la empresa; el acceso no autorizado podría ser intencional como romper el algoritmo de cifrado y leer la información o no intencional en donde los individuos pueden acceder a la información por descuido de los administradores de la red de la misma empresa.

Integridad

Es la capacidad de garantizar que la información no ha sido alterada, es decir la capacidad de proteger la información o datos de alteraciones no autorizadas; se deben alcanzar tres aspectos importantes:

- Prevención de la modificación de la información por usuarios no autorizados.
- Prevención de cambios no autorizados y no intencionales por usuarios autorizados.
- Prevención de la consistencia interna y externa de la información.

En el caso de la consistencia interna se considera que los datos internos sean consecuentes con la realidad de la empresa u organización; en el caso de la consecuencia externa se quiere que los datos almacenados en las bases de datos sean consecuentes con el mundo real.

Disponibilidad

Se refiere a que usuarios autorizados tengan acceso oportuno a la red, sistema, hardware y software permitidos por el administrador y que éstos sean confiables y puedan recuperarse rápida y completamente de un evento de interrupción si es que ocurriere.

Identificación

Es la capacidad de identificar al usuario de un sistema de información; el usuario profesa su identidad al sistema mediante un *Login ID (Identification)*

Autenticación

Es el proceso por el cual el usuario se identifica de forma unívoca y en muchos casos sin la posibilidad de repudio, se hace la verificación de la identidad demanda por el usuario; en algunos casos se puede utilizar contraseña, pruebas biométricas y hasta combinaciones de éstas.

Autorización

Es el proceso mediante el cual se da acceso a una persona al sistema, refiriéndose a los privilegios que tiene éste dando o no el acceso a un recurso en especial.

Responsabilidad

Es la determinación de las acciones y del comportamiento de un individuo dentro de un sistema y la responsabilidad de sus acciones.

1.2.1 ANTECEDENTES HISTÓRICOS DE SEGURIDADES DE REDES

En los años 80 donde el Internet comenzaba a crecer de manera exponencial con varias cantidades de redes que iban adhiriéndose para compartir en la mayoría de los casos información y recursos, es ahí cuando surgen diversos problemas que comprometen los factores como la disponibilidad, integridad, confidencialidad. Uno de los primeros problemas fue el gusano de Internet, generado en el año de 1988 por Robert T. Morris donde miles de computadores quedaron inutilizados durante varios días, lo que causó pérdidas económicas millonarias.

Durante 1997, el 54 por ciento de las empresas norteamericanas sufrieron ataques de *Hackers* en sus sistemas. Las incursiones de los piratas informáticos, ocasionaron pérdidas totales de 137 millones de dólares en ese mismo año [5]. El Pentágono, la CIA, UNICEF, la ONU y demás organismos mundiales han sido víctimas de intromisiones por parte de estas personas que tienen muchos conocimientos en la materia y también una gran capacidad para resolver los obstáculos que se les presenta.

Un *Hacker* puede tardar meses en vulnerar un sistema ya que son cada vez más sofisticados. Es por eso que los administradores de las redes empezaron a tener en cuenta ciertos aspectos básicos, para elaborar seguridades de redes informáticas. De esta manera se comienzan a reunir grupos de investigación como es el caso de DARPA (*Defense Advanced Research Projects Agency*) que creó el CERT (*Computer Emergency Response Team*), un grupo formado en su mayor parte por voluntarios calificados de la comunidad informática, cuyo objetivo principal es facilitar una respuesta rápida a los problemas de seguridad que afecten a los computadores de Internet.

Como es lógico hasta la fecha se han ido creando otros grupos de investigación y de trabajo relacionado con las seguridades de redes informáticas, pero a la par de esto surgen grupos que quieren atacar y vulnerar las seguridades informáticas.

Los problemas siguen incrementándose ya que en años anteriores la información que atenta contra la seguridad era limitada, hoy en día se tiene grandes cantidades de información y hasta programas que con mala suerte un aprendiz de *hacker* puede ocupar para atacar a un servidor desprotegido y dejarlo fuera de operación sin la necesidad de tener conocimientos amplios de redes.

Un punto importante de recalcar es que entre 500 corporaciones y agencias de Gobierno, el 90% detectaron intrusiones en su red, 80% reconocieron pérdidas económicas como consecuencia, 40% cuantificaron su pérdida financiera equivalente a $455.848.000 dólares que asciende anualmente a más de 20 millones de dólares, cifra que cada año se incrementa en más del 35% [6].
Si no se toman medidas pertinentes se pueden producir verdaderos desastres que en muchos casos puede llevar a la desaparición de aquellas empresas con altísimo grado de dependencia tecnológica (bancos, servicios automatizados).

Muchos ataques de este tipo comienzan con ingeniería social[1], y la falta de cultura por parte de los usuarios para facilitar a extraños sus identificaciones dentro del sistema.

Esta primera información es usualmente conseguida a través de una simple llamada telefónica, otros pueden ser realizados por empleados internos que abusan de sus permisos de acceso, por atacantes externos que acceden remotamente o interceptan el tráfico de red vía *Sniffer* o analizadores de tráfico, lo que hace importante en las organizaciones una definición de políticas de seguridad.

Políticas de seguridad

El término política de seguridad es el conjunto de requisitos definidos por los responsables directos o indirectos de un sistema informático que indica en términos generales qué está y qué no está permitido en el área de seguridad durante la operación general de dicho sistema. Al tratarse de términos generales, aplicables a situaciones o recursos muy diversos, suele ser necesario refinar los

[1] ***Ingeniería Social:*** En terminología *hacker*, hacer Ingeniería Social es persuadir a otra persona (a través de diversas formas) para que de un dato útil para cumplir un objetivo. Este dato puede ser un *password*, un código, etc. incluso pueden dar nombres de familiares y fechas importantes, que siempre se usan para formar claves.

requisitos de la política para convertirlos en indicaciones precisas de qué es lo permitido y lo denegado en cierta parte de la operación del sistema, lo que se denomina política de aplicación específica [3].

1.2.1.1 Ataques más comunes

Mediante algunos sistemas se puede saber estadísticamente los ataques más comunes que se pueden presentar en las redes; la mayoría de los ataques se centran en alterar registros de bases de datos y creación de puertas traseras para que los atacantes puedan hacer de las suyas de una manera fácil e inesperada. El análisis de vulnerabilidades se ocupa de la detección de tales puertas traseras y cerrarlas, para que no puedan hacer uso de las mismas y con esto precautelar los datos de la organización.

En los primeros años, los ataques involucraban poca sofisticación técnica. Los atacantes internos (empleados no conformes o personas externas con acceso a recursos dentro de la empresa) utilizaban sus permisos para alterar archivos o registros. Los atacantes externos (personas que atacan desde afuera de la ubicación física de la organización) ingresaban a la red simplemente averiguando un *password* válido.

A través de los años se han desarrollado formas cada vez más sofisticadas de ataque para explotar las vulnerabilidades en el diseño, configuración y operación de los sistemas que se encuentran operando. Estos nuevos métodos de ataque han sido automatizados, por lo que en muchos casos sólo se necesita conocimiento técnico básico para realizarlos; con esto el atacante puede adquirir derechos a lugares que le permitan dejar un virus o bombas lógicas para paralizar todo un sistema antes de huir del sistema atacado.

1.2.1.1.1 Negación del Servicio

En este tipo de ataques lo que se trata es de privar del uso de recursos de datos a los usuarios legítimos; con varios intentos de acceso hacia un servidor, el intruso

puede hacer cada vez más lenta la capacidad de procesamiento del servidor y con ello no pueda atender a solicitudes de usuarios legítimos que quieren realizar o están operando en la red afectada.

Un típico ataque de negación de servicio es el *Ping* de la muerte en donde el atacante envía una serie de peticiones de *Ping* al servidor atacado, éste debe responder a cada una de estas solicitudes y cuando llega al límite de su procesamiento el sistema entero comienza a colapsar y se pierde conectividad con el computador afectado.

Otra clase de ataque de negación de servicio que la mayoría de proveedores de Internet han sufrido llegando a bajas temporales del servicio, son aquellos ataques que explotan el protocolo *TCP (Transmission Control Protocol)*; en este ataque el atacante satura el sistema con mensajes que requieren establecer conexión.

Sin embargo, en vez de proveer la dirección *IP* del emisor, el mensaje contiene falsas direcciones *IP*, donde el sistema responde al mensaje, pero como no recibe respuesta, acumula el *buffer* con información de las conexiones abiertas, no dejando lugar a nuevas conexiones legítimas.

Toca recalcar que se pueden presentar ataques de negación de servicios distribuidos, esto se da cuando se presentan ataques de negación de servicios a más de un *Host* en la red; estos *Host* pueden estar distribuidos en diferentes partes geográficamente.

Ejemplos de Negación de servicio:

- Sobreflujo de *Buffer*
- Ataques de sincronismo
- *Teardrop Attack*
- *Smurf*
- *Spam*

Otro modo conocido y explotado de interrumpir el servicio de un sistema es el uso del *Spam*, esto se da cuando un usuario recibe correos electrónicos en cantidades grandes sin que éste los haya pedido; en este caso es importante definir cuál de los mensajes recibidos en la bandeja de correo son legítimos y cuáles no, para irlos desechando si es del caso.

El *IDS* (*Intrusion Detection System*) es una herramienta de gran importancia en este tipo de ataques ya que puede discriminar entre correo legítimo y *Spam*.

1.2.1.1.2 Puertas Traseras

Un ataque de puerta trasera se da mediante conexiones con *Dial up* o conexiones asincrónicas a la red de datos, de manera que no se pase la conexión por los mecanismos de control.

1.2.1.1.3 Escaneo

Ésta es una manera de ataque mediante la cual los atacantes pueden tener información de diferentes parámetros:

- Parámetros de los sistemas de redes.
- Actividades de los *Host.*
- Tipos de sistemas de seguridad en la red.
- Tipos de recursos.
- Tipos de servicios.
- Sistemas operativos utilizados.
- Vulnerabilidades presentes en la red (puertos abiertos).

Los escáners de puertos son los recursos más apropiados para realizar las actividades antes mencionadas.

Cabe mencionar que el uso de escáner de puertos puede ser utilizado de manera positiva o negativa, ya que si se lo utiliza para auditar una red viendo qué puertos están abiertos innecesariamente dará la pauta para cerrarlos y evitar posibles ataques; caso contrario puede ser utilizado para buscar puertos abiertos para realizar ataques.

1.2.1.1.4 Spoofing

El *Spoofing* se utiliza para hacer creer al sistema que se está comunicando con una entidad conocida, lo cual implica alteraciones del paquete en la capa *TCP*; esto es, un atacante puede modificar o crear un paquete *TCP* y enviarlo al *Host* atacado, para de esta manera establecer una sesión con un usuario no autorizado.

Como existen varias clases de *Spoofing,* se van a mencionar las más importantes y lo que implica cada una de ellas:

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

Este método es muy utilizado para capturar nombres de usuario, *ID* y contraseñas de usuarios, que generalmente viajan en claro (sin encriptar) al ingresar a sistemas de acceso remoto *RAS.* Cuando son utilizados para capturar números de tarjetas de crédito y direcciones de *e-mails* entrantes y salientes se le conoce como *Eavesdropping,*

Hay algunos casos en donde además de interceptar el tráfico de red, el atacante ingresa a los documentos, mensajes de *e-mail* y otra información guardada,

realizando una copia de la información a su propia computadora; esto puede ser realizado por simple curiosidad, pero también es realizado con fines de espionaje y robo de información, lo cual se le conoce como *Snooping.*

Cuando existe modificación desautorizada a los datos, esto es borrado de archivos y alteración de documentos se lo conoce como *Tampering.* Un ejemplo claro de esto es cuando las víctimas sufren el cambio de su página Web por imágenes terroristas o humorísticas; o el reemplazo de software por otros con el mismo nombre pero que incorporan código malicioso (virus, troyanos).

Cuando se utiliza para actuar en nombre de otros usuarios, usualmente para realizar tareas, el intruso normalmente utiliza un sistema para obtener información e ingresar en otro, y luego utiliza éste para entrar en otro, y en otro. Este proceso, llamado *Looping*, el cual tiene la finalidad de evaporar la identificación y la ubicación del atacante. El camino tomado desde el origen hasta el destino puede tener muchas estaciones, que exceden hasta los límites de un país.

1.2.2 ESTADÍSTICAS DE SEGURIDAD

Los ataques que se producen en Internet han ido aumentando conforme pasan los años, es por eso que se tiene que hacer pruebas de seguridad para ver por donde el atacante puede entrar y hacer daño y para poder diseñar un sistema de seguridad confiable.

La figura 1.1 muestra como han ido incrementando el número de *Host* y los números de incidentes de seguridad por año, según datos recogidos por la agencia *CERT.*

Según se puede observar en los datos de la tabla 1.2, hay un incremento en los incidentes de seguridad en el transcurso de los años [7].

AÑO	# INCIDENTES REPORTADOS
1995	171
1996	345
1997	311
1998	262
1999	417
2000	1090
2001	2437
2002	4129
2003	3784
2004	3780
2005	5990
2006	3997

Tabla 1.2 Número de Incidentes de Seguridad reportados por año

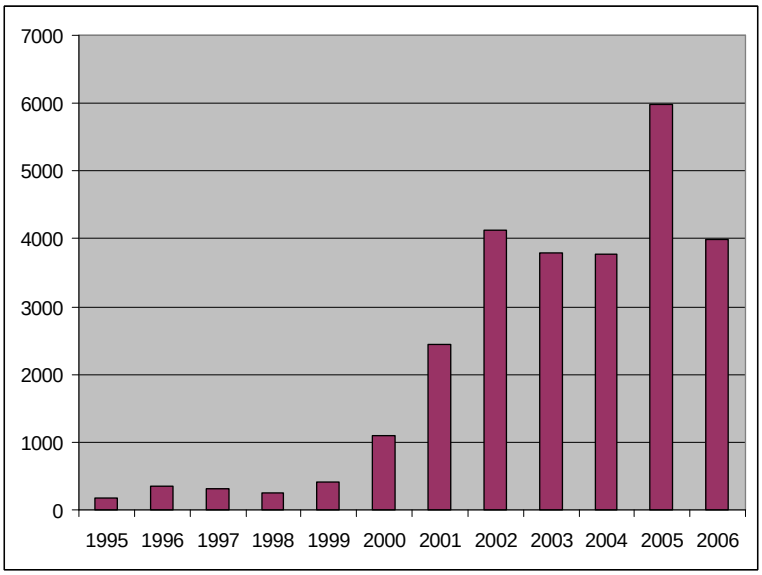

Fig. 1.1 Cuadro de Estadísticas de incidentes de seguridad

1.3 AUDITORÍAS DE SEGURIDAD

La auditoría de seguridad es algo cada vez más frecuente dentro de las empresas, debido a la obligatoriedad de su realización; sin embargo, en muchos casos, se desconocen todos los aspectos relacionados con una auditoría de seguridad o incluso los distintos tipos que existen.

Para conocer los parámetros importantes de la auditoría de seguridad de redes de datos, se va a mencionar el estándar *OSSTMM* donde se explicará los parámetros más importantes.

OSSTMM (O*pen Source Testing Methodology Manual*) [3]

Este estándar presenta de manera clara y precisa todos los principales procedimientos que debe hacer y tener en cuenta un auditor de seguridad de redes informáticas, para lograr realizar un buen trabajo de auditoría.

Un buen auditor debe tener en cuenta durante el proceso de auditoría los siguientes aspectos:

- Los bienes que se pueden acceder y en qué momento pueden provocar el máximo riesgo de seguridad.

- Dónde y cuando se encuentran la mayor parte de las vulnerabilidades.

- Cuándo se está más propenso a aplicar confidencialidad, integridad y accesibilidad a la prueba.

Manteniéndose sistemático y persistente, el efecto acumulativo de estas pruebas dará como resultado un panorama exacto de los riesgos, debilidades, filtraciones de información y vulnerabilidades.

Los siguientes puntos ayudarán adecuadamente a crear y llevar a cabo las pruebas de seguridad con altos estándares de calidad.

¿Qué auditar y porqué auditar?

El esperar para hacer la prueba, así como esperar para reportar los problemas y solucionarlos, es un error. Cuando se deja la casa al irse de vacaciones, ¿se

espera hasta su retorno para asegurarse el haber cerrado con llaves las puertas? La respuesta es no. Se echó llave a la puerta e intentó abrirla para asegurarse de que estaba bien cerrada. Esperar hasta su retorno para comprobarlo requeriría también examinar la casa para verificar qué cosas faltan.

Probar las cosas pequeñas, porque en definitiva, todas son cosas pequeñas

Se deben probar todos los detalles, ya que muy a menudo los pequeños detalles llevan a las más importantes fallas de seguridad. Además, es la acumulación de las cosas pequeñas, que individualmente no representan mucho riesgo, las que pueden llevar a una falla de seguridad

Hacer más con menos

Mientras los presupuestos de seguridad sigan siendo bajos, el auditor de seguridad necesita operar con eficiencia y creatividad para hacer más en menos tiempo. Si una prueba ineficiente de seguridad se vuelve demasiado costosa, es tentador para una organización ver esta prueba de seguridad como un costo innecesario, y peor aún si no se lograron los resultados esperados.

Esto es algo desafortunado porque los riesgos asociados con no llevar a cabo estas pruebas de seguridad siguen siendo desconocidos. En consecuencia, cuando se balancea minuciosidad y eficiencia en las pruebas de seguridad, los resultados van a hablar por sí mismos una y otra vez y muchas otras organizaciones verán las pruebas de seguridad como un arma de costo justificado en su actitud defensiva.

No subestimar en ninguna forma la importancia de las políticas de seguridad

La política es la declaración oficial de la compañía con respecto a los objetivos que quiere lograr. Muy poca gente llega alguna vez a alguna parte sin antes

desarrollar políticas de seguridad. Una política de seguridad expresa en su contenido la intención y los objetivos de seguridad de una organización.

Las políticas de seguridad de una organización son con frecuencia muy complejas y con muchas personas involucradas en su desarrollo y su mantenimiento. Los errores de políticas de una sección frecuentemente derivarán en un efecto de flujo negativo que impactará en otras secciones. Por ejemplo, si la política no está implementada para especificar controles que verifiquen que la gente no se vaya llevando cajas o equipamiento, entonces una filtración de información puede ocurrir.

Las Políticas de Seguridad especifican muchos más controles que tienen un efecto directo sobre los estándares y procedimientos, tales como las reglas de salida que existen en un *router* de filtrado, o cuáles correos electrónicos el usuario puede reenviarlos desde dentro de la compañía.

Un auditor requiere cierta competencia en dirección de proyectos, discernimiento acerca de las necesidades del cliente y tener excelentes destrezas comunicativas, informando el tiempo necesario para que la prueba sea realizada y la existencia de suficientes recursos en el presupuesto para reparar las vulnerabilidades descubiertas.

El resultado final de la prueba de seguridad será entregado al cliente o a la administración del cliente con todos los factores financieros previstos de antemano.

Las auditorías de seguridad se pueden clasificar en técnicas, que se centran en los riesgos existentes en los sistemas de información de la organización y en la calidad técnica de las medidas de protección introducidas como son la correcta configuración de los equipos, servidores, sistemas, y no técnicas, que habitualmente estudian el cumplimiento efectivo de la Política de Seguridad de la organización y de sus procedimientos.

1.3.1 AUDITORÍAS TÉCNICAS

Dependiendo de la profundidad de los trabajos a realizarse, se habla de auditorías de vulnerabilidades, que tratan de localizar configuraciones erróneas con exceso de agujeros de seguridad en el software, las que pueden ser ocupadas por los atacantes para dejar fuera de servicio un sistema.

Con el apoyo de herramientas que automatizan parte del trabajo y proyectos de *hacking* controlado, las pruebas de intrusión o auditorías a nivel de aplicación tratan de explotar los errores de programación, la arquitectura de red, las debilidades de los protocolos de comunicación, y los controles de acceso.
Se trata de simular los ataques a una infraestructura de red bajo los siguientes perfiles:

- Atacante externo con distinto nivel de calificación.
- Usuario interno.
- Auditor.
- Administrador.
- Competencia.
- Bajo las mismas circunstancias y capacidades (información inicial, puntos de acceso, recursos disponibles).

Por otro lado, dependiendo de la aproximación que se tome para realizar las auditorías técnicas, se habla de pruebas de caja negra, que buscan las debilidades desde el exterior de los sistemas (habitualmente realizadas de forma remota, desde Internet), y pruebas de caja blanca, que realizan una revisión de seguridad analizando la configuración del propio sistema, con acceso al mismo.

Una auditoría de seguridad de caja negra normalmente comienza con trabajos desde el exterior, para encontrar puntos débiles y ganar algún tipo de acceso a los sistemas, y una vez conseguido este acceso, examinar el sistema para escalar

privilegios y tomar control sobre él. Estas pruebas desde hace tiempo se vienen realizando basándose en el estándar *OSSTMM* mencionado anteriormente.

En el caso de una auditoría de caja blanca el objetivo no es lograr el acceso (la empresa lo proporciona para realizarla) sino revisar las medidas de seguridad implementadas en el sistema y su conformidad con estándares reconocidos, como por ejemplo, los trabajos del instituto de estándares norteamericanos *NIST (National Institute of Standards and Technology), el CSI (Center of Internet Security) o del SANS Institute.*

1.3.1.1 Pruebas de Caja Negra

Las pruebas de caja negra, para que sean realmente efectivas, deben realizarse sin ningún conocimiento de la infraestructura, garantizando de esta forma que el análisis no tratará de utilizar ningún tipo de información que facilite la tarea.

El propósito de estas pruebas es que el auditor se comporte como si realmente fuese un atacante de la infraestructura. Durante un análisis de caja negra normalmente se llevarán a cabo pruebas de visibilidad (para conocer los servicios y versiones de estos activos y visibles desde el exterior en cada uno de los sistemas), pruebas de identificación de servicios (para determinar qué programas ofrecen los servicios ofrecidos, a través de las cabeceras obtenidas o respuestas programáticas y no fiándose de la lista de puertos *TCP/IP* conocidos), obtención de información (recuperación de información o datos de configuración del sistema final o sistemas adyacentes que desvelen detalles de la infraestructura auditada) y pruebas de vulnerabilidades en software estándar.

Estas últimas pruebas son las más complejas y se realizan una vez determinados los servicios que están “corriendo”, junto con la información disponible de versiones y sistemas operativos. Se basan en una parte que puede ser realizada por herramientas de diagnóstico automáticas y otra parte que debe ser realizada de forma manual por el auditor. Esta fase tiene que realizarse con ciertas precauciones puesto que son frecuentes los casos en que las pruebas de

vulnerabilidades que puedan tener éxito producen cortes de servicio o caídas en los sistemas auditados.

Una vez que se ha conseguido penetrar con éxito en un sistema, la auditoría de caja negra puede continuar hacia otros sistemas adyacentes (generalmente más expuestos una vez traspasado el perímetro) y también derivar hacia análisis de caja blanca.

1.3.1.2 Pruebas de Caja Blanca

Por el contrario, las pruebas de caja blanca, examinan el sistema desde su interior. Por lo tanto es necesario tener acceso a los sistemas, el cual generalmente se obtiene porque directamente se le proporciona al auditor un acceso al equipo para que pueda realizar un análisis en profundidad de la configuración del sistema, aunque en algunos casos una prueba de caja negra se convierte en caja blanca por haber logrado un acceso al sistema a través de alguna vulnerabilidad del mismo u obtener información que pueda analizarse de esta forma (código fuente de las aplicaciones utilizadas).

Es importante destacar que estas pruebas son complementarias a las anteriores, ya que el hecho de no haber encontrado vulnerabilidades en las pruebas de caja negra, no significa que no las haya, si no que generalmente no se han dedicado recursos suficientes a descubrirlas.

En otras palabras, el hecho de que un sistema sea o no vulnerable no radica en que se encuentre una vulnerabilidad, si no en que exista dicha vulnerabilidad. Siguiendo con esta filosofía, es necesario ampliar la información que se posee sobre los sistemas al máximo, incluyendo topología, protocolos utilizados, reglas en los *firewalls*, software empleado; así, durante esta fase normalmente se realizan las siguientes tareas:

- Análisis de la configuración de todos los sistemas operativos implantados: usuarios, ficheros.

- Análisis de la fiabilidad de las contraseñas utilizadas.

- Análisis de la configuración del software de base (*Web*, *Mail, firewall*, etc.).

- Análisis del código fuente de las aplicaciones instaladas o desarrolladas a medida.

- Determinación de las vulnerabilidades presentes en los sistemas debido a la desactualización en la aplicación de parches de seguridad.

Es por ello que la actividad de auditoría de procesos y gestión estará precedida por un trabajo de análisis del entorno de la organización incluyendo:

- Análisis de la política de seguridad.

- Análisis de los procesos implantados, incluyendo, entre otros, los de gestión y administración.

- Entrevistas con los responsables de la seguridad de información de la organización para determinar los controles establecidos

1.4 SISTEMA DE DETECCIÓN DE INTRUSOS [8]

Ya teniendo en cuenta los principales problemas de las redes de datos, actualmente se puede decir que una manera para prevenir y controlar las posibles intrusiones no deseadas es el Sistema de Detección de Intrusos o mas conocido

por sus siglas en inglés *IDS* (*Intrusion Detection System*) que es un modelo de seguridad aplicable tanto a computadores como a redes de datos.

Un sistema *IDS* recolecta y analiza información procedente de distintas áreas de un computador o red de computadores con el objetivo de identificar posibles fallos de seguridad. Este análisis en busca de intrusiones incluye tanto los posibles ataques externos (desde fuera de nuestra organización) como los internos (debidos a un uso abusivo o fraudulento de los recursos) como se explicó anteriormente.

Los sistemas *IDS* suelen utilizar técnicas de análisis de vulnerabilidades, es decir examinan todos los componentes en búsqueda de alguna vulnerabilidad. Como se mencionó antes el *IDS* puede aplicarse solo a computadores o a redes de computadores para el respectivo análisis.

En el caso de computadores se realiza a nivel de sistema operativo para controlar los accesos de usuarios, modificación de ficheros del sistema, uso de recursos (CPU, memoria, red, capacidad de disco duro) con el objetivo de detectar cualquier comportamiento anómalo que pueda ser indicativo de un abuso del sistema; a esto se le conoce como *HIDS* (*Host Intrusion Detection System*).

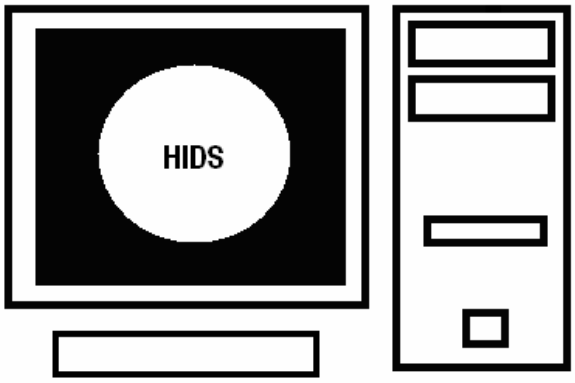

Fig. 1.2 Esquema de un HIDS

Un ejemplo de aplicación de este tipo de herramientas en sistemas operativos de computadores puede ser el conocido software comercial *TRIPWIRE*.

En el caso de redes de computadores se puede monitorizar el uso de ancho de banda, acceso a o desde direcciones no permitidas, uso de direcciones falsas, con el objetivo de encontrar un comportamiento anómalo; a esto se le conoce como *NIDS* (*Network Intrusion Detection System*).

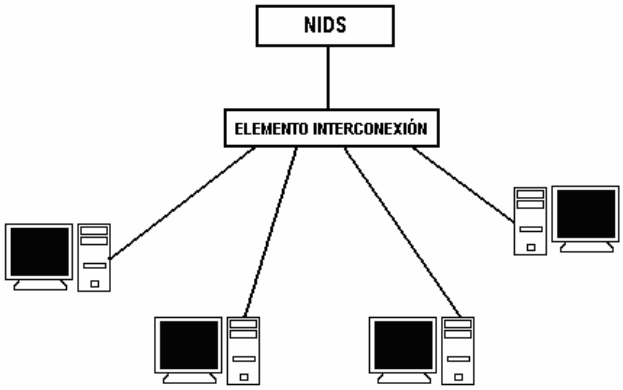

Fig. 1.4 Esquema de un NIDS

Realizar la supervisión de una red de comunicaciones implica necesariamente hacer un estudio detallado y pormenorizado de todo el tráfico que circula por ésta. Si se hace unos simples cálculos se puede observar que resulta del todo imposible filtrar toda la información que circula por la red en tiempo real.

1.4.1 CLASIFICACIÓN DE LOS NIDS

Según la aproximación utilizada para la monitorización del tráfico de la red, se agrupa los *NIDS* en tres clases distintas

Basados en firmas (*signature based nids*)

Al igual que muchos antivirus, estos *IDS* se basan en la búsqueda de patrones conocidos como firmas en el tráfico de la red.

Basados en anomalías (*anomaly based nids*)

Se basan en analizar el tráfico de la red creando estadísticas y asignándoles pesos. Cuando se detecta tráfico sospechoso, se confronta con las estadísticas anteriores y en función del peso asignado y la cantidad de ocurrencias del evento se dispara alguna alarma o no.

Modelo de protocolo (*protocol modeling nids*)

Estos sistemas de detección de intrusos buscan paquetes que contengan anomalías o configuraciones poco comunes de los protocolos de red; como se conoce los datos van encapsulados en distintos datagramas de distintos protocolos.

En un sistema de detección de intrusos básico se tienen dos elementos fundamentales los sensores y la consola.

1.4.2 SENSORES DE UN *IDS*

Los sensores de un *IDS* son elementos pasivos que examinan todo el tráfico de su segmento de red en búsqueda de eventos de interés. Dependiendo del paradigma que utilicen para comunicarse con la consola del sistema, el detector de intrusos pueden clasificarse en dos grupos:

1.4.2.1 *Sensores Push*

Cuando se detecta un evento de interés el sensor crea un paquete de datos que envía a la consola. Un protocolo muy utilizado con este tipo de sensores es el *SNMP* (*Simple Network Management Protocol*) que permite la definición de *Traps* para el envío de información a un receptor (la consola en este caso).

Su principal inconveniente es que pueden ser observados por el atacante para descubrir cómo ha sido configurado y ante qué tipo de patrones reacciona, lo que

permite establecer los patrones que ignora el sensor y por tanto cuáles emplear en un ataque.

1.4.2.2 *Sensores Pull*

Almacenan los eventos de interés hasta que la consola pregunta por ellos al sensor. Si se establece un protocolo de intercambio de mensajes cifrado y se pregunta regularmente a los sensores, puede ofrecer un mecanismo eficaz de comunicación con la consola. Para evitar susceptibilidades, en caso de que el sensor no detecte eventos incluirá una cadena aleatoria de caracteres para evitar enviar siempre el mismo paquete cuando no existen eventos detectados.

1.4.3 CONSOLA DE *IDS*

La consola de un sistema de detección de intrusos se encarga de recibir toda la información de los sensores *pull* o *push* y presentarla de forma tabulada y ordenada al auditor. Desde la consola se pueden configurar los distintos sensores así como emprender acciones en respuesta a los datos recibidos de los sensores.

1.4.4 UBICACIÓN DE LOS *IDS*

Una vez explicados los componentes básicos de un sistema *IDS*, se decidirá de qué tipo serán los distintos sensores que utilizará el *NIDS pull* o *push*, y finalmente se concretará en qué lugar o lugares de la red deben colocarse.

Antes del *firewall*

Esta arquitectura se basa en detectar todos los paquetes que llegan a la red antes de ser filtrados por el *firewall*. De esta forma, se realiza una búsqueda de eventos de interés en todo el tráfico recibido sin interferencias de filtros del *firewall.*

En el *firewall* o después del *firewall*

Consiste en situar el sensor en el propio *firewall* o a la salida del mismo, de esta forma se evitan ataques de intrusos a los sensores externos y se eliminan muchos resultados que pudieran ser falsos, ya que se procesa únicamente el tráfico que el *firewall* deja pasar.

Antes del *firewall* y después del *firewall*

Es la opción más costosa pero que ofrece mayor seguridad, ya que permite obtener lecturas del tráfico total y del tráfico filtrado por el *firewall.* Permite examinar la configuración del *firewall* y comprobar si filtra correctamente o no.

Hay que recalcar que un *IDS* es un complemento a un *firewall* y no con esto se quiere decir que si se coloca y se configura correctamente un *IDS* en un sitio de entrada ya no se necesitaría un *firewall* ni viceversa.

Esto es por que el trabajo realizado por uno y otro es diferente, pero complementario para poder brindar una mayor seguridad, el objetivo primordial es el llegar a un 100% pero nunca se podrá tener este valor sino uno muy próximo.

1.4.5 ANÁLISIS DE LOS DATOS OBTENIDOS

Si se almacenarían todo los datos que circulan en la red se necesitarían discos de gran capacidad, es por eso que haciendo filtraje de tráfico se disminuye substancialmente esta cantidad; sin embargo, no solamente interesa el tráfico registrado en un solo día, sino que interesan históricos de la semana, mes o incluso de años.

Toda esta cantidad de información puede ser almacenada de forma óptima para ser consultada ágilmente, de otra forma, dejará de ser utilizada y por lo tanto ser útil, es por eso que los *IDS* incorporan soporte de bases de datos.

Sea cual sea el formato usado para el almacenamiento de la información debe cumplir las siguientes características: realizar una reducción importante del volumen de datos pero conservando la información importante, poseer un formato compacto, fácil de consultar, escribir y actualizar; debe permitir la interrelación de todos los datos entre sí.

Los datos deberán ser almacenados en un nodo de datos cuádruple que contiene los siguientes campos (Fecha, Dirección origen, Dirección de destino, Tipo de protocolo), de manera de ir compactando la información registrada.

Los principales objetivos de almacenar los datos históricos de tráfico que se presenta en la red son:

1. Realizar informes y estadísticas sobre incidentes en la red.

2. Conocer cuando un ataque presenta características similares a otro recibido anteriormente, ya que se puede obtener información de cómo reaccionar ante él de preferencia en tiempo real evitando ser pasivos.

Otro aspecto sumamente importante que hay que definir es acerca de las correlaciones, que es la relación mutua entre dos o más elementos de un conjunto dado. De esta forma, gran parte de las consultas a las bases de datos se basarán únicamente en buscar correlaciones entre los eventos de interés detectados y los datos históricos almacenados.

Correlaciones por dirección de origen: Se basan en encontrar similitudes entre conexiones provenientes de la misma fuente.

Correlaciones por dirección de destino: Considera las conexiones que tienen como destino la misma dirección *IP (Internet Protocol)*

Correlaciones de firmas: Se basan en buscar conexiones desde o hacia un puerto determinado (muchos virus y programas de *backdoor* basan su comunicación en puertos no estándar).

También se buscan configuraciones no estándar de los distintos campos del paquete *IP*, como por ejemplo las banderas. Varios ataques de Negación de servicio (*DoS)* que se basan en activar todas las opciones de los datagramas *IP* para ver si el sistema operativo no sabe como reaccionar ante ellos y queda inutilizado.

Correlaciones de contenidos: Estas correlaciones hacen referencia al contenido de datos de los distintos paquetes de información que circulan por la red. Se debe buscar patrones como en conexiones *TELNET o FTP (File Transfer Protocol)*, inspeccionar conexiones *HTTP* (*Hypertext Transfer Protocol*) en busca de comandos que atenten contra la seguridad como por ejemplo: exec c:\winnt\command\format.

1.4.6 FALSOS POSITIVOS Y FALSOS NEGATIVOS

Un punto básico a tratar tras el análisis de las muestras obtenidas en la red es la detección de falsos positivos y falsos negativos.

1.4.6.1 Falso positivo (*false positive*)

Es un término aplicado a un fallo de detección en un sistema de alertas. Sucede cuando se detecta la presencia de una intrusión en el sistema que realmente no existe.

1.4.6.2 Falso negativo (*false negative*)

Es un término que hace referencia a un fallo en el sistema de alerta. Sucede cuando una intrusión existe en el sistema y es ignorada o no detectada por el sistema de alerta.

Los falsos positivos pueden agruparse en cinco grupos dependiendo de la naturaleza de su origen:

Alarmas de reacción de tráfico (*reactionary traffic alarms*)

Se detecta un comportamiento sospechoso como consecuencia de tráfico generado anteriormente (generalmente no malicioso). Por ejemplo la detección de muchas respuestas *ICMP* (*network unrecheable*) procedentes de un *router* porque el equipo destino no se encuentra operativo o accesible en esos momentos.

Alarmas relacionadas al equipo de red (*equipment related alarms*)

Las alarmas del *NIDS* detectan paquetes dentro del tráfico de la red que identifica como no normales. Esto puede ocurrir por ejemplo con balanceadores de carga, puesto que generan paquetes específicos para el control de todos los nodos.

Violaciones de protocolo (*protocol violations*)

Estos avisos se producen por software mal programado o que implementan de forma incorrecta o anticuada algunas partes de los protocolos de Internet.

Verdaderos falsos positivos (*true false positives*)

Todos aquellos falsos positivos que no se encuadren en ninguna de las categorías anteriores.

Alarmas que no son maliciosas (*non malicious alarms*)

Alarmas producidas al detectar rastros de comportamientos maliciosos pero que en ese contexto determinado no lo son. Si se publica en la página *Web* el código de un virus analizándolo, cada vez que una persona descargue la página creará una alerta en el *IDS* porque detectará el virus en la red.

Cabe recalcar aquí que el sistema de detección de intrusos debe producir los mínimos falsos positivos posibles y ningún falso negativo.

1.5 ESPECIFICACIÓN DEL SOFTWARE

1.5.1 OBJETIVOS

El software a desarrollarse debe cumplir con la necesidad de que sea una herramienta fácil de utilizar para cualquier usuario que tenga conocimientos mínimos de redes de datos, considerando las opciones y operaciones más importantes para evaluar de una manera óptima la red de datos de una organización.

Es por eso que del análisis de los problemas anteriormente presentados se van a programar y configurar dos módulos que cubrirán de una manera efectiva los mecanismos más comunes de prueba de vulnerabilidades; éstos son el escáner de puertos y el sistema de detección de intrusos *Snort* mediante una interfaz gráfica para la configuración de los principales parámetros del *IDS* considerando que esta herramienta es una de las más utilizadas y con menos errores de detección que se tienen hoy en día y además es gratuita (*Freeware*).

1.5.2 RESULTADOS ESPERADOS

Los resultados o salida de datos del programa dependerán del módulo operante. En el caso del módulo del *IDS* la ejecución final será la ejecución del programa en el modo *DOS* y los resultados se exportarán a archivos (*logs*) para su análisis posterior; estos *logs* generados serán de acuerdo a las opciones escogidas desde el menú de usuario de inicialización del programa.

En el módulo de escáner de puertos se presentará una lista que contendrá los siguientes datos: el número de puerto, estado del puerto, número de veces que se

hizo la prueba y nombre del protocolo; los resultados de este módulo podrán ser almacenados en archivos de excel (*.xls) para su análisis posterior.

1.5.3 DATOS DE ENTRADA PARA LA OPERACIÓN DE LOS MÓDULOS

Al igual que el punto anterior, los datos necesarios para las pruebas dependen del módulo a ejecutarse; estos datos deberán ser ingresados por el usuario desde las interfaces gráficas para el ingreso de los mismos.

Datos de entrada módulo de interfaz *IDS*

En el módulo de Interfaz del *IDS* se escogerán las opciones de inicialización: El interfaz a analizar, la creación o utilización de reglas de filtraje y la exportación de reportes de datos para su análisis posterior.

Datos de entrada módulo de escáner de puertos

En el módulo de escaneo de puertos los datos de entrada dependerán del tipo de prueba, sea ésta remota o local. La prueba local requerirá el número de puerto o rango de puertos y número de intentos; la prueba remota además de todos los parámetros enunciados anteriormente deberá incluir la dirección *IP* del elemento de red remoto a analizar.

1.5.4 PROCESAMIENTO DE LOS DATOS DE ENTRADA

Procesamiento en el módulo de interfaz de configuración de *Snort* de *DOS*

- Se presentará una interfaz netamente programada en Visual C++ para que el usuario ingrese los datos requeridos de manera fácil y entendible.

- Mediante el uso de *Check Box* y *Radio button,* el usuario podrá seleccionar las tareas que se quieren realizar.

- Según las opciones seleccionadas por el usuario se tendrán que solicitar datos adicionales necesarios.

- Teniendo todos los datos necesarios, se crearán archivos *BATCH* (procesamiento por lotes) para que sean ejecutados desde el sistema *DOS*, al ejecutar estos archivos se perderá el control mediante el interfaz y se manejará mediante *DOS*.

Procesamiento en el módulo de escaneo de puertos

- Se presentará una pantalla con los cuadros de texto necesarios para el ingreso de los datos necesarios para realizar la operación según sea la prueba local o remota.

- Los datos ingresados serán procesados como cadenas de texto; con los datos ingresados se creará un *SOCKET,* mediante este *SOCKET* se intentará una conexión en el puerto a ser auditado, analizando si se puede o no establecer dicha conexión, con esto se podrá conocer si está el puerto abierto o cerrado; esto se lo hará empleando banderas las que retornaran un verdadero cuando el puerto esté abierto y un falso cuando esté cerrado.

- Para escritura de archivos de texto de los reportes de resultados, se crearán nodos con todos los datos necesarios como son: la dirección del elemento de red, el puerto y el estado; todos estos datos serán manejados como cadenas de texto para la escritura de los archivos de una manera fácil y óptima.

- Los resultados podrán ser almacenados en archivos de excel (*.xls) con las funciones dadas por el MFC de Visual C++ incluyendo las librerías necesarias para el trabajo correcto.

1.6 ANÁLISIS COMPARATIVO CON SOLUCIONES COMERCIALES

En el mercado y especialmente en el Internet, existen diversas opciones de software y paquetes para realizar algunas de las pruebas mencionadas anteriormente en las redes de datos.

Hasta el momento de escribir este capítulo no se encontró ningún paquete que tenga los dos módulos que presenta el software a desarrollarse, es por eso que si se quiere cubrir estas necesidades o pruebas para la auditoría mediante opciones comercialmente bajadas de Internet se tendrían que instalar por separado paquetes correspondientes a la prueba de puertos (abiertos o cerrados) en elementos de red y el sistema de detección de intrusos.

Otro problema que se tiene, aunque no relacionado específicamente al software de auditoría de red sino al software bajado mediante el Internet, es el hecho de que son paquetes de evaluación y su duración está limitada o presenta funciones restringidas de uso (no con todas las funciones del paquete completo). Adicionalmente, se presenta el riesgo de que los programas sean alterados introduciendo virus, troyanos que en vez de ayudar a la seguridad van a empeorarla; claro está que esto se puede controlar mediante la utilización de software antivirus, bajar de páginas seguras o controlando mediante funciones *HASH* dadas por el fabricante para comprobar que no haya sido alterado el software.

Programas comerciales para el módulo escáner de puertos

Uno de los principales problemas de los sistemas de escáner de puertos encontrados en el Internet, es que tienen interfaces complejas para su utilización en relación a funcionalidad presentada; comparado con el software a desarrollarse que tendrá una interfaz de fácil utilización y ayudas complementarias para su manejo.

Existen varias soluciones específicas comerciales en el mercado, se van a enunciar las principales que se pudieron encontrar.

Nombre: Advanced Port Scanner V1.2
Licencia: Freeware
Sistema Operativo: Microsoft Windows

Operación:

Tiene posibilidades de mostrar el estado de los puertos de una máquina especificando la dirección *IP*; al igual que el software a desarrollar permite incluir el rango de puertos a ser auditados de una dirección *IP* específica.

Este programa tiene la funcionalidad de hacer pruebas en un rango específico de direcciones *IP*; una diferencia con el programa a desarrollarse es que no se cuenta con la opción de poner un número de intentos en caso de que el resultado de una prueba de puerto de como resultado “cerrado”, opción mediante la cual se puede tener un resultado más confiable.

Nombre: AD Security Port Scanner 4.61
Licencia: Shareware (Limitado a 15 días de uso o 15 ejecuciones)
Sistema Operativo: Microsoft Windows.

Operación:

Tiene como parámetros de ingreso las direcciones *IP* de las máquinas a realizar la prueba, número de puerto o rango de puertos; se puede notar que la interfaz es muy complicada en relación a su funcionalidad. Se diferencia del módulo a implementarse en que no tiene opción de intentos cuando el resultado del puerto es “cerrado”; una falencia adicional es que el programa es lento en evaluar el rango de puertos.

Nombre: Angry *IP* Scanner 2.21
Licencia: Freeware
Sistema Operativo: Microsoft Windows

Operación:

Éste es un programa que permite auditar los puertos abiertos o cerrados de máquinas de red; presenta además otra funcionalidad que permite observar las máquinas que están conectadas a la red. En relación a las pruebas se puede escanear los puertos por nombre de aplicación y número; una variación importante es que permite ingresar una lista de los puertos a hacer las pruebas.

Al igual que todos los software mencionados anteriormente no tiene la funcionalidad de poner un número de intentos de escaneo en caso de tener un resultado negativo (puerto cerrado).

Programas comerciales para módulo sistema de detección de intrusos

Los sistemas de detección de intrusos también presentan interfaces muy complejas para su configuración y utilización.

Las pocas y nuevas herramientas no son confiables y probadas como es en el caso de *Snort* que ha sido probado en distintos tipos de redes dando resultados garantizados y confiables; además como es una herramienta que se ejecuta en modo *DOS* hace que consuma pocos recursos de la computadora.

Nombre: Nuzzler Intrusion Detection System
Licencia: Shareware (Limitado a 30 días de Uso)
Sistema Operativo: Microsoft Windows.

Operación:
Interfaz compleja, para realización de análisis de tráfico; la inicialización es lenta ya que se tienen que configurar una serie de parámetros que no se pueden

encontrar fácilmente, como reglas, tarjeta de red, tiempo de funcionamiento. Al igual que el módulo del software que se va a implementar, depende de *WinPcap* el cual debe estar instalado para que funcione correctamente. Una desventaja es que no se pueden crear nuevas reglas para el filtrado, solo se tiene opción de cargar reglas, lo que no se da en el software a desarrollarse donde se podrán crear nuevas reglas de filtrado e igualmente incorporar algunas que estén almacenadas.

Nombre: XRAY Intrusion Detection System - DEMO
Licencia: Shareware (No tiene todas las funcionalidades)
Sistema Operativo: Microsoft Windows

Operación:

De todos los programas mencionados en este documento, éste tiene la interfaz más amigable y de fácil configuración, ya que los principales datos para la inicialización son simples de introducir e inicializar la escucha de la red. Como es una versión demo, está limitado la opción de hacer *logs* de todas las posibles intrusiones que se tengan, se presume que el formato de tales reportes deben constar de la misma información que se dispondrá el módulo del programa relacionado.

Al inicializar este software, se pudo notar que brinda estadísticas de todos los paquetes que están circulando por la red, dando una idea de la cantidad de paquetes de cada protocolo que están siendo utilizados.

DISEÑO, DIAGRAMACIÓN Y CODIFICACIÓN DEL PROGRAMA

En este capítulo se indicarán los procedimientos necesarios para elaborar un sistema informático basado en el principio de la Ingeniería de Software, específicamente mediante *UML (Unified Modeling Language).*

Para comenzar se definirán los actores, se tendrán que elaborar todos los diagramas necesarios, con esto programar el sistema, para que por último ejecutando pruebas y observando la compilación del software determinar los requerimientos mínimos de hardware y software necesarios para la correcta ejecución del mismo.

2.1 DIAGRAMACIÓN *UML* DE LOS MÓDULOS DEL SOFTWARE

Para la elaboración del software se sigue el proceso *UML,* siendo ésta una metodología que tiene varias actividades para transformar los requerimientos en un sistema informático.

El que un sistema informático sea considerado bueno involucra que cumpla con las necesidades del usuario: [10]

Útil y aprovechable: Un buen software hace la vida de los usuarios más fácil o mejor.

Fiable: Un buen software tiene pocos errores.

Flexible: Como las necesidades del usuario cambian durante el tiempo, es importante poder realizar los cambios requeridos sin alterar resultados y sin producir errores.

Objetivos de *UML* [11]

UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.

UML no pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso. *UML* incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno interactivo, basado en construir una sólida arquitectura para resolver requisitos.

UML debe ser tan simple como sea posible, pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir. *UML* necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.

Debe ser un lenguaje universal, como cualquier lenguaje de propósito general, e imponer un estándar mundial.

Este proceso unificado tiene varias características entre las cuales se pueden resaltar los casos de uso, arquitectura a utilizarse y un ciclo de vida interactivo.

En el desarrollo de este sistema se va a presentar los diagramas de casos de uso, diagramas de colaboración, diagrama de clases, diagramas de interacción, diagramas de componentes y de navegación.

2.1.1 FASE DE INICIO

2.1.1.1 Objetivo

El objetivo del software a desarrollar es el monitoreo y obtención de reportes de una red de datos que esté trabajando bajo la arquitectura *TCP/IP*. El contenido de

estos reportes de datos dependerá del módulo escogido, sea éste, escáner de puertos o la interfaz gráfica para la configuración del *IDS Snort*.

2.1.1.2 Alcance

El software a desarrollar permite obtener datos de las redes trabajando bajo la arquitectura *TCP/IP*, es por eso que podrá analizar los equipos que pertenezcan a una red de datos sean éstos que estén interconectados de manera cableada o inalámbrica. Si es que se quiere analizar todo el tráfico circundante en la red se necesitará un elemento de interconexión, como un *switch* con capacidad de configurar a un puerto como Puerto espejo[2] *(Port mirrorring)* que permite analizar todo el tráfico que fluye por la red.

2.1.1.3 Descripción

Como se ha mencionado en la actualidad toda organización por más pequeña y reciente que sea, es dependiente de sus recursos informáticos así como también del acceso a Internet; cuando se presenta una falla de alguno de estos recursos informáticos y considerando que el ancho de banda de acceso a Internet es un recurso limitado, donde su desperdicio implica retrasos de trabajo conllevando a pérdidas de dinero. Se debe dar una orientación a los administradores y gerentes de las organizaciones que se tiene que invertir en esta área; se debe observar que al inicio puede ser un gasto alto pero que evaluando los beneficios a futuro es una buena inversión.

Con estos pequeños antecedentes se presenta esta herramienta computacional que permitirá analizar la información circundante de la red de datos de la organización en busca de tráfico malicioso, que pueda perjudicar la operatibilidad de la organización; además se tendrá la opción de generar y obtener reportes de datos según lo configurado por el usuario y el módulo utilizado.

El primer módulo, considerado el más importante es el que generará alertas de las posibles intrusiones que se estén dando en la red de datos. Como se mencionó anteriormente, esto se lo hará analizando el tráfico que fluye en un

[2] *Puerto Espejo:* Configuración de un puerto de un *switch* para poder captar todo el tráfico de una red de datos; se configura de tal manera que se envía una copia de los paquetes entrantes y salientes al puerto que es de administración o de análisis de tráfico. [4]

segmento de red, discriminándolo según parámetros y reglas que van a ser configurados logrando tener un historial del tráfico circulante; se indicará al administrador de la red si éste es nocivo.

El segundo módulo del software permitirá conocer de una manera rápida si se tiene problemas de puertas abiertas para atacantes; estas puertas abiertas se refieren a la que dejan los puertos que se encuentren abiertos y que puedan ser utilizados por atacantes para acceder al equipo intentando una conexión para hacer daño.

De cualquiera de los dos módulos se podrá generar y obtener reporte de los resultados para un análisis posterior.

2.1.1.4 Requerimientos específicos

Funcionalidad

Los requerimientos funcionales del sistema son:

- La interfaz del usuario para el manejo del programa debe ser lo más amigable y con las funcionalidades totales de uso.

- Los datos generados en cada uno de los módulos del software deben ser exportados a archivos de texto o hacia archivos que podrán ser abiertos con programas específicos.

- El software debe funcionar en cualquier plataforma de Microsoft Windows 98 o superior.

- Sólo un módulo del programa podrá funcionar a la vez, este módulo de funcionamiento será escogido desde el menú principal del sistema.

Confiabilidad

- El sistema debe ser confiable, obteniendo resultados que tengan consistencia con la realidad; se debe eliminar en lo posible los falsos negativos y reducir los falsos positivos.

- La disponibilidad del sistema debe ser primordial, ya que éste puede estar ejecutándose el tiempo que se requiera, en relación al número de paquetes de datos a analizar o puertos a probar, según sea el módulo.

Desempeño

- El tiempo de respuesta en cualquiera de los módulos dependerá de la prueba a realizar; por ejemplo, en el módulo escáner de puertos el procesamiento de la información dependerá del número de puertos que se analice en cada solicitud y del número de intentos.

Consultas e informes

Según el módulo de funcionamiento se requerirán los siguientes aspectos:

- **Requerimiento 1:** Informe de Puertos abiertos y cerrados de equipos de red (Módulo de escáner de puertos).

- **Requerimiento 2:** Informe de Tráfico obtenido de un número de paquetes circulantes, con relación a una regla preestablecida (Módulo de Sistema de detección de intrusos).

Almacenamiento

Según el módulo de funcionamiento se requerirán los siguientes aspectos:

- **Requerimiento 3:** Dirección *IP* de máquina auditada, número de puerto, número de intentos, estado de puerto (Módulo de escáner de puertos).

- **Requerimiento 4:** Dirección *IP* Origen, Dirección *IP* destino, Puertos, Protocolos, Información de datos de capa a ser analizada (Módulo de Sistema de detección de intrusos).

2.1.2 MODELO DE CASOS DE USO

Definición de los Actores

Usuario

Es el actor que ejecutará el programa, no tiene ninguna restricción en relación a uso y acceso a todo el sistema. Este usuario tiene la posibilidad de realizar tres operaciones fundamentales: REALIZAR PRUEBAS, REPORTAR RESULTADOS, GUARDAR ARCHIVOS, tal como muestra el diagrama de casos de la figura 2.1

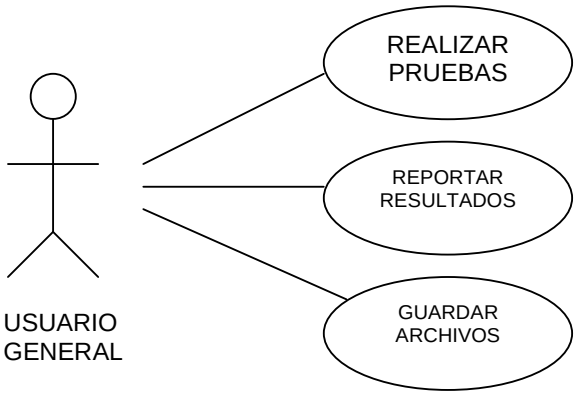

Fig. 2.1 Diagrama de casos de uso para el Sistema de Auditoría

Hardware o Computador

Es el actor secundario el cual se encargará de ser la interfaz entre el usuario y la red de datos, éste procesará las solicitudes del usuario para poderlas enviar a través de la red de datos o procesarlos localmente, para con esto realizar las operaciones y obtener los resultados.

2.1.2.1 Descripción de casos de uso

Módulo Interfaz *IDS*

Nombre: Interfaz *IDS*	
Actores: Usuario, Hardware o Computador	
Función: Configurar los parámetros para el sistema de detección de intrusos *Snort* mediante una interfaz gráfica.	
Descripción: Ingresar parámetros y datos necesarios para la configuración y ejecución del sistema de detección de intrusos *Snort* de *DOS*; esto es: inicialización de *Snort* en cualquiera de los modos de funcionamiento, opciones para mostrar información de capa aplicación, de capa enlace, creación de reglas de filtrado y alertas, carga de reglas de filtrado desde archivos, generación de reportes (*logs*).	
Referencias: Ninguna	
De requerimientos: R2, R4	
Actividades de Caso: Modo *Sniffer*	
Evento	**Respuesta del Sistema**
1. Selección de opción modo de funcionamiento *Sniffer.*	2. Capturar la selección.
	3. Mostrar pantalla de captura de datos para funcionamiento en Modo *Sniffer.*
4. Ingresar y seleccionar los datos requeridos como número de paquetes a analizar y nombre de archivo *Log* a ser generado.	5. Capturar los datos requeridos.

	6. Mostrar pantalla que solicita el nombre de archivo de ejecución por lotes para *Snort* a ser generado.
7. Ingresar el nombre de archivo a ser generado (secuencia por lotes de *DOS*).	8. Generar el archivo *BATCH* para procesamiento.
	9. Ejecutar archivo *BATCH* generado.
Eventos Alternativos de Caso: Modo *Sniffer*	
Número de Línea	**Respuesta del Sistema**
Línea 2:	No se digitó ninguna selección y muestra mensaje informativo.
Línea 5:	Si algún dato está en blanco, informarle y no permitirle seguir hasta que se lo digite o seleccione.
Línea 7:	Advertir si el nombre de archivo nuevo es existente, habrá posibilidad de reemplazarlo o mantenerlo.
Línea 8:	Si la generación del archivo falla, mostrar un mensaje y regresar a operación anterior.
Línea 9:	Si la ejecución del archivo falla, mostrar un mensaje y regresar a operación anterior.
Actividades de Caso: Modo *IDS*	

Evento	Respuesta del Sistema
1. Selección de opción modo de funcionamiento *IDS.*	2. Capturar la selección.
	3. Mostrar pantalla de captura de datos para funcionamiento en Modo *IDS.*
4. Ingresar y seleccionar los datos requeridos como tipo de información a analizar, número de paquetes, nombre de archivo *Log* a ser generado y reglas de filtrado.	5. Capturar los Datos.
	6a. Mostrar interfaz para generar nuevas reglas de filtrado o alertas.
	6b. Mostrar archivos preescritos de reglas para escoger alguno para su utilización y ejecución.
7. Ingresar y seleccionar los datos o archivos requeridos para la generación de reglas.	8a. Generar el archivo *.rule* para llamarlo desde archivo de ejecución por lotes para *Snort.* 8b. Grabar selección en una variable para su uso en el archivo de ejecución por lotes de *Snort.*
9. Aceptar todos los parámetros ingresados	10. Generar el archivo *BATCH* para procesamiento.
	11. Ejecutar archivo generado.

Eventos Alternativos de Caso: Modo *IDS*	
Número de Línea	**Respuesta del Sistema**
Línea 2:	No se digitó ninguna selección y se muestra un mensaje informativo.
Línea 5:	Si algún dato está en blanco informarle y no permitirle seguir hasta que se lo digite o seleccione, además los datos ingresados deben tener coherencia, se los revisará mediante los respectivos filtros; se indicará al usuario de cualquier error.
Línea 10:	Si la generación del archivo falla, mostrar mensaje y regresar a operación anterior.
Linea 11:	Si la ejecución del archivo falla, mostrar mensaje y regresar a operación anterior.

Módulo Escáner de Puertos

Nombre: Escáner de Puertos
Actores: Usuario, Hardware o Computador
Función: Comprobar qué puertos están abiertos o cerrados en un elemento de red

Descripción:

Módulo en el cual las pruebas se las podrá realizar remota o localmente. La prueba local requerirá el número de puerto o rango de puertos y número de intentos; la prueba remota además de todos los parámetros enunciados anteriormente deberá incluir la dirección *IP* del elemento de red remoto a escanear. Los resultados de este módulo se presentarán en una lista indicando el número de puerto, seguido del estado del mismo. Este módulo tendrá la opción de almacenar los reportes como archivos de excel (*.xls) en una ubicación que el usuario lo crea conveniente.

Referencias: Ninguna

De requerimientos: R1, R3

Actividades de caso: Escáner Local

Evento	Respuesta del Sistema
1. Selección de opción prueba local.	2. Capturar la selección.
	3. Habilitar los controles de pantalla para la captura de datos de funcionamiento de escáner local.
4. Ingresar los datos requeridos para el funcionamiento.	5. Capturar los datos en las variables correspondientes.
6. Seleccionar finalizar, para aceptar el ingreso y selección de los parámetros y datos.	7. Guardar los datos en el nodo de almacenamiento correspondiente para su utilización.

8. Seleccionar generar reporte de datos.	9. Solicitar el nombre para generar el archivo correspondiente.
10. Ingresar el nombre del archivo a grabar el reporte de datos.	11. Ejecutar la prueba correspondiente.
Eventos Alternativos de Caso: Escáner Local	
Número de Línea	**Respuesta del Sistema**
Línea 2:	No se tiene ninguna selección ni ingreso de datos; se debe mostrar un mensaje informativo. No se digitó ninguna selección y se muestra un mensaje informativo.
Línea 5:	Si algún dato está en blanco informarle y no permitirle seguir hasta que se lo digite o seleccione.
Línea 7:	Si algún dato no tiene coherencia o está en blanco notificar al usuario del problema.
Línea 10:	Si la generación del archivo, falla mostrar mensaje y regresar a operación anterior .
Actividades Caso de Uso: Escáner Remoto	

Evento	Respuesta del Sistema
1. Selección de opción prueba remota.	2. Capturar la selección.
	3. Habilitar los controles de pantalla para la captura de datos de funcionamiento de escáner remoto.
4. Ingresar los datos requeridos para el funcionamiento.	5. Capturar los datos en las variables correspondientes.
6. Seleccionar finalizar para aceptar el ingreso y selección de los parámetros.	7. Guardar los datos en el nodo de almacenamiento correspondiente para su utilización.
8. Seleccionar generar reporte de datos.	9. Pedir nombre para generar el archivo correspondiente.
10. Ingresar el nombre del archivo a grabar el reporte de datos.	11. Ejecutar la prueba correspondiente.
Eventos Alternativos de Caso: Escáner Remoto.	
Número de Línea	**Respuesta del Sistema**
Línea 2:	No se tiene ninguna selección ni ingreso de datos; se debe mostrar un mensaje informativo.
Línea 4:	Si algún dato está en blanco se informa y no se permite seguir el proceso hasta que se digite o seleccione.

Línea 7:	Si algún dato no tiene coherencia o está en blanco, notificar al usuario del problema.
Línea 9:	Si la generación del archivo falla, mostrar mensaje y regresar a operación anterior.
Linea 11:	Si la ejecución del archivo falla, mostrar mensaje y regresar a operación anterior.

2.1.3 PAQUETE DE ANÁLISIS

Para poder realizar el análisis del estado del puerto de una manera óptima y rápida, se crea un paquete de análisis en donde se va a tener todos los datos que se necesitan en un solo nodo.

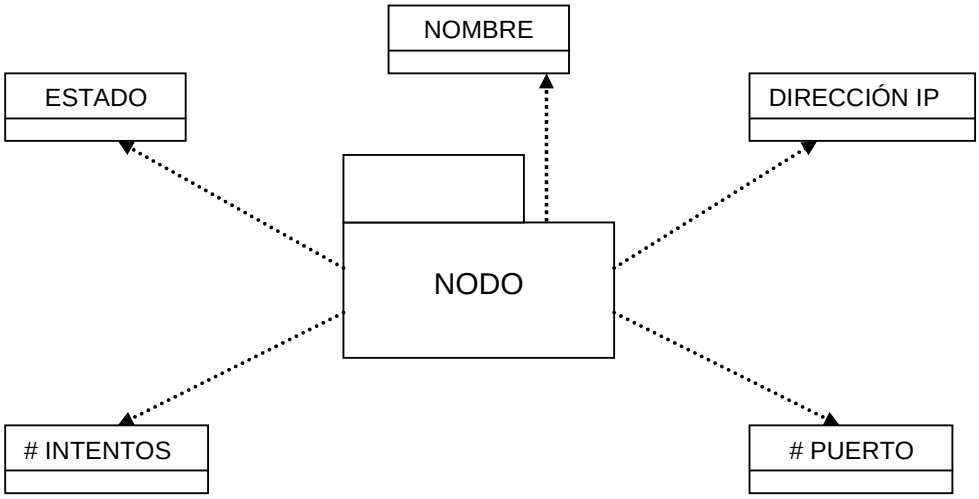

Fig. 2.2 Nodo de datos para Análisis

2.1.4 MODELO DE DISEÑO

Como ya se ha mencionado, el usuario que ocupa el sistema puede realizar dos tipos de pruebas diferentes como son: el escáner de puertos y la del sistema de detección de intrusos.

Si se escoge el escaneo de puertos, el usuario tiene la posibilidad de hacer la prueba local y remota, lo que determinará los datos necesarios que deberán ser ingresados y de su envío o no a través de la red de datos; así mismo en el caso del sistema de detección de intrusos se debe escoger el modo de funcionamiento. Igualmente dependiendo de la prueba se necesitarán diferentes datos y se crearán o no reglas para filtrado de tráfico.

Todo lo anterior mencionado se encuentra explicado claramente en el diagrama de clases de la figura 2.3 Cabe resaltar que se presentan las clases desarrolladas y no las creadas por el sistema.

2.1.5 DIAGRAMA DE CLASES

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento.

Un diagrama de clases está compuesto por los siguientes elementos:

Clase: atributos, métodos y visibilidad.
Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

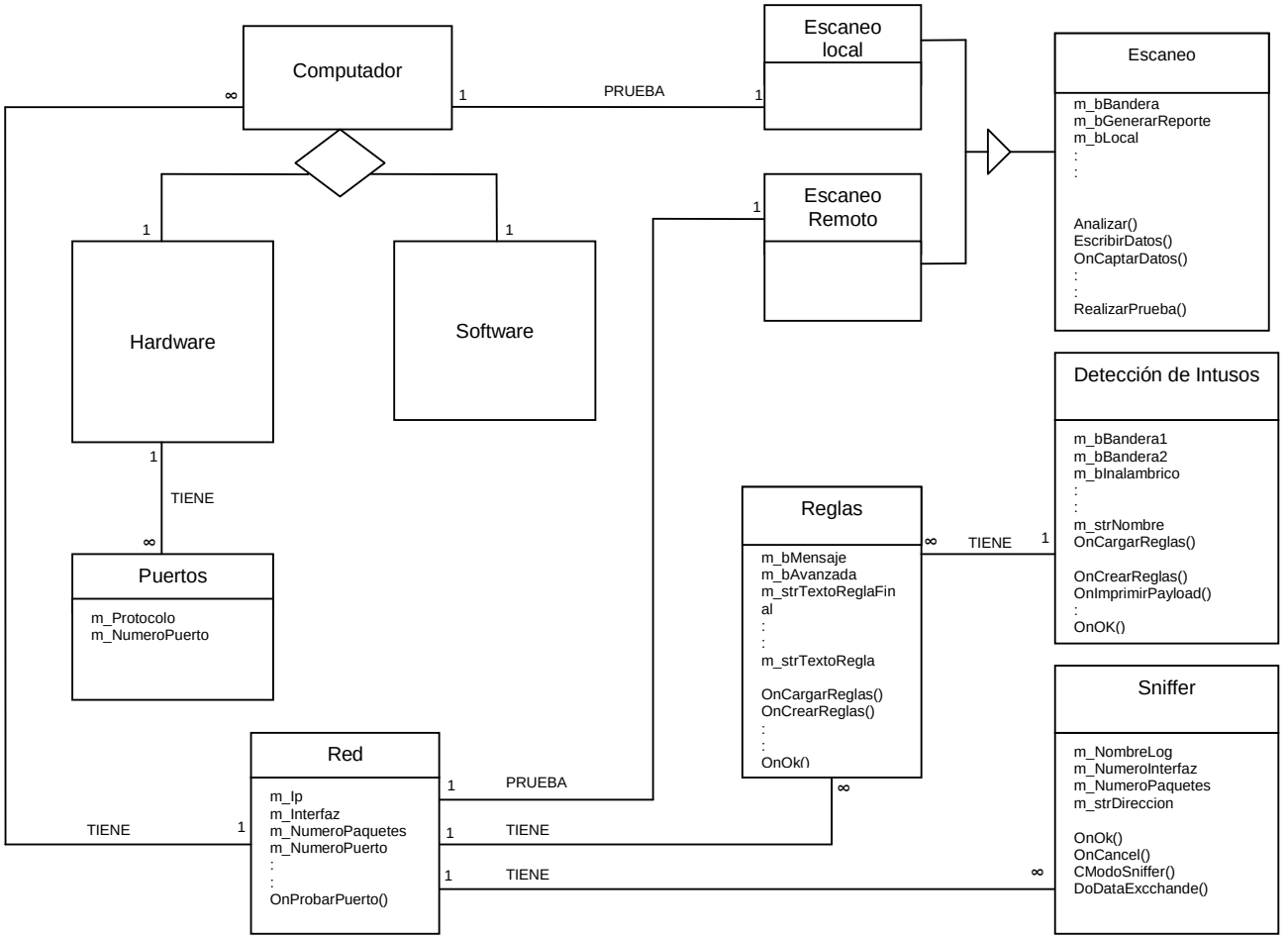

Fig. 2.3 Diagrama de Clases

2.1.6 DIAGRAMA DE INTERACCIÓN

Estos diagramas son aquellos que muestran las interacciones de un usuario con el sistema.

Para entender mejor se va a definir lo que es una Interacción: es una cadena de mensajes enviados entre los objetos en respuesta a un evento generado por el usuario sobre la aplicación.

Los diagramas de interacción pueden ser realizados dentro de los diagramas de secuencia o diagramas de colaboración; en el caso de este trabajo se los realizará en los diagramas de secuencia.

Estos diagramas conforman la etapa del diseño de la aplicación, y se crean a partir de los diagramas de Casos de Uso.

El responsable o ACTOR es quien inicia el ciclo interactuando inicialmente con la interfaz de usuario: GUI; enseguida se inician todos los objetos que intervienen en el funcionamiento del aplicativo.

En este diagrama se comienza a observar el comportamiento del sistema a partir de los eventos generados por los actores. Aquí se interactúa con instancias, no con clases.

Los diagramas de interacción que se necesitarán en el desarrollo de la aplicación son los que se indican en las figuras 2.4 a 2.8.

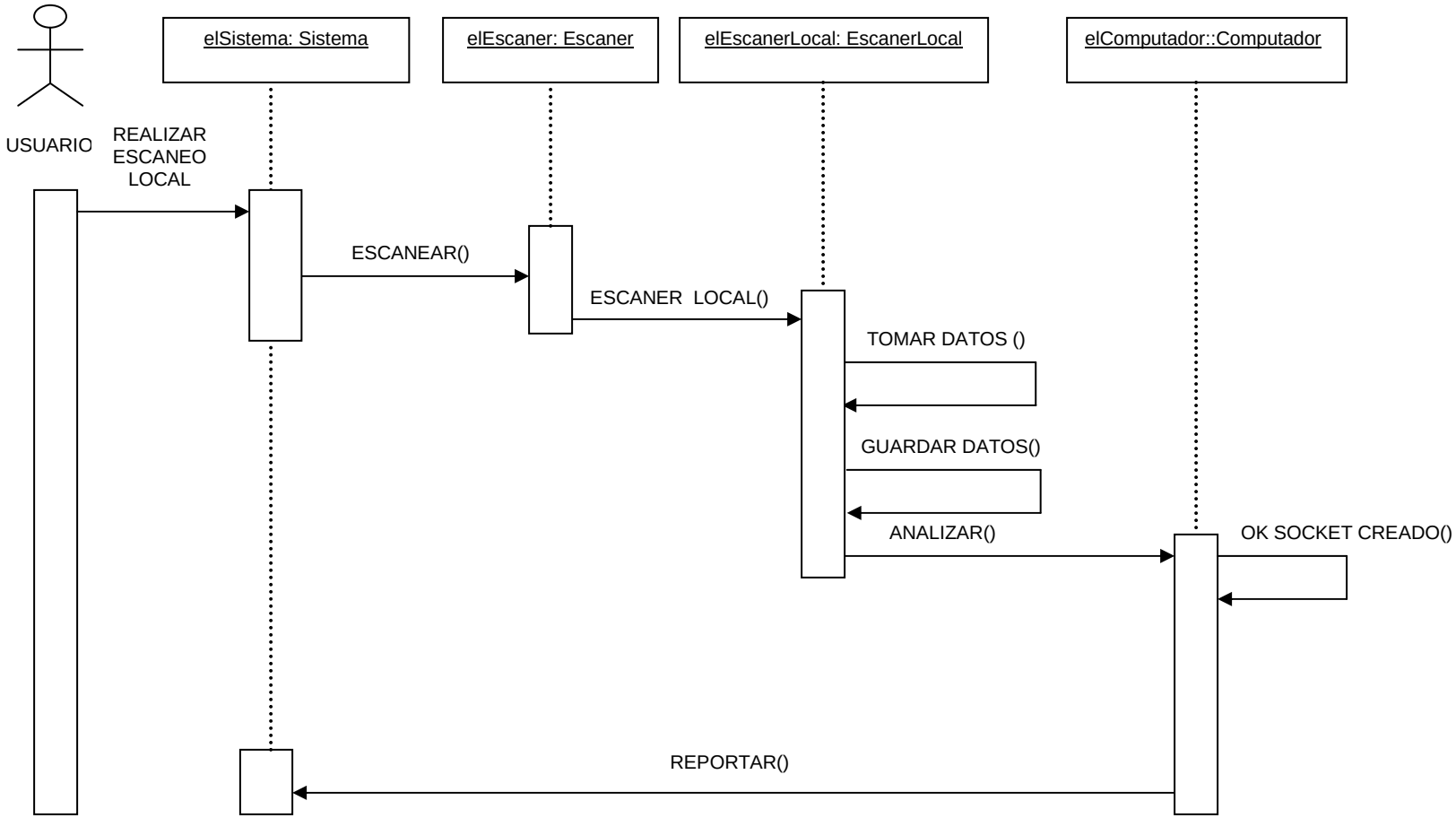

Fig. 2.4 Diagrama de interacción del módulo escáner de puertos local

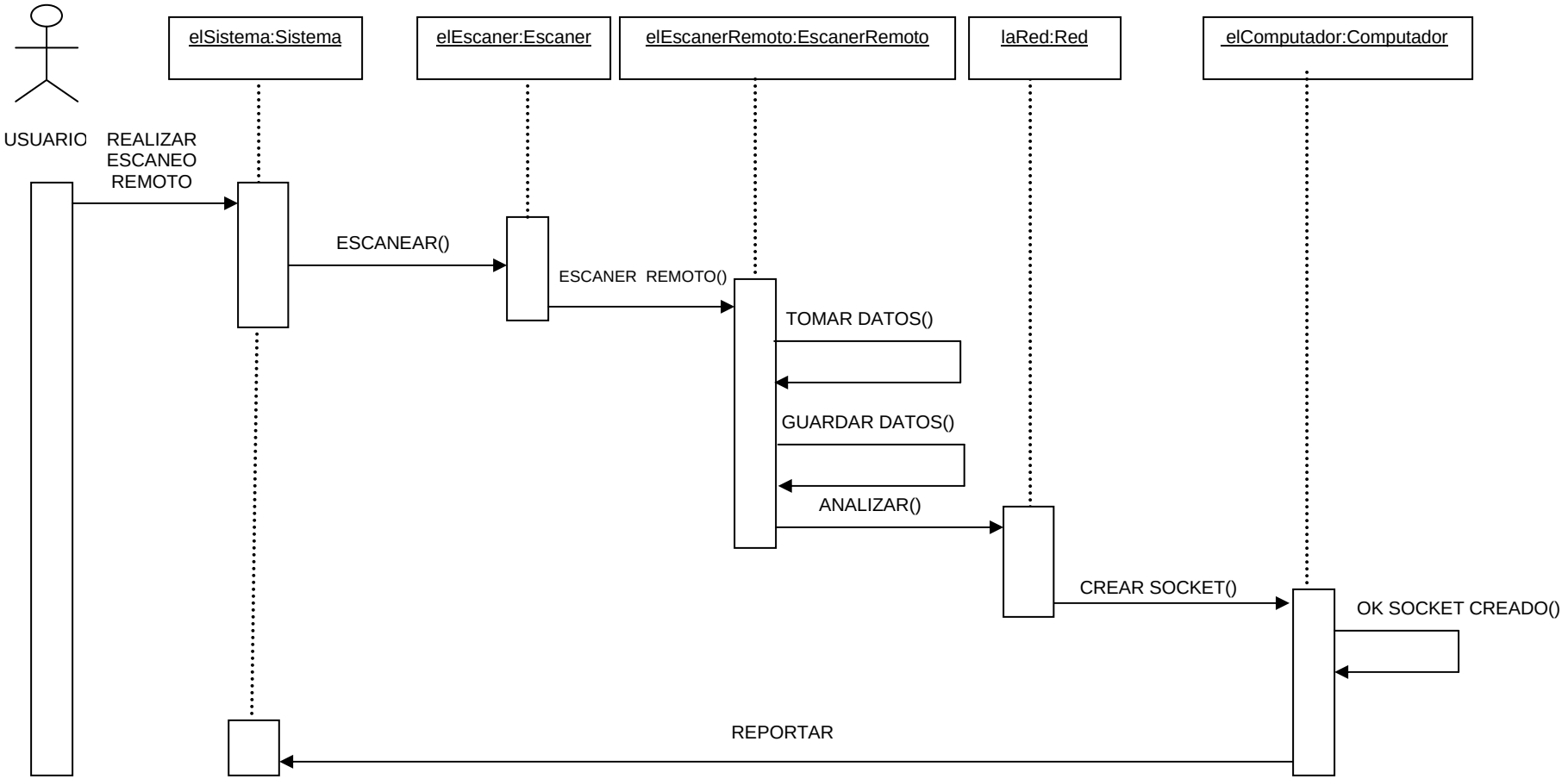

Fig. 2.5 Diagrama de interacción del módulo escáner de puertos remoto

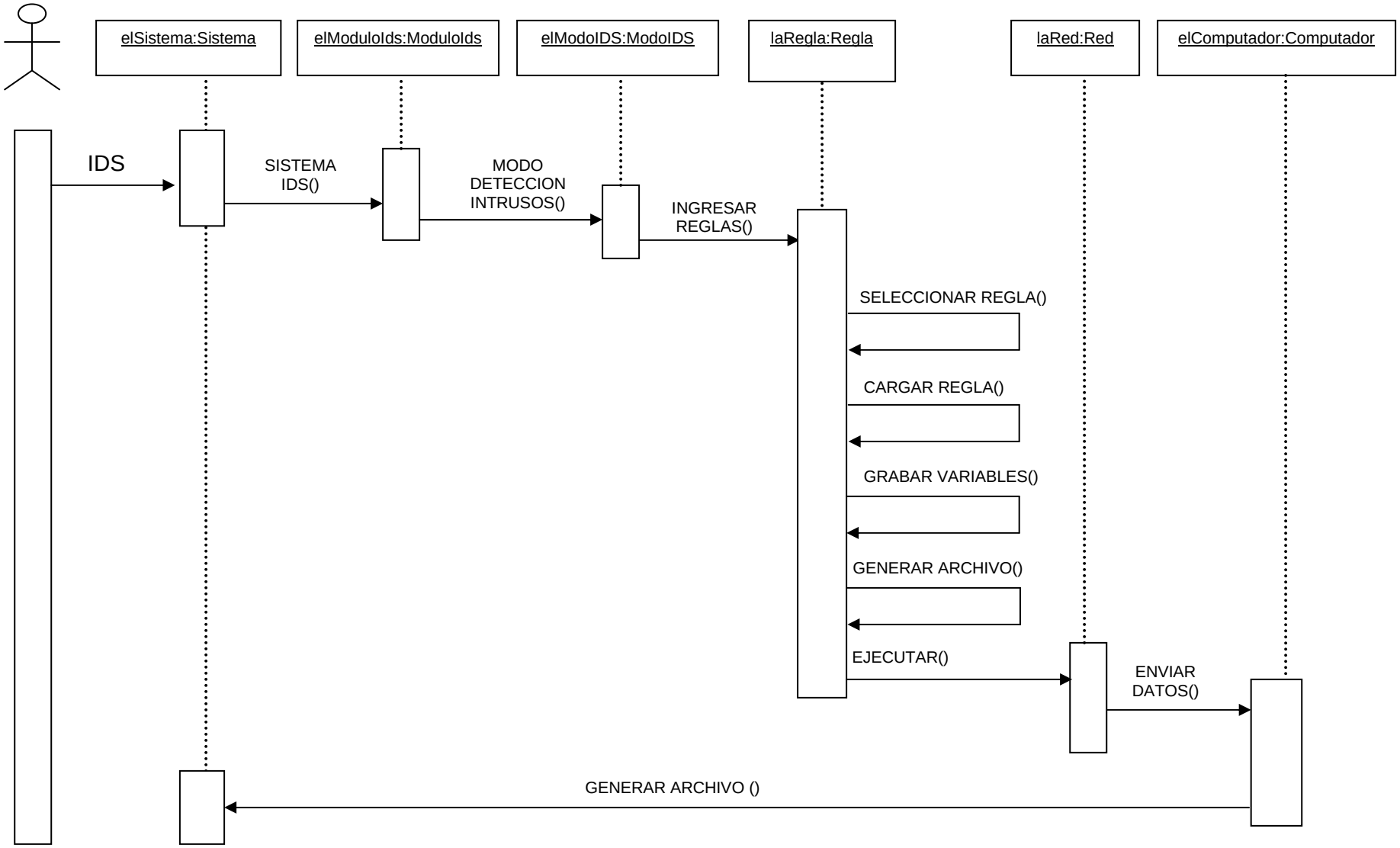

Fig 2.6. Diagrama de interacción del módulo interfaz IDS – modo IDS – Regla Almacenada

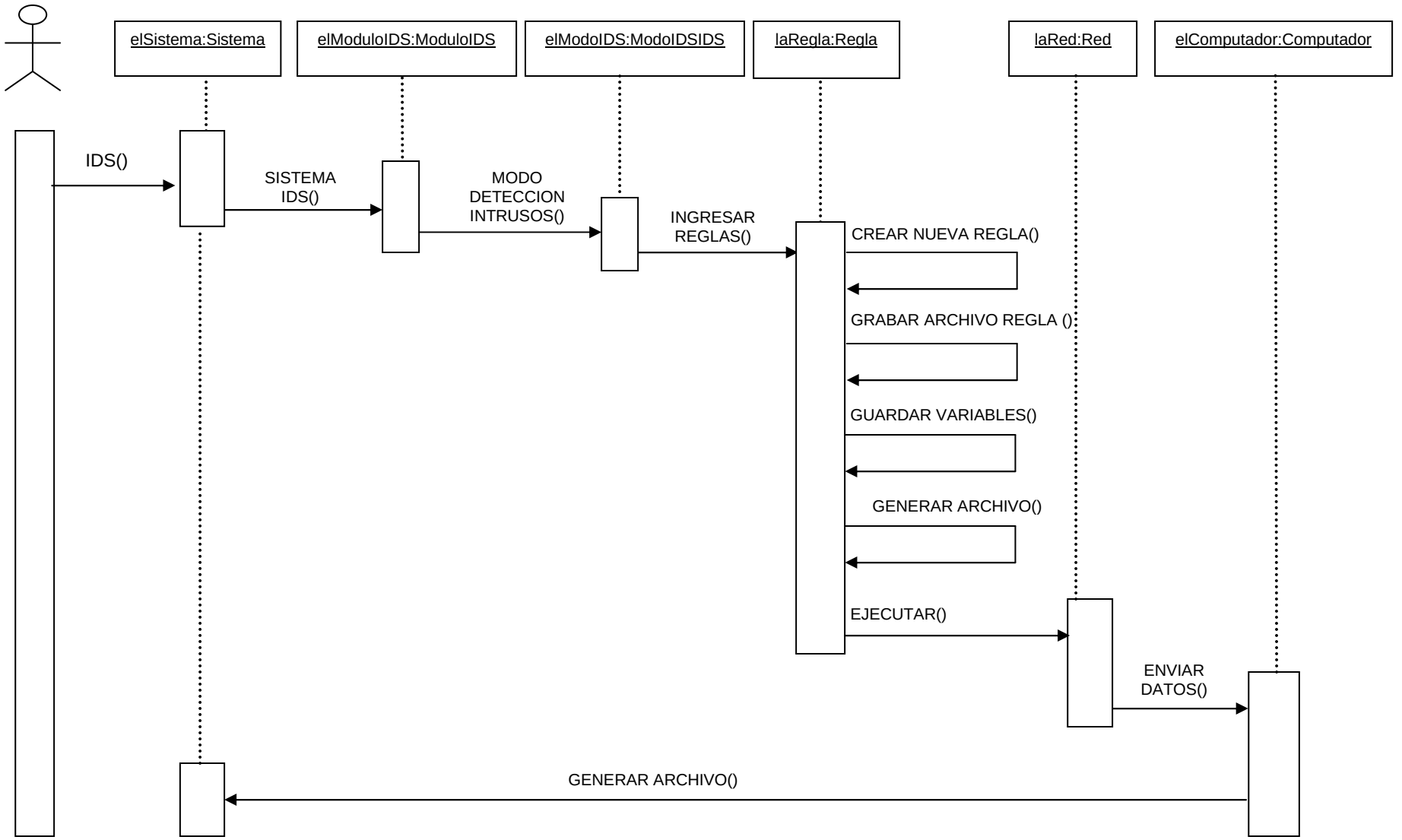

Fig 2.7. Diagrama de interacción del módulo interfaz IDS– modo IDS – Crear nueva regla

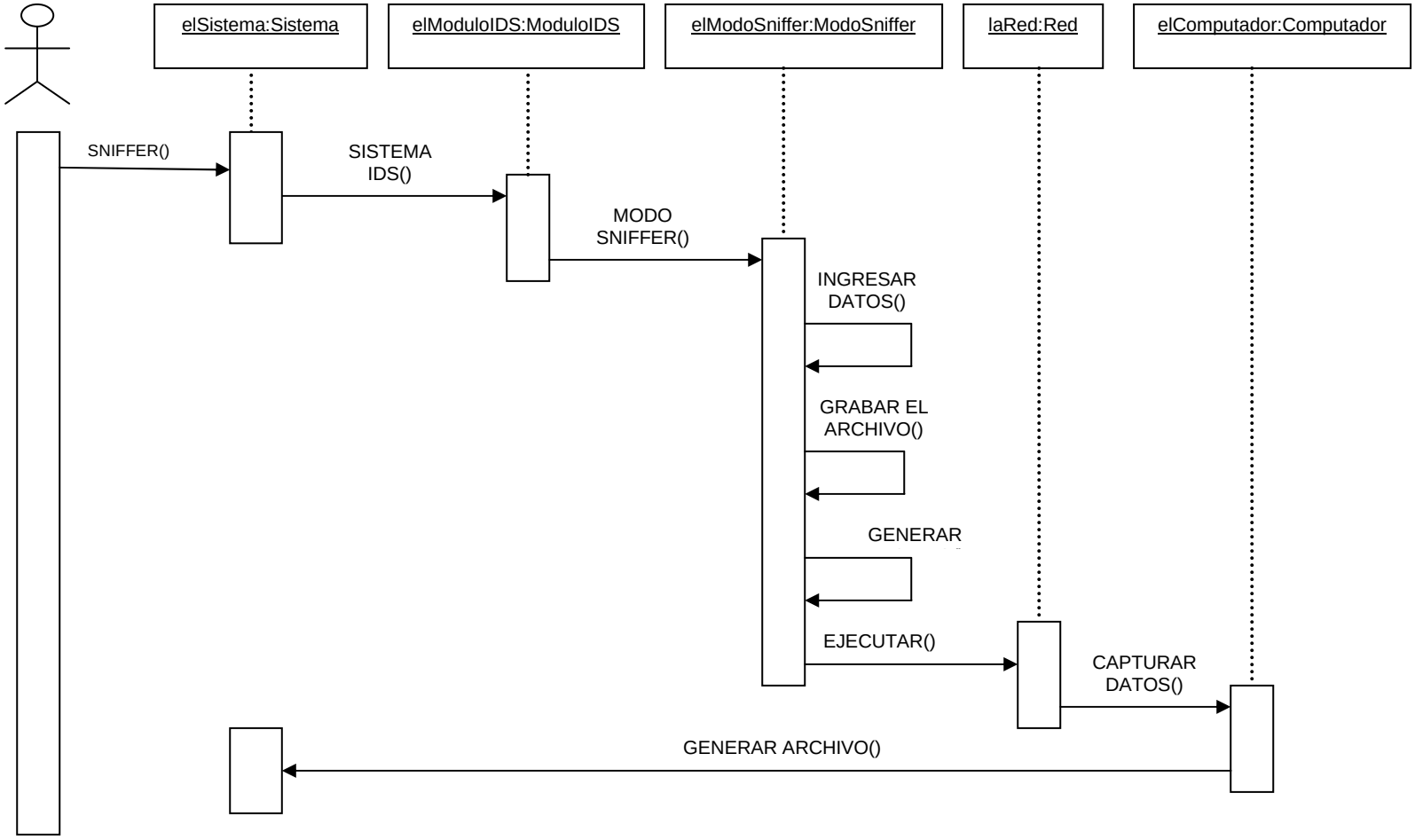

Fig 2.8. Diagrama de interacción del módulo interfaz IDS – modo *sniffer*

2.1.7 DIAGRAMAS DE COMPONENTES

Con este tipo de diagramas se muestran las dependencias que tienen los componentes del sistema en tiempo de compilación de enlace en un programa C++ y en tiempo de ejecución entre componentes; esto quiere decir que para que el sistema funcione correctamente deben estar presentes todos los componentes sean de compilación o ejecución sin errores y listos para su uso.

Cabe recalcar que el diagrama de la figura 2.10 muestra los componentes dependientes que no son librerías estándar de la aplicación y que se tienen que incluir manualmente.

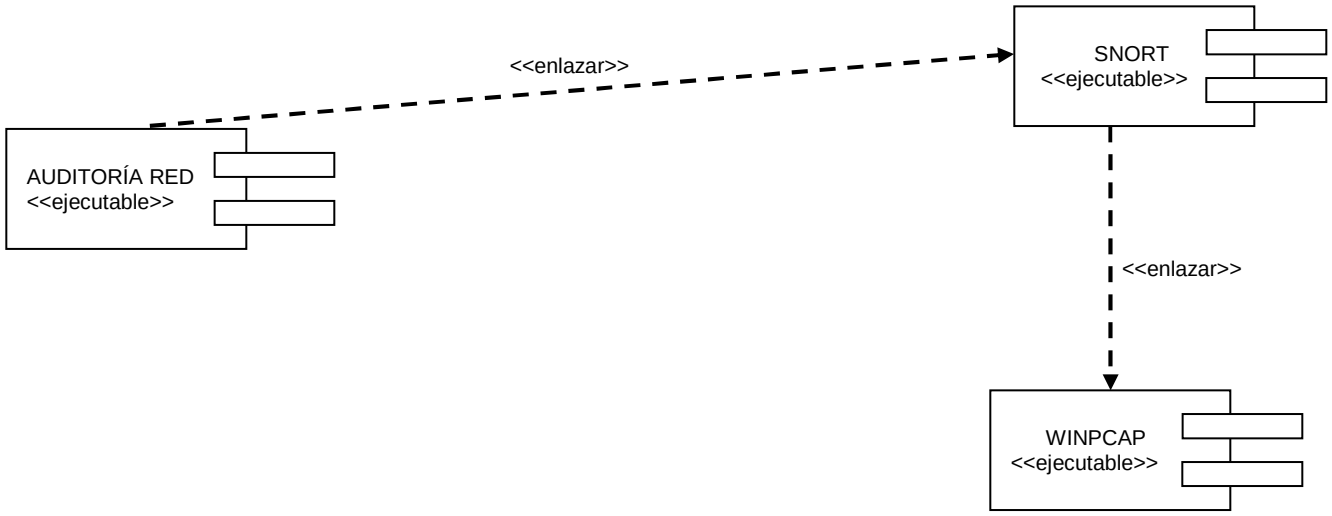

Fig. 2.9 Diagrama de componentes que muestra las dependencias en tiempo de ejecución

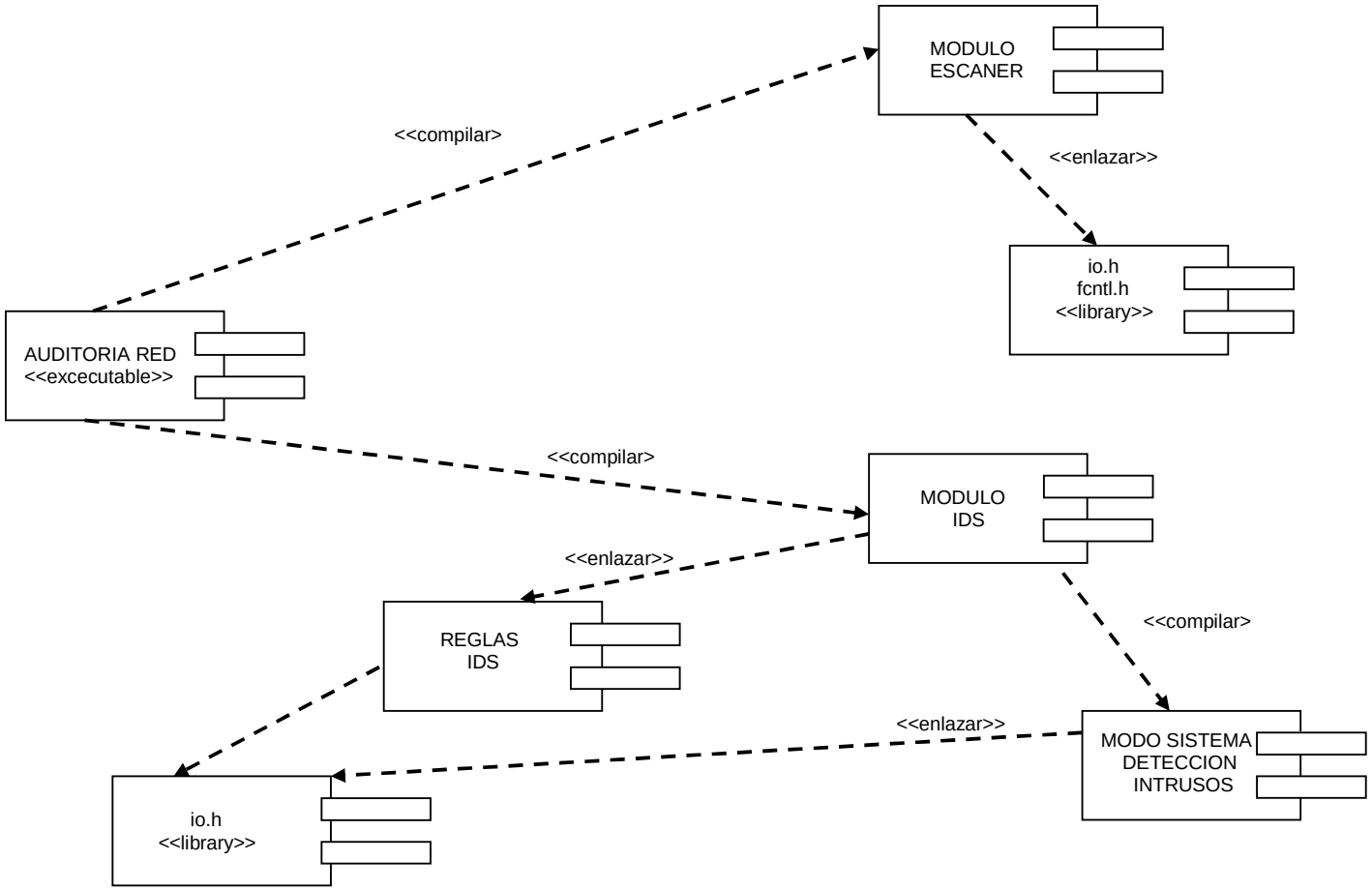

Fig. 2.10 Diagrama de componentes que muestra las dependencias en tiempo de compilación

2.1.8 DIAGRAMA DE COLABORACIÓN

Definido el modelo estático o de clases para la programación, se tiene que definir el modelo conceptual de colaboración, en el cual se muestran los conceptos en el dominio del problema.

En los diagramas de colaboración, figuras 2.11 a 2.14, se indica cómo se relacionan las clases para lograr una tarea específica.

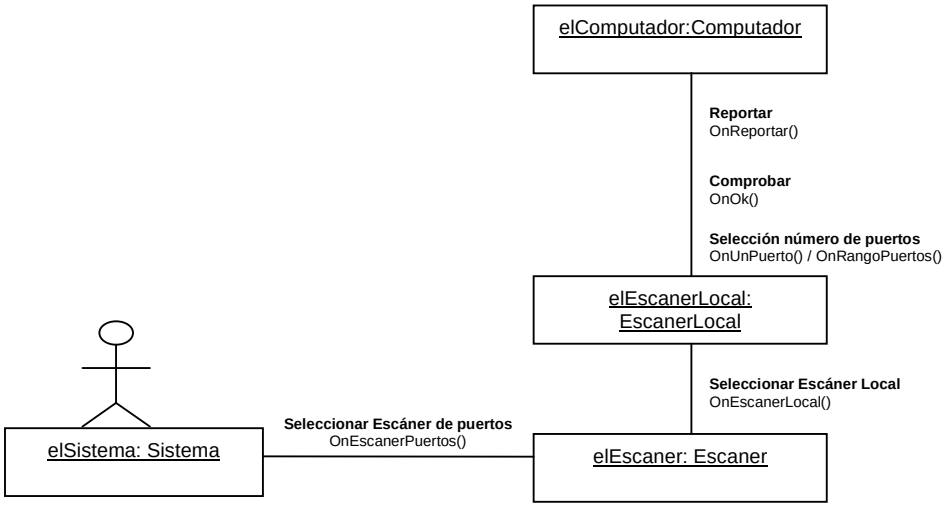

Fig. 2.11 Diagrama de colaboración para pruebas de escaneo local

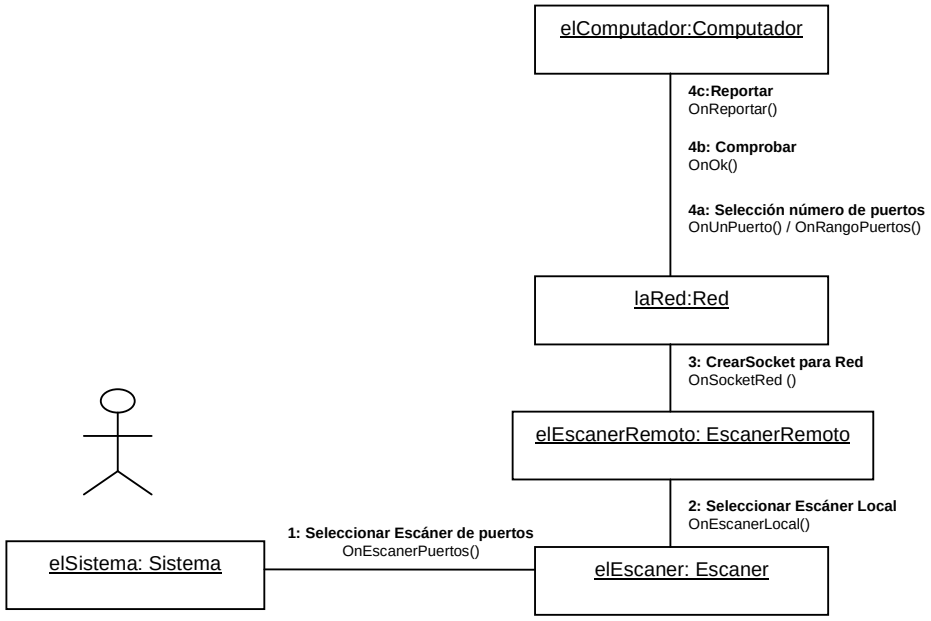

Fig. 2.12 Diagrama de colaboración para pruebas de escaneo remoto

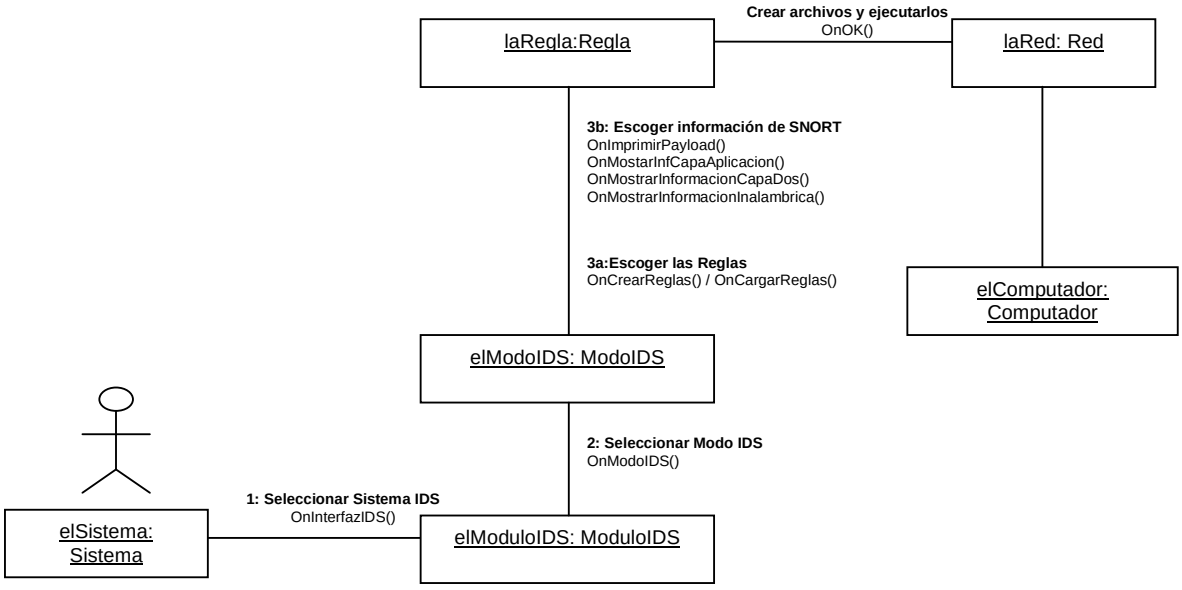

Fig. 2.13 Diagrama de colaboración para prueba *IDS*

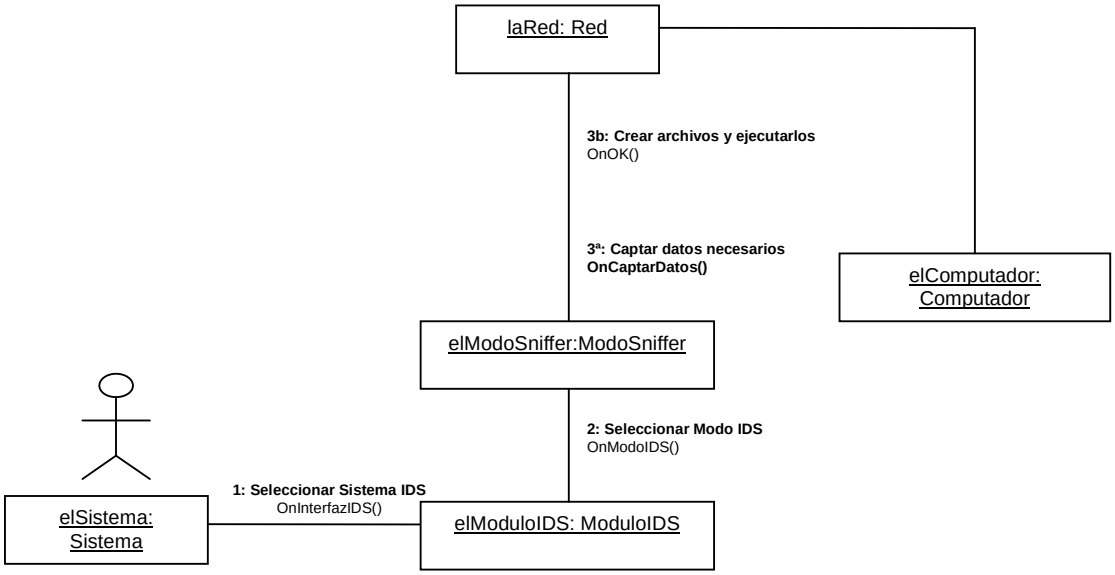

Fig 2.14 Diagrama de colaboración para prueba de *Sniffer*

2.1.9 DIAGRAMA DE NAVEGACIÓN

El diagrama de navegación muestra los posibles caminos que puede seguir el usuario del programa al hacer uso del mismo; este diagrama es de importancia para el programador ya que muestra la relación que existe entre los diferentes diálogos.

El diagrama de navegación de la figura 2.15, es el que se debe tomar en cuenta para la realización de la aplicación, logrando crear una aplicación amigable para el usuario.

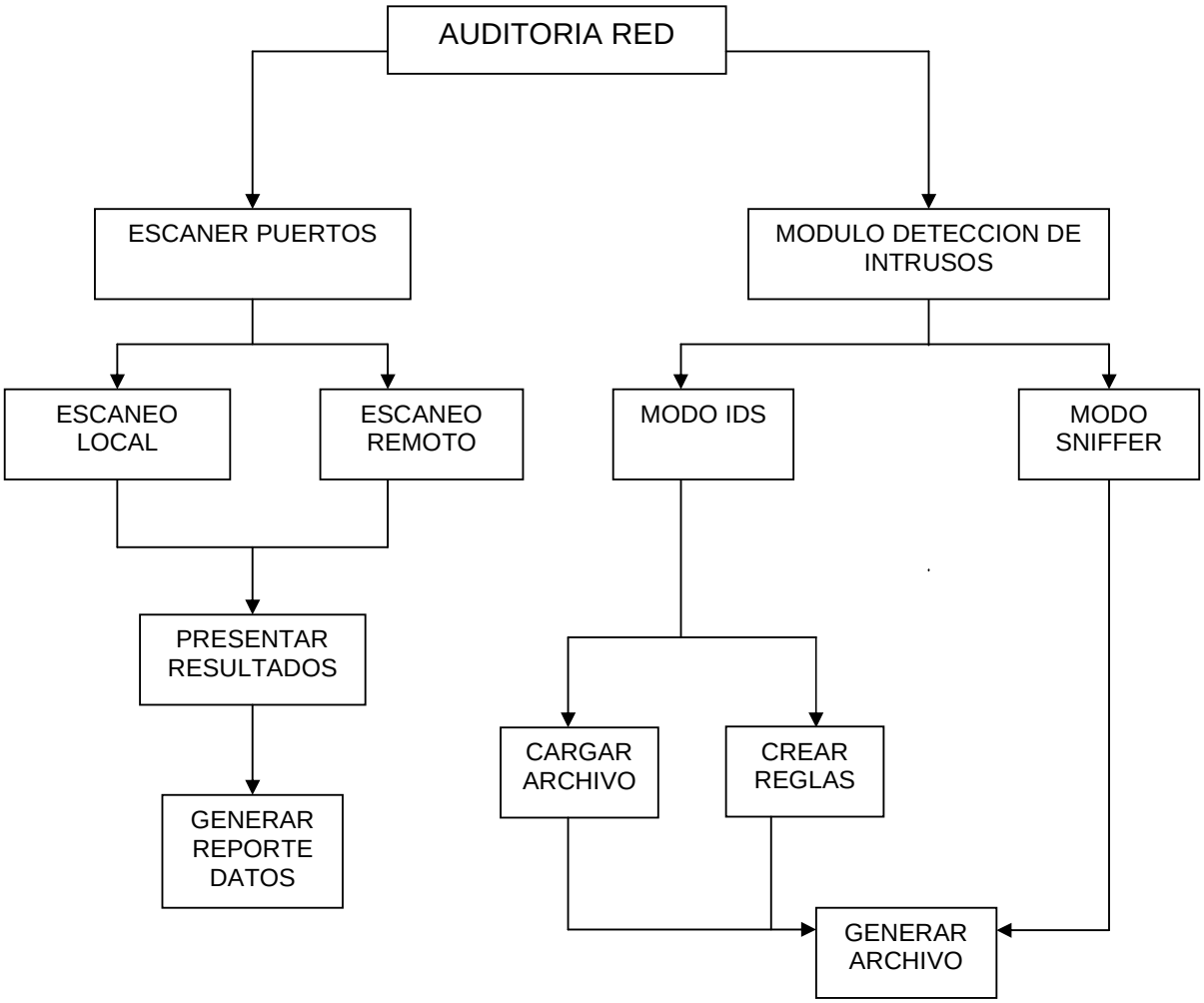

Fig. 2.15 Diagrama de Navegación

2.1.10 DESCRIPCIÓN INTERFAZ DE USUARIO

Uno de los principales problemas en el manejo del sistema de detección de intrusos *Snort,* es que se tiene que ingresar los datos solamente con comandos lo que dificulta su utilización; es por eso que para un manejo más fácil del programa se crearán pantallas gráficas para suprimir en lo mayor posible el ingreso de comandos en forma de texto, sustituyéndolo por medio de botones y herramientas visuales, dando con esto un uso más amigable del sistema.

Otro inconveniente se tenía en los clásicos escáner de puertos, en donde se piden una serie de datos, algunos innecesarios para la prueba; es por eso que se implementará el escáner de puertos de manera que se soliciten los datos necesarios, con esto se hará la prueba de una manera rápida y sencilla desde las interfaces gráficas.

Para poder cumplir con lo expuesto anteriormente se crearán varios diálogos gráficos los cuales serán invocados de manera *DoModal()* (función de visual C++) para que solo se pueda interactuar con éstos y no con otros a la vez, en los cuales se tendrán botones y herramientas visuales necesarios para ingresar y almacenar los datos en las variables correspondientes para su uso posterior; así mismo para "limpiar" los datos se tendrán botones que harán retornar a las variables a sus valores iniciales.

Cada una de estas interfaces van a ser explicadas en el anexo correspondiente al Manual de Usuario, donde se indicará la funcionalidad de cada uno de los diálogos y controles utilizados.

2.2 CODIFICACIÓN Y PROGRAMACIÓN DEL MÓDULO DE INTERFAZ DE CONFIGURACIÓN Y EJECUCIÓN DEL SISTEMA DE DETECCIÓN DE INTRUSOS

Para la codificación y programación de este módulo del programa se crearán y se ocuparán varias clases de Visual C++; se van a mencionar las que se desarrollaron, ya que como se conoce en este lenguaje de programación se crean automáticamente varias clases del sistema.

Del menú principal se podrá invocar al módulo de escáner de puertos o sistema de detección de intrusos, que serán escogidos mediante dos botones, Módulo *IDS* y Módulo Escáner de Puertos.

Dentro del módulo *IDS* se tendrá la opción de escoger los modos de funcionamiento, Modo *IDS* o Modo *Sniffer* que serán invocados mediante los

diálogos relacionados a las clases CModoIDS o CModoSniffer respectivamente.

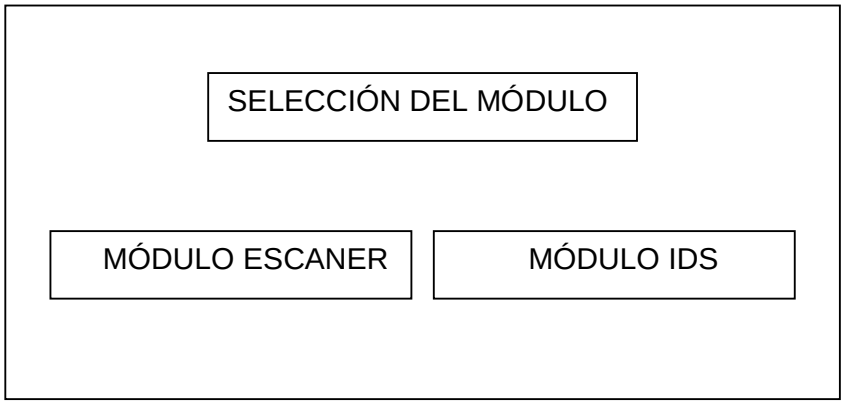

Fig. 2.16 Menú Principal Sistema

Para explicar de una manera más clara la invocación de los diálogos se citará una parte del código de programación:

```
void CModuloIDS::OnModoIDS()
{
        CModoIDS DlgModoIDS;
        DlgModoIDS.DoModal();
}
```

En esta parte se invoca al diálogo del Modo *IDS* mediante la creación de un objeto del tipo CModoIDS llamado DlgModoIDS e invocando a la función *DoModal()* que pertenece a dicha clase.

Se debe resaltar que es de tipo *VOID* porque no retorna ningún valor. De esta manera serán invocados todos los diálogos correspondientes para el ingreso de datos necesarios para la configuración del programa.

Para el manejo de los datos y funciones asociados a cada diálogo, se crearán variables y funciones miembros de cada clase, que se encontrarán

relacionadas a cada *Edit Box, Radio Button, Check box y Buttons* utilizadas en los diálogos.

Por ejemplo, las variables y funciones relacionadas al diálogo de modo Sistema de Detección de Intrusos son las siguientes:

Variables Miembro correspondientes a los *Edit Box*

CEdit m_Log
CNumeroEdit m_NumeroPaquetes

Funciones miembro relacionadas a los *Check Box*

void OnImprimirPayload();
void OnMostrarInfCapaAplicacion();
void OnMostrarInformacionCapaDos();
void OnMostrarInformacionInalambrica();

Funciones miembro relacionadas a los Botones

void OnCargarReglas();
void OnCrearReglas();

Para obtener el dato que escoge el usuario, las funciones miembros relacionadas con los *Check box* manejarán datos boleanos como banderas, que darán un verdadero cuando están seleccionadas y falso cuando están sin seleccionar; con esto se conocerá lo que se escribirá en el archivo a escribir y ejecutar.

Dependiendo del caso, los diálogos pueden invocar a otros diálogos, por ejemplo si se escoge el modo de funcionamiento como *IDS*, se presentará el correspondiente diálogo donde además de la parte de ingreso de datos se tendrá la opción de crear nuevas reglas o escoger reglas preelaboradas, cada una de estas opciones representada por las funciones OnCargarReglas() y

OnCrearReglas() las cuales están asociadas a dos diferentes diálogos que serán llamados de la misma manera explicada.

Una vez que se tengan todos los datos necesarios, se generarán archivos de ejecución o de reglas necesarios, invocando a la clase de Visual C++ conocida como CFileDialog(par1, par2,…) con los parámetros correspondientes, que permite grabar el contenido del archivo de ejecución por lotes para *Snort*, en una ubicación determinada de Windows, esto es:

```
char strFilter[] = { "Archivo de Ejecución por Lotes para Snort (*.bat)|*.bat|" };
CFileDialog* pDlg = new CFileDialog(FALSE, ".rule", NULL, 0, strFilter);
pDlg->m_ofn.lpstrInitialDir="C:\\Snort\\Archivos Ejecución";
```

La primera parte de código anterior char **strFilter[] = { "Archivo de Ejecución por Lotes para Snort (*.bat)|*.bat|" };** hace un filtro para que solo puedan grabarse archivos con extensión (*.bat).

Otra parte del código que se debe recalcar es que se ocupará la variable miembro de la clase **m_ofn.lpstrInitialDir** para que ese diálogo se abra en la carpeta requerida como es: c:\Snort\Archivos de Ejecución\.

Cabe recalcar que para ocupar estas funciones y clases en el proyecto, se debe incluir su correspondiente cabecera (*.h) en el archivo de trabajo (*.cpp) desde donde se lo está invocando.

Finalmente teniendo todos los datos necesarios y los archivos generados se tendrán que ejecutar estos archivos en *DOS.* Para esta tarea del programa se invocará a la función del sistema system (parámetro) donde parámetro es la dirección del archivo creado.

2.3 CODIFICACIÓN Y PROGRAMACIÓN DEL MÓDULO DE ESCANEO DE PUERTOS

Igual que el módulo anterior, para la explicación de este módulo se presentará las principales funciones que se van a utilizar.

En este módulo se creará una clase relacionada con el diálogo que contenga toda la información correspondiente y necesaria para ingresar los datos y hacer la prueba de puertos abiertos y cerrados. Este diálogo estará relacionado a la Clase CModuloEscaner() y contendrá las siguientes variables y funciones miembro:

Funciones miembros

```
void OnEquipoRemoto();
void OnEquipoLocal();
void OnUnPuerto();
void OnRangoPuertos();
void OnLimpiar();
void OnOK();
void OnReportar ();
```

Cada una de estas funciones tendrá una funcionalidad que será comentada en el código fuente de la aplicación.

Variables miembro

```
CIPAddressCtrl m_IpAddress;
CEdit  m_nPuerto;
CEdit  m_nPuertoAlto;
CEdit  m_nPuertoBajo;
CEdit  m_nIntento;
CListCtrl m_ListCtrlResultados;
```

Todos estos datos se captarán y grabarán en un nodo de almacenamiento que contendrán las variables importantes como son (dirección *IP*, número de puerto

y estado de puerto) teniendo así toda la información para ser usada en cualquier instancia del programa.

Para poder manejar la información de manera práctica lo que se hará es tomar todos los datos de las casillas *Edit Box*, *IP ADDRESS* y transformarlas en variables de tipo *CString* para su manipulación, este procedimiento se lo realizará mediante la función miembro GetWindowText().

Lo descrito se indica a continuación.

```
m_nPuerto.GetWindowText(m_strSoloPuerto);
m_IpAddress.GetWindowText(m_strIP);
```

Una vez que se tengan los datos necesarios para realizar las pruebas, se tomarán éstos para la creación de *SOCKETS*[3]; los datos usados para este procedimiento son la dirección *IP* y el puerto.
El proceso se indica a continuación:

```
bool CModuloEscaner::SocketRed(CString m_strIP, int m_intPuerto)
{
	CTheSocket* pSocket;
	pSocket = new CTheSocket;
	ASSERT(pSocket);
	if (!pSocket->Create())
	{
		delete pSocket;
		pSocket = NULL;
		return FALSE;
	}
	while (!pSocket->Connect(m_strIP , m_intPuerto))
	{
		delete pSocket;
```

[3]*Socket:* Número de identificación compuesto por dos números: la dirección IP y el número de puerto TCP. En la misma red, el número IP es el mismo, mientras que el número de puerto es el que cambia. En máquinas de distintas redes, pueden tener el mismo número de puerto sin llevar a confusión, pues el número IP las distingue.

```
        }
        pSocket->Close();
        delete pSocket;
        return TRUE;
}
```

Para poder utilizar correctamente esta función se tendrá que incluir la librería correspondiente en el archivo (*.cpp) del módulo de donde se está llamando. En la función se puede notar que los parámetros necesarios son los mencionados, los cuales deben ser enviados como *Strings* para su utilización, tal como lo exige la función del sistema.

Como se nota, para conocer el estado del puerto se ocupará una variable boleana en donde si se logra una conexión el valor devuelto será *TRUE*, caso contrario *FALSE*, así se podrá hacer la conversión a parámetro de escritura como texto abierto o cerrado y se presentará en la lista correspondiente del diálogo.

En este módulo del programa los resultados obtenidos podrán o no ser guardados; para exportar los resultados a un archivo de excel (*.xls) se tendrá un botón con el texto “reportar” el cual llamará la función OnReportar() que se encargará de generar el archivo correspondiente.

Para evitar que se puedan realizar pruebas fallidas y malos funcionamientos, se validarán los datos que deben ser ingresados, esto es, en los casos que se encuentren vacíos o escritos de una manera no lógica, mostrando mensajes que indicarán dicho problema.

2.4 REQUERIMIENTOS MÍNIMOS DE SOFTWARE Y HARDWARE DE LOS EQUIPOS

Para una correcta ejecución del programa en la máquina, ésta debe cumplir con requerimientos mínimos de hardware y software, los cuales se van a determinar a continuación.

2.4.1 REQUERIMIENTOS DE HARDWARE [12]

Consideración del lenguaje de programación Visual C++

La siguiente lista muestra los requerimientos hardware y software mínimos necesarios para ejecutar un software compilado por la versión de 32 bits del compilador Visual C++ de Microsoft optimizando el ciclo de desarrollo de dichos programas.

PC con procesador Pentium velocidad de 200 MHz (o superior).
32 MB de RAM.
1 GB de espacio de disco duro.
Monitor Super VGA.

Consideraciones de pruebas de ejecución

Se considerará que en las pruebas correspondientes de ejecución del software no se utilizará gran cantidad de recursos informáticos como son memoria y procesador, ya que el administrador de tareas de Windows al momento de la utilización de la aplicación tiene los siguientes datos:

Memoria Usada: 16 MB
Porcentaje de CPU Usada: 2 % de una computadora con CPU de 3.0 Ghz.

Consideración de espacio de aplicaciones necesarias

Para la determinación del espacio necesario se toman en cuenta el tamaño de las aplicaciones dependientes y de la aplicación principal; los datos son los que se muestran a continuación:

Programa Elaborado: 4 MB
Snort: 3 MB
Ethereal: 50 MB
WinpCap: 1 MB

Entonces tomando en cuenta todas las consideraciones expuestas se llega a la conclusión que se necesitará en relación a Hardware:

Computadora Pentium 200 Mhz o Superior.
Memoria RAM de 32 MB
Espacio mínimo en Disco Duro de 1 GB
Tarjeta de Red 10/100 Mbps
Tarjeta *Wireless* (Opcional)
Tarjeta de Fax Modem (Opcional)

Además se resalta que el software permitirá realizar pruebas a computadoras o equipos de red que se encuentren interconectados mediante una red de datos, sea ésta cableada o inalámbrica siempre y cuando pertenezcan a la misma red física y lógica.

En el caso del escáner de puertos remoto necesariamente se requerirá la dirección *IP* del equipo remoto, para poder identificar de manera lógica a la máquina que pertenece a la red de datos.

En el caso del sistema de detección de Intrusos se podrá analizar el tráfico según las reglas creadas; para poder analizar todo el tráfico circundante en toda la organización se tendrá que configurar un puerto de un *switch* como espejo, o simular esto mediante un *Hub* o concentrador, tal como se mencionó anteriormente.

2.4.2 REQUERIMIENTOS DE SOFTWARE [12]

Considerando que el programa es realizado en una aplicación MFC con visual C++, compilando con librerías estáticas lo que involucra independencia de archivos DLL y de sistema operativo Microsoft, se determina los siguientes requerimientos:

Sistema Operativo Windows 98 o superior.
Programa WinCap (preinstalado)
Programa *Snort* - Sistema de detección de intrusos (preinstalado)
Programa Ethereal (Opcional)

2.5 DOCUMENTACIÓN DEL SOFTWARE

Debido a la extensión del código de programación en esta parte del capítulo, solo se van a enlistar las clases creadas por Visual C++ y las elaboradas manualmente; como se conoce una clase debe tener archivos .h (encabezado) y .cpp (implementación).

El código de programación creado esta comentado y se encuentra en el anexo correspondiente.

Clases Ocupadas en la elaboración del Software

CAboutDlg (Creada por Visual C++)
CAuditoriaRedApp (Creada por Visual C++)
CAuditoriaRedDoc (Creada por Visual C++)
CCreacionReglas (Clase relacionada con el diálogo para creación de nuevas reglas).
CMainFrame (Creada por Visual C++)
CModoIDS (Clase relacionada con el diálogo de ingreso de datos en el modo *IDS*)

CModoSniffer (Clase relacionada con el diálogo de ingreso de datos en el modo *Sniffer*)

CModuloEscaner (Clase relacionada con el diálogo de opción de su mismo nombre)

CModuloIDS (Clase relacionada con el diálogo de opción de su mismo nombre)

CSocketRed (Clase derivada de CSocket para la creación de *SOCKETS* y probar conexión relacionado con la red)

AUDITORÍAS, PRUEBAS DE FUNCIONAMIENTO Y ANÁLISIS DE RESULTADOS

3.1 AUDITORÍA DE LA RED OPERANTE

Como se mencionó en la introducción de este proyecto de titulación, la auditoría de seguridad de redes de datos es algo que las empresas deben hacer e implementar obligatoriamente para poder operar de una manera más confiable. El brindar seguridad en la red informática da más confianza a la Institución y atrae a nuevos clientes, los cuales son la razón de ser de las organizaciones.

Las falencias de seguridad de las redes informáticas radican principalmente en varias áreas, entre las cuales se van a citar las siguientes:

- Factor humano
- Factores de directivas
- Configuración incorrecta del hardware o software
- Suposiciones infundadas
- Desconocimiento del tema
- No mantenerse actualizado

La percepción de los ejecutivos no entendidos en el tema, es creer que los virus informáticos representan el mayor riesgo, aunque la realidad indica que el fraude es la práctica más peligrosa [15].

Aunque la industria tecnológica aumenta los esfuerzos para investigar en materia de seguridad informática, los máximos responsables de las compañías no toman conciencia sobre el asunto; para el 80% de las organizaciones el tema no es prioritario [14].

Uno de los principales problemas es que no ha cambiado significativamente el cuidado por la seguridad informática en las empresas, mientras que los riesgos han crecido considerablemente en los últimos años.

También se puede mencionar que las empresas siguen sin reaccionar después de haberse generado un problema, y que las inversiones en seguridad informática aparecen después de existir un problema grave en la organización.

La aparición de nuevos virus, una inadecuada utilización de los sistemas, la pérdida de confidencialidad y el *Spam* (envío masivo de *e-mails* no solicitados) producen la negación de servicio, situación que puede poner en riesgo la evolución de un negocio.

En el 67% de las empresas, se considera muy importante a las políticas de seguridad informática; sólo el 15% reporta a los directorios de las empresas el estado de seguridad o incidentes informáticos sucedidos cada mes. Un 16% lo hace cada cuatro meses, un 8% cada seis meses, un 11% nunca y el 10% sólo una vez por año [15].

Al ser consultadas sobre los obstáculos que encuentran para implementar un plan efectivo en la materia, destacan que hay falta de conciencia en los usuarios finales, limitaciones de presupuesto, indisponibilidad de personal especializado, dificultad para probar el valor de la seguridad informática (el retorno de la inversión) y el ritmo acelerado de los cambios tecnológicos.

El nivel de concientización entre los empleados de una organización crece en la medida que la organización lo establece como política prioritaria.

3.1.1 ANÁLISIS DE LA RED DE PRUEBA

La red de datos utilizada para las pruebas del programa, tiene la estructura que se muestra en la figura 3.1.

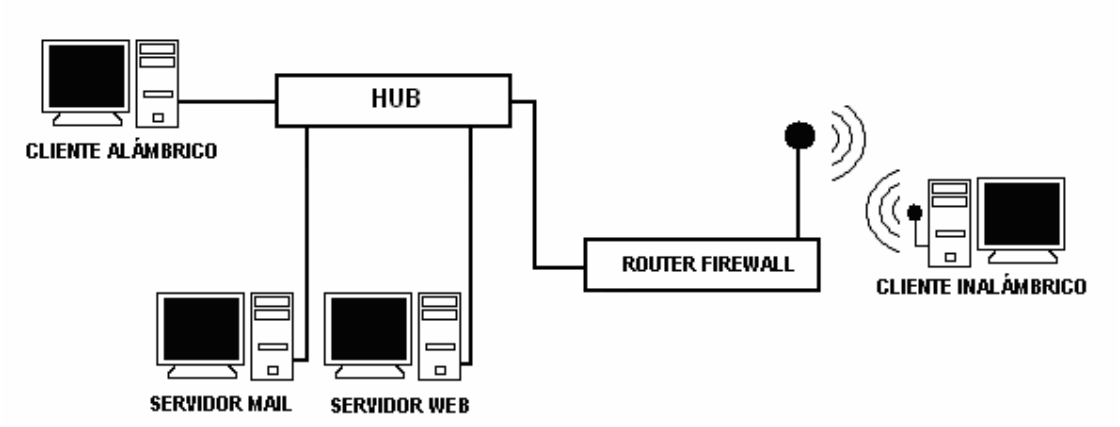

Fig. 3.1 Esquema de la red de prueba

Como se puede observar, el sistema consta de tres máquinas con conexión alámbrica, dos de las cuales son servidor *Web* y *Mail* y una cliente; adicionalmente se tiene una máquina con conexión inalámbrica también como cliente. El equipo de interconectividad de los computadores alámbricos e inalámbricos es un *router firewall* inalámbrico, configurado para realizar las pruebas. Para poder captar todo el tráfico circulante en la red se ocupa un *hub* que ayudará a simular un puerto espejo; además todas las estaciones y servidores mencionados emplearán plataforma Microsoft.

Mas específicamente relacionado con las falencias de seguridad de una red de datos, la auditoría de seguridad analizará los siguientes puntos críticos:

- La configuración de los sistemas operativos implantados: usuarios, ficheros.
- La fiabilidad de las contraseñas utilizadas.
- La configuración del software y hardware (*Web*, *Mail*, *firewall*).
- Determinación de las vulnerabilidades presentes en los sistemas debido a la desactualización en la aplicación de “parches” de seguridad.

Luego de este análisis, se completará la auditoría con el funcionamiento del programa. Se realizarán pruebas para encontrar posibles intrusiones, mediante

los puertos abiertos, tráfico mal intencionado y mala configuración que facilitan cualquier tipo de ataques, como los mencionados en el capítulo número uno.

3.1.1.1 Configuración de los sistemas operativos implantados: usuarios, ficheros

En cada una de las máquinas, independientemente del sistema operativo, se pueden tener creadas varias cuentas de usuarios y sistemas de ficheros, que de una manera pueden ser problemas para la seguridad de la organización, ya que si se crean cuentas de usuarios con privilegios de administración se pueden modificar configuraciones importantes que alterarían el funcionamiento de los sistemas.

Uno de los grandes errores en la administración de las redes se da cuando las personas que cambian de cargo o a veces son despedidas de la organización por cualquier razón, siguen teniendo sus cuentas de administración activas en los equipos, las mismas que al no eliminarse por descuido de los administradores, pueden producir alteraciones de configuración de equipos importantes.

3.1.1.2 La fiabilidad de las contraseñas utilizadas

Uno de los puntos problemáticos en la seguridad de redes radica principalmente en la fragilidad de la memoria humana relacionado directamente a las contraseñas. Esto conlleva al usuario a seleccionar contraseñas fáciles de recordar que normalmente coinciden con las que son fáciles de adivinar, sea porque son palabras muy comunes, o porque están íntimamente relacionadas con su dueño (la fecha de aniversario, de nacimiento, el nombre del hijo, etc.).

Pero la realidad indica que no es tan complicado utilizar buenas contraseñas que a la vez no sea difícil de recordar.

¿Pero que es una buena contraseña? una buena contraseña es aquella que es difícil de averiguar, a la vez que es fácil de recordar para el usuario (así no hace falta apuntarla en ningún sitio).

Existen normas comúnmente aceptadas para que una contraseña pueda ser calificada de segura. Una contraseña segura debería contener:

- Al menos seis caracteres (mejor a partir de ocho) y ser alfanumérica
- Debe ser combinación de letras (mayúsculas y minúsculas), números y símbolos
- Si es posible, utilizar una frase completa.

3.1.1.3 La configuración del software y hardware

Programas instalados en equipos importantes como son los servidores de aplicaciones tales como correo electrónico, páginas *Web*, bases de datos, pueden ocasionar varios problemas como: consumo de recursos informáticos del equipo (memoria, espacio en disco duro, procesamiento del CPU), abrir puertos innecesarios, que dejan puertas abiertas para ataques de personas mal intencionadas.

Además si se tienen tarjetas extras sean éstas de red ethernet, inalámbricas (*WIFI*), fax modem, las mismas que pueden permitir conectarse a otras redes inseguras y ajenas a la organización, se puede ganar acceso a áreas o segmentos de red críticos de la organización, ocasionando problemas a la operabilidad del sistema.

3.1.1.4 Determinación de las vulnerabilidades presentes en los sistemas debido a la desactualización en la aplicación de “parches” de seguridad

Posiblemente no exista software en el mundo que se encuentre libre de manifestar un error en algún momento del futuro. Los errores existen en todos los programas, aunque aún no hayan sido descubiertos, hay gente investigando en todos los aspectos del funcionamiento, analizando meticulosamente cada pequeño detalle; por lo que se puede afirmar que tarde o temprano alguien, en algún lugar, encontrará una forma de aprovechar un error hasta entonces no descubierto en el software.

Cuando se instala un sistema nuevo, normalmente se utiliza una versión muy reciente de software, tanto en el sistema operativo como en las aplicaciones del servidor. Si es así, este servidor (suponiendo que esté bien configurado) no manifestará problemas de seguridad durante algún tiempo. Pero sin duda llegará el día en que alguien descubra y reporte una vulnerabilidad en alguno de los programas o en el propio Kernel del sistema.

Cualquiera de estos programas que manifieste un error está exponiendo nuestro servidor o computador personal al ataque de un intruso. Casi al mismo tiempo de descubierta la vulnerabilidad, el autor del software lanzará una nueva versión, o en su defecto un “parche” o actualización para solucionar el problema de seguridad en versiones ya instaladas sin necesidad de sustituirlas.

Entonces, es en este punto donde surge la disyuntiva que puede salvar los sistemas, o condenarlos a la intrusión de *hackers*; si el administrador de sistemas se entera de la existencia del problema aplicará el “parche”. Si no se entera entonces tarde o temprano un intruso hará uso del agujero de seguridad.

3.1.2 ANÁLISIS DE LAS MÁQUINAS QUE CONFORMAN LA RED DE DATOS

Las máquinas presentes de la red de datos son equipos con plataforma Microsoft y se describen a continuación.

- Servidor de correo electrónico opera en una máquina con sistema operativo Windows 2003 Server.

- Servidor de páginas Web opera en una máquina con sistema Operativo Windows 2000 Server.

- Clientes de la organización tienen instalado el sistema Windows XP Professional

3.1.2.1 Resumen de Máquinas de la red de datos

EQUIPO / SERVICIO	SISTEMA OPERATIVO	CUENTAS DE USUARIO	SISTEMA DE FICHEROS	FIABILIDAD CONTRASEÑA	CONFIGURACIÓN DE HARDWARE	CONFIGURACIÓN DE SOFTWARE
Servidor de correo electrónico	Windows 2003 Server	Administrador	No levantado ningún sistema	Formato común y simple	Computador PIV con tarjetas de video, sonido, unidad óptica CD-R, tarjeta de red ethernet, el equipo se conecta mediante dicha tarjeta a la red de datos de la organización, no se tiene otra tarjeta de conexión.	Levantado el Servidor de Correo electrónico se lo considera de propósito particular ya que solo tiene objeto de administrar correo electrónico dentro de la organización. Instalado antivirus
Servidor de páginas Web.	Windows 2000 Server	Administrador	No levantado ningún sistema	Formato común y simple	Computador PIV con tarjetas de video, sonido, unidad óptica CD-R, tarjeta de red ethernet, el equipo se conecta mediante dicha tarjeta a la red de datos de la organización, no se tiene otra tarjeta de conexión.	Levantado el Servidor de páginas Web; se lo considera de propósito particular ya que solo tiene el servicio de http para la organización. Instalado antivirus.
Cliente 1	Windows XP Professional	Administrador	No levantado ningún sistema	Formato común y simple	Computador PIV con tarjetas de video, sonido, unidad óptica CD-RW, tarjeta de red ethernet, tarjeta de fax Modem; el equipo se conecta mediante la tarjeta ethernet a la red de datos de la organización.	Instalada varias aplicaciones, las cuales serán detalladas más adelante.
Cliente 2	Windows XP Professional	Administrador	No levantado ningun sistema	Formato común y simple	Laptop PIV con tarjetas de video, sonido, unidad óptica DVD + CD-RW, tarjeta de red ethernet, tarjeta de fax Modem y tarjeta de red Wireless; el equipo se conecta mediante la tarjeta inalámbrica a la red	Instalada varias aplicaciones, las cuales serán detalladas más adelante.

					de datos de la organización.	
Router Wireless Belkin 802.11 b/g	-	Cuenta de administración	-	Formato de la contraseña que usa es simple	Consta de 4 puertos LAN y 1 puerto WAN (conexión a Internet) todos con interfaces RJ45. El puerto WAN (Internet) se lo conecta a un punto de la red de datos existente en el edificio, pudiendo dar una independencia de la red interna de la organización con la red del edificio.	

Tabla 3.1 Análisis de las máquinas en la red de datos

Problemas encontrados en el Servidor de correo electrónico

Las actualizaciones automáticas para el sistema operativo no se encuentran levantadas, es por eso que se tienen problemas de no estar al día con los diferentes tipos de "parches" que el fabricante da para evitar problemas de vulnerabilidades del sistema, como son accesos remotos, ejecución de códigos remotos, acceso a información confidencial, etc.

Se encuentra instalado un antivirus que no se actualiza automáticamente.

Problemas encontrados en el Servidor de páginas *Web*

Las actualizaciones automáticas para el sistema operativo no se encuentran activadas, es por eso que, igual que el anterior servidor, no se está al día con los "parches" de seguridad que publica el fabricante, lo único que se detecta es que se tiene instalado el *Service Pack 2* para Windows 2000 Server el cual ya se encuentra obsoleto, puesto que al momento de escribir este proyecto de titulación se tenía el *Service Pack 3*.

Se encuentra instalado un antivirus que no se actualiza automáticamente.

Problemas encontrados en los clientes Alámbricos e Inalámbricos

Al tener tarjetas de fax modem y tarjetas de red extras instaladas en los equipos de los usuarios, se tiene la posibilidad ya sea por desconocimiento o

por mala intención de conectarse a Internet o a algún servicio *RAS (Remote Access Service)*, evitando las políticas del *firewall* de la organización, lo que pone en peligro al equipo y por ende la red de la organización.

Las actualizaciones automáticas para el sistema operativo y aplicaciones no se encuentran activadas, es por eso que igual que los servidores no se está al día con los "parches" de seguridad, lo único que se detecta es que se tiene instalado el *Service Pack 2*, pero conociendo que existen actualizaciones semanales y hasta diarias, es un problema de seguridad.

Uno de los principales problemas es el poseer privilegios de administrador en sus equipos, tener instalados una serie de software peligroso, como son Internet Relay Chat (*IRC),* programas *Peer to Peer* (*P2P)* para descarga de archivos innecesarios; esto determina que se desperdicie el ancho de banda de conexión a Internet, así como el descuido de los usuarios al estar ocupando programas de Chat en horas de oficina. Cabe recalcar que el problema no se da en todas las máquinas pero con la ayuda del software desarrollado se podrá conocer qué máquinas (mediante la dirección de red *IP*) están ocupando las aplicaciones.

Uno de los principales problemas de todo lo antes mencionado, es que se puede derivar en una serie de problemas de virus y troyanos alterando la operabilidad del sistema y conllevando a un fraude informático.

Problemas encontrados en el *Router Wireless Belkin* 802.11 b/g

Se puede notar dos principales problemas en el *router wireless,* el *firewall* interno del dispositivo está configurado con políticas de seguridad de manera restrictiva y no de manera permisiva, la cual es la mejor opción de configuración; el otro problema es que no se encuentra levantada una clave de encripción de la señal inalámbrica, lo que implica que cualquier persona ajena a la organización y que tenga una máquina con tarjeta de red inalámbrica puede ingresar a la red.

3.2 PRUEBAS DE FUNCIONAMIENTO DEL SOFTWARE DESARROLLADO EN LA RED

Para complementar el trabajo realizado del análisis de los elementos, se hicieron pruebas de puertos abiertos en las computadoras pertenecientes a la red, mediante el módulo de escáner de puertos del software desarrollado.

Como se conoce, para poder realizar estas pruebas se necesitan las direcciones *IP* de las máquinas pertenecientes a la red y un número de puerto; para que el análisis sea más técnico se necesitará además conocer el servicio que está prestando cada máquina.

Como es lógico en el caso del servidor de páginas *Web* solo debería estar abierto el puerto 80 que está relacionado con la aplicación *HTTP (Hyper Text Transfer Protocol)*, igualmente en el servidor de correo electrónico debe estar abierto el puerto 110, *POP3 (Post Office Protocol 3)*.

La figura 3.2 muestra las direcciones y tipo de configuración de cada una de las computadoras.

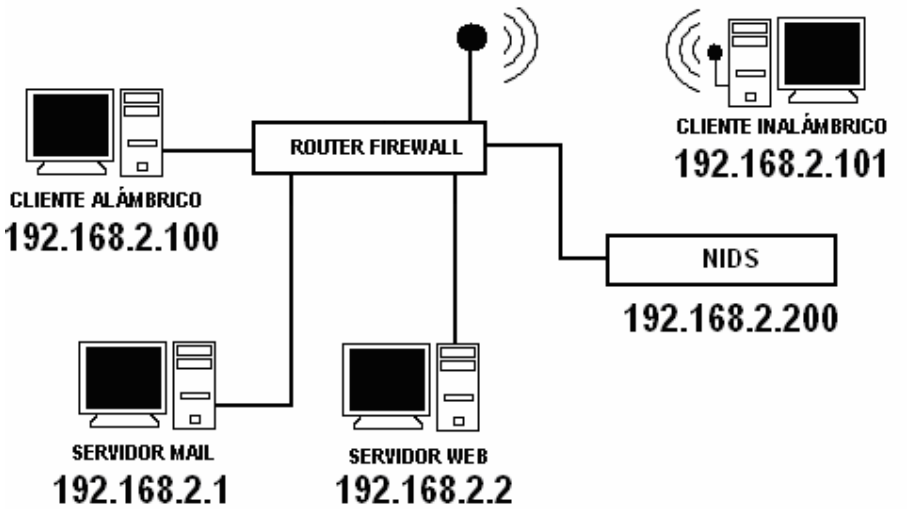

Fig. 3.2 Direcciones de los equipos de la red

Para realizar estas pruebas se hizo funcionar el módulo de escáner de puertos, desde una máquina auditora hacia cada una de las máquinas remotas

pertenecientes a la red de datos de la organización, tal como se indica en el esquema de la figura 3.3.

Por ejemplo para comprobar si el puerto 80 (*HTTP*) en la máquina servidor de páginas *Web* está abierto, se realizó lo siguiente: en el menú principal se escogió el módulo Escáner de Puertos, se seleccionó prueba remota y se llenaron todos los campos necesarios con los datos, que para el caso de la prueba fueron:

Dirección *IP*: 192.168.2.2, Puerto: 80, Número de intentos: 1

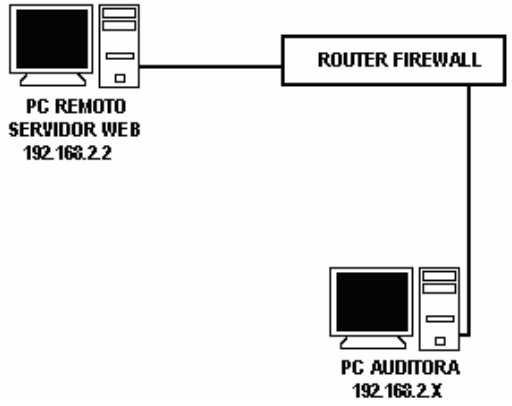

Fig. 3.3 Esquema de prueba

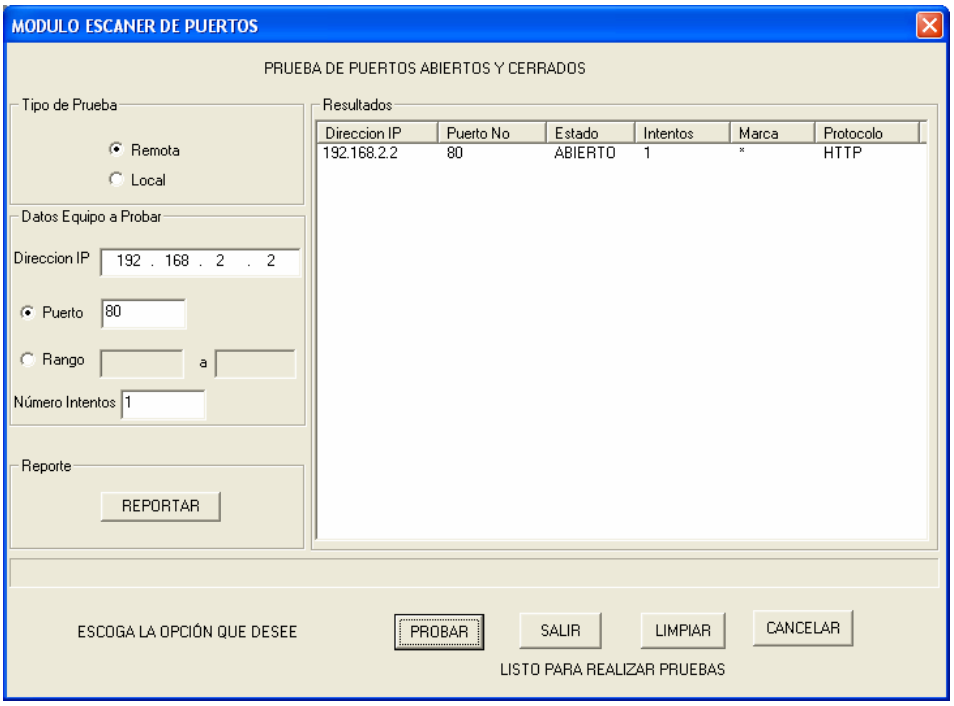

Fig. 3.4 Datos necesarios para prueba remota a la máquina 192.168.2.2

Al realizar la prueba, se desplegó el resultado en la lista correspondiente. Esta lista consta de la dirección *IP* del elemento de red, el puerto probado, el estado del puerto, número de intento, la marca, la cual será "*" cuando esté abierto y blanco en caso de estar cerrado, seguido del nombre del protocolo si es que es conocido (ver figura 3.4).

De la misma manera citada en el ejemplo, se hicieron pruebas a los puertos comunes de aplicaciones *HTTP, SMTP* (*Simple Mail Transfer Protocol*), *SNMP* (*Simple Network Management Protocol*), *POP3, DNS* (*Domain Name Server*), *SSL* (*Security Socket Layer*) entre otros, en cada una de las máquinas pertenecientes a la red de datos.

Los resultados obtenidos son los que se encuentran tabulados en las tablas 3.2 a 3.5; además los reportes del programa se encuentran en el anexo de pruebas.

192.168.2.1 (SERVIDOR MAIL)	21	FTP	CERRADO
	23	TELNET	CERRADO
	25	SMTP	CERRADO
	53	DNS	CERRADO
	80	HTTP	CERRADO
	110	POP3	ABIERTO
	135	RPC	ABIERTO
	139	Netbios-ssn	ABIERTO
	162	SNMP	CERRADO
	161	SNMPToadNode	CERRADO
	443	SSL	CERRADO

Tabla 3.2 Resultados de pruebas en Servidor de correo electrónico

192.168.2.2 (SERVIDOR WEB)	21	FTP	CERRADO
	23	TELNET	CERRADO
	25	SMTP	CERRADO
	53	DNS	CERRADO
	80	HTTP	ABIERTO
	110	POP3	CERRADO
	135	RPC	ABIERTO
	139	Netbios-ssn	CERRADO
	162	SNMP	CERRADO
	161	SNMPToadNode	CERRADO
	443	SSL	ABIERTO

Tabla 3.3 Resultados de pruebas en Servidor de Páginas Web

192.168.2.100 (CLIENTE)	21	FTP	ABIERTO
	23	TELNET	CERRADO
	25	SMTP	CERRADO
	53	DNS	CERRADO
	80	HTTP	ABIERTO
	110	POP3	CERRADO
	135	RPC	ABIERTO
	139	Netbios-ssn	CERRADO
	162	SNMP	CERRADO
	161	SNMPToadNode	CERRADO
	443	SSL	ABIERTO

Tabla 3.4 Resultado de pruebas en Cliente 1

192.168.2.101 (CLIENTE)	21	FTP	ABIERTO
	23	TELNET	ABIERTO
	25	SMTP	CERRADO
	53	DNS	CERRADO
	80	HTTP	ABIERTO
	110	POP3	CERRADO
	135	RPC	ABIERTO
	139	Netbios-ssn	CERRADO
	162	SNMP	CERRADO
	161	SNMPToadNode	CERRADO
	443	SSL	ABIERTO

Tabla 3.5 Resultado de pruebas en Cliente 2

A continuación se van a mencionar algunas estadísticas importantes para entender la importancia de probar los puertos de las aplicaciones no comunes, y la futura prohibición de estas aplicaciones.

3.2.1 ESTADÍSTICAS RELACIONADAS A LA UTILIZACIÓN DE PROGRAMAS P2P

Para tener una idea amplia de los problemas más comunes en las organizaciones dependientes de Internet se citan las siguientes estadísticas:

¿Dónde utilizan más el intercambio de ficheros P2P los usuarios? [14]

En el trabajo: 38,6 %

En casa: 61,4 %

Promedio de tiempo al día que emplean los usuarios de P2P en el intercambio de archivos [14]

16 minutos o más: 69,7 %
15 minutos o menos: 30,3 %

Entonces teniendo en cuenta las estadísticas planteadas, se priorizará el análisis para comprobar si se están utilizando aplicaciones no recomendadas como son los programas *P2P*, mensajería instantánea y otro tipo de descargas. Una de las maneras más fáciles de identificar este tipo de programas es analizando los puertos utilizados en la conexión de estas aplicaciones.

Aplicaciones P2P	Puerto comúnmente utilizado
Aimster: 5025	5025
Bearshare	6346
EDonkey	4661, 4662, 4665, 4672
EMule	4661, 4662, 4665, 4672
Gnutella	6346
Grokster	1214
Hotline	1234, 5498, 5499, 5500, 5501
Kazaa	1214
LimeWire	6346
Morpheus	1214
ToadNode	6346
Xolox	6346
Direct Connect	411, 412

Tabla 3.6 Puertos de Aplicaciones P2P

Aplicaciones de descargas	Puerto comúnmente utilizado
Bit torrent (puerto de tracker)	6969
Bit torrent (puerto por defecto)	6881

Tabla 3.7 Puertos de Aplicaciones de descargas

Mensajería Instantánea	Puerto comúnmente utilizado
MSN Messenger	1863
AOL / AOL Instant Messenger	5190
IRC (*Internet Relay Chat*)	6667

Tabla 3.8 Puertos de Aplicaciones de mensajería instantánea

3.2.2 RESULTADOS DE PRUEBAS DE MÓDULO ESCÁNER DE PUERTOS EN LAS MÁQUINAS DE LA RED

Por la razón explicada anteriormente, se hicieron pruebas de los puertos de las aplicaciones no recomendadas; las pruebas se las realizó a cada una de las máquinas pertenecientes a la red de datos. Los resultados de las pruebas constan en el anexo 3; aquí se encuentran tabulados los datos obtenidos.

EQUIPO DE RED	PUERTO	APLICACIÓN	RESULTADO
192.168.2.1 (Servidor Mail)	4662	Emule	CERRADO
	6346	BearShare Gnutella Limwire Xdox	CERRADO
	5025	Aimster	CERRADO
	1214	Grokiler Kazza	CERRADO
	1234	Hotline	CERRADO
	5498	Hotline	CERRADO
	6343	ToadNode	CERRADO
	411	Directconnect	CERRADO
	412	Directconnect	CERRADO
	1863	MSN Messenger	CERRADO
	6667	IRC	CERRADO
	6881	Bit torrent (1)	CERRADO
	6969	Bit torrent (2)	CERRADO

Tabla 3.9 Resultados de pruebas aplicaciones prohibidas en Servidor *Mail*

EQUIPO DE RED	PUERTO	APLICACIÓN	RESULTADO
192.168.2.2 (Servidor Web)	4662	Emule	CERRADO
	6346	BearShare Gnutella Limwire Xdox	CERRADO
	5025	Aimster	CERRADO
	1214	Grokiler Kazza	CERRADO
	1234	Hotline	CERRADO
	5498	Hotline	CERRADO
	6343	ToadNode	CERRADO
	411	Directconnect	CERRADO
	412	Directconnect	CERRADO
	1863	MSN Messenger	CERRADO
	6667	IRC	CERRADO
	6881	Bit torrent (1)	CERRADO
	6969	Bit torrent (2)	CERRADO

Tabla 3.10 Resultados de pruebas aplicaciones prohibidas en Servidor Web

Clientes de la red de datos

EQUIPO DE RED	PUERTO	APLICACIÓN	RESULTADO
192.168.2.100 (Cliente 1)	4662	Emule	ABIERTO
	6346	BearShare Gnutella Limwire Xdox	ABIERTO
	5025	Aimster	CERRADO
	1214	Grokiler Kazza	POSITIVO
	1234	Hotline	CERRADO
	5498	Hotline	CERRADO
	6343	ToadNode	CERRADO
	411	Directconnect	CERRADO
	412	Directconnect	CERRADO
	1863	MSN Messenger	CERRADO
	6667	IRC	CERRADO
	6881	Bit torrent (1)	CERRADO
	6969	Bit torrent (2)	CERRADO

Tabla 3.11 Resultados de pruebas aplicaciones prohibidas Cliente 1

EQUIPO DE RED	PUERTO	APLICACIÓN	RESULTADO
192.168.2.101 (Cliente 2)	4662	Emule	ABIERTO
	6346	BearShare Gnutella Limwire Xdox	ABIERTO
	5025	Aimster	CERRADO
	1214	Grokiler Kazza	CERRADO
	1234	Hotline	CERRADO
	5498	Hotline	CERRADO
	6343	ToadNode	CERRADO
	411	Directconnect	CERRADO
	412	Directconnect	CERRADO
	1863	MSN Messenger	CERRADO
	6667	IRC	CERRADO
	6881	Bit torrent (1)	CERRADO
	6969	Bit torrent (2)	CERRADO

Tabla 3.12 Resultados de pruebas aplicaciones prohibidas en Cliente 2

En todas las pruebas, el puerto 1863 relacionado con la aplicación Messenger daba como resultado “Puerto cerrado”, lo que generaron dudas, ya que este programa es muy utilizado y difundido. Para evitar que el resultado obtenido no sea consistente a la realidad de la organización y considerando que es un programa que no debería estar permitido su uso en horas de oficina, se va a realizar otro tipo de prueba con el segundo módulo del software para comprobar su utilización o no.

3.2.3 RESULTADOS DE PRUEBAS DE MÓDULO SISTEMA DE DETECCIÓN DE INTRUSOS EN LAS MÁQUINAS DE LA RED

Una de las aplicaciones importantes del *IDS* es analizar tráfico malicioso en relación a firmas digitales de virus o troyanos, ya que conociendo cuál es el comportamiento de éste, se puede discriminarlo según reglas establecidas; por ejemplo se pudo conocer qué personas o usuarios hacían búsquedas de pornografía, contenidos sexuales o información peligrosa, mediante el módulo *IDS*, específicamente modo Sistema de Detección de Intrusos, creando una nueva regla de filtrado tal como se describe a continuación.

En el menú principal se escoge el módulo *IDS*; se selecciona el modo de funcionamiento como Sistema de Detección de Intrusos, se ingresa el número de la interfaz tal como se muestra en la figura 3.5.

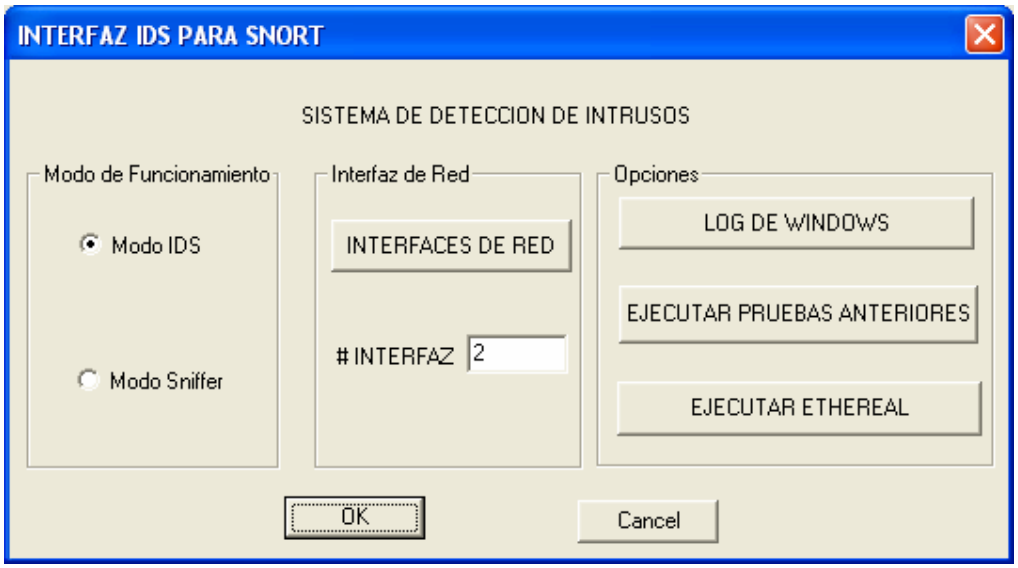

Fig. 3.5 Selección de Modo de Funcionamiento e ingreso de número de interfaz

Escogido el modo de funcionamiento y puesto el número de interfaz se presiona el botón OK, donde se presenta el diálogo de la figura 3.6, en el cual se seleccionan las tres primeras opciones. Estas opciones son: Imprimir los payloads solo con datos no hexadecimal, mostrar información de capa aplicación y mostrar información de la segunda capa, los cuales se ocuparán en el análisis del *log* generado.

Al no tener una regla creada para dicho propósito se presionó sobre el botón crear nueva; haciendo esto se presenta la pantalla de la figura 3.7, donde se escoge el tipo de protocolo que en este caso es *TCP*, por cuanto el contenido de los paquetes a analizar se encapsulan en este tipo de tramas.

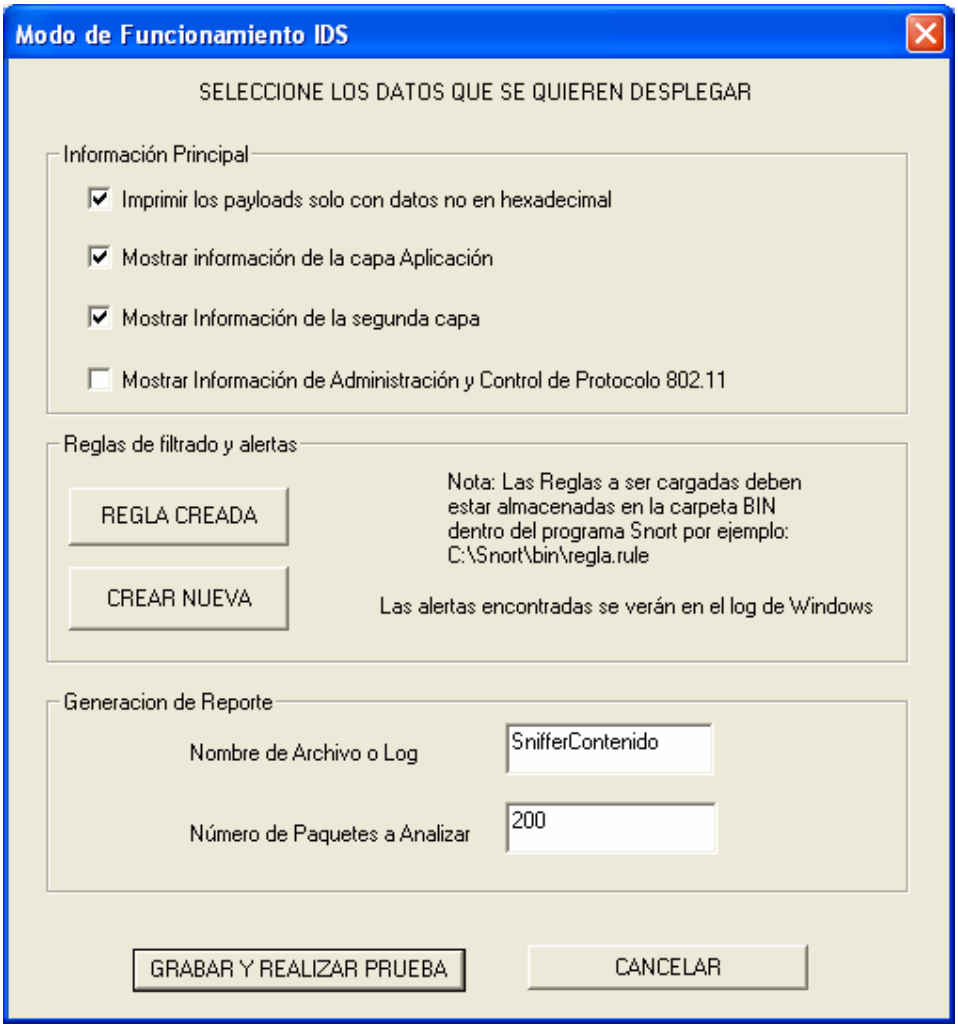

Fig. 3.6 Datos para funcionamiento *IDS*

Además, como se quería analizar todo el tráfico sin ninguna discriminación, las Dirección 1 y 2 se dejaron en blanco o por defecto, al igual que los puertos y se hizo el análisis bidireccionalmente (en ambos sentidos “<>”). Como se quizo detectar el contenido “Hacking”, se puso dicho texto en la parte de entrada de datos “Texto en Payload del Datagrama”; en la parte de entrada de datos “Mensaje de Alerta” se puso el texto de la alarma del visor de sucesos de Windows que para el ejemplo fue: “Paquete con contenido hacking encontrado”, (ver figura 3.7).

Al presionar el botón Grabar Regla, se guarda la regla creada, siendo la extensión de la regla escrita automáticamente, ya que solo se tuvo que poner el

nombre de la misma; se guardó el archivo de la regla dentro de la carpeta C:\Snort\bin\ tal como se muestra en la figura 3.8

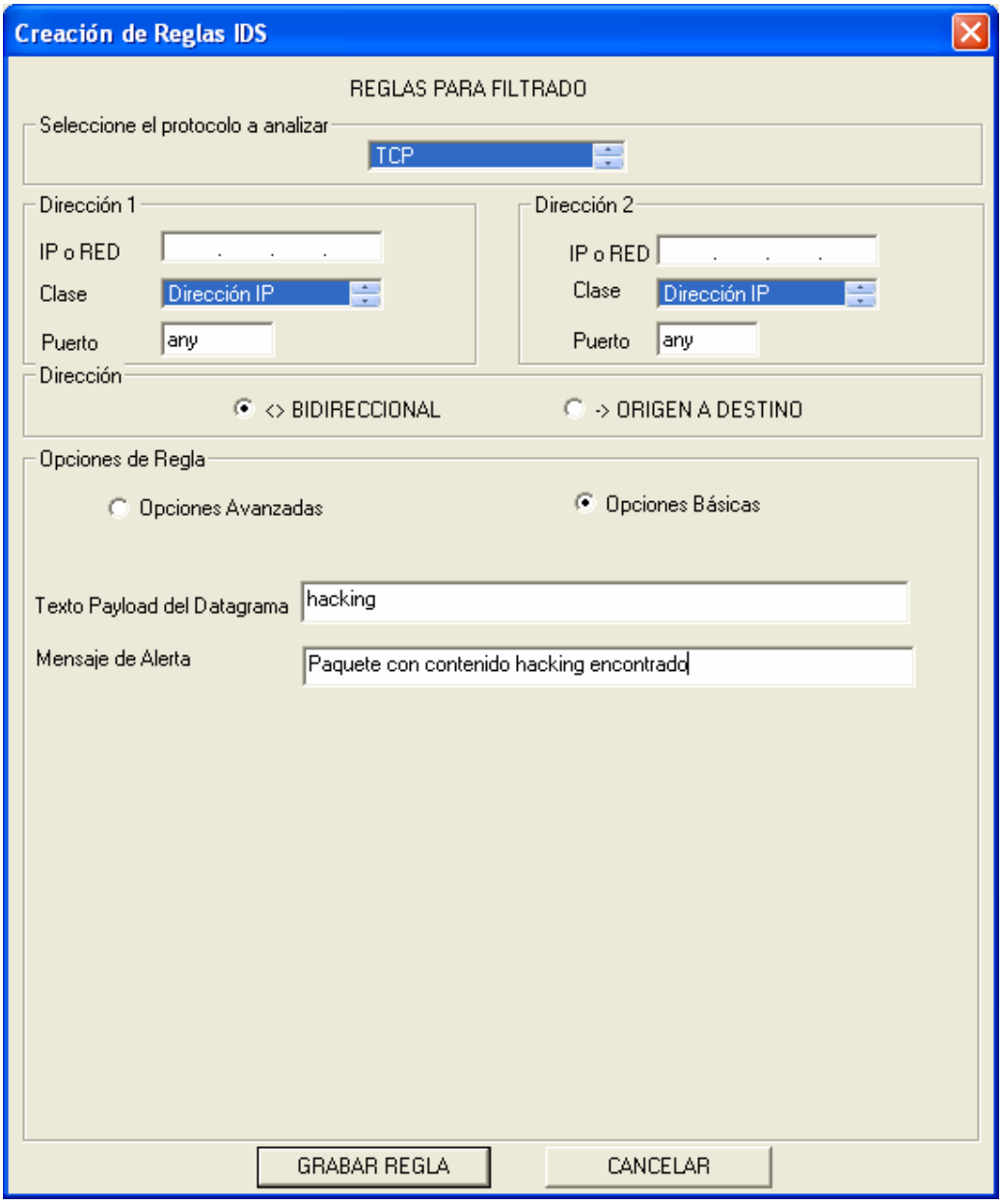

Fig. 3.7 Nueva Regla de Filtrado

El archivo de regla grabado puede ser utilizado posteriormente en otra prueba que se quiera realizar con el Sistema de detección de intrusos, bajo las mismas características.

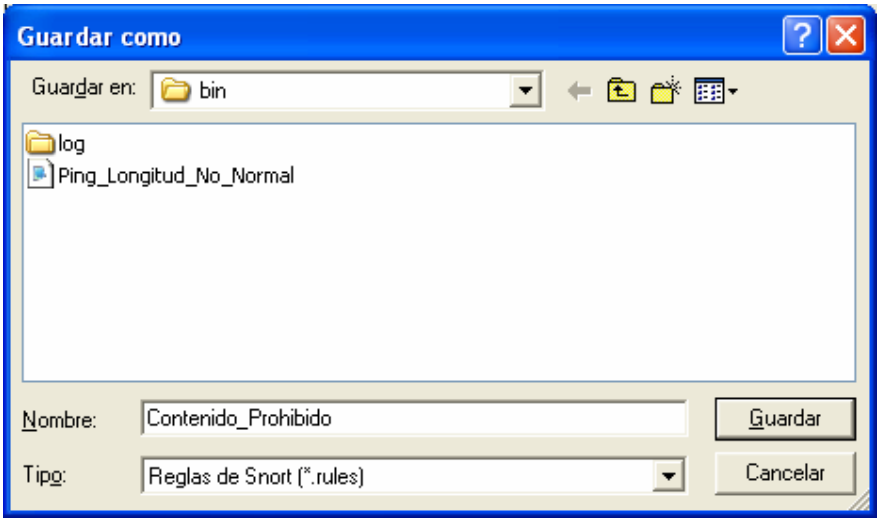

Fig. 3.8 Guardar una regla de *Snort*

Creada la regla, se muestra un cuadro de mensaje indicando su texto, para cerciorarse de que estuvo escrita correctamente, como se observa en la figura 3.9.

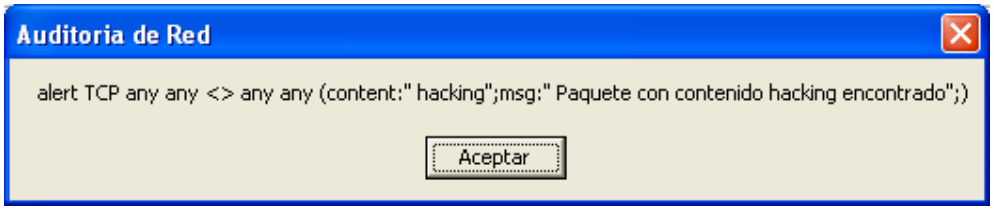

Fig. 3.9 Texto de Regla Creada

Una vez verificado que el texto de la regla está escrito correctamente, se llena la última parte del modo de funcionamiento *IDS* como es el nombre del *Log* a ser generado y el número de paquetes a analizar; al aceptar todos los parámetros escritos y escogidos se crea el archivo de ejecución por lotes para *Snort*, el cual puede ser utilizado posteriormente para otros análisis similares, tal como se indica en la figura 3.10.

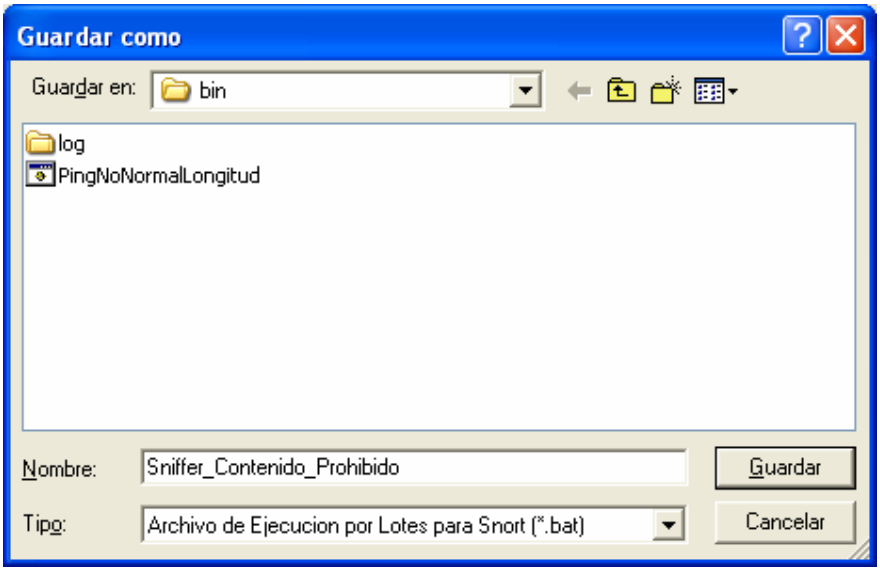

Fig. 3.10 Guardar archivo de ejecución por Lotes para *Snort*

Después se guarda el archivo de ejecución por lotes y si éste no tuvo errores, el programa llama al símbolo del sistema para la ejecución del archivo generado, (ver figura 3.11).

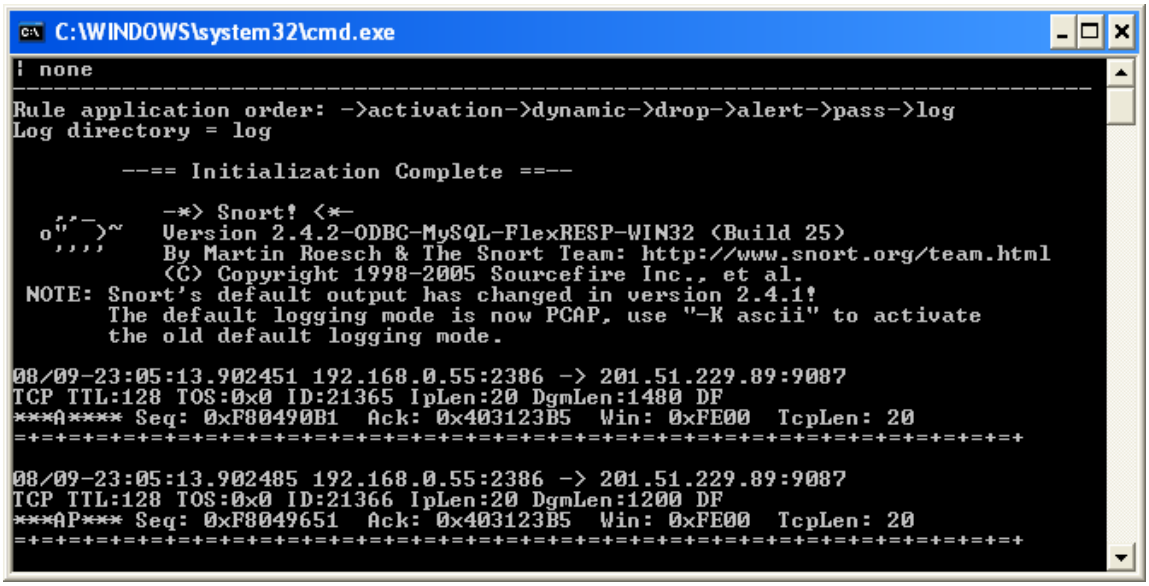

Fig. 3.11 Ejecución de *Snort* en modo DOS

Por último como se generan alertas, éstas se muestran en el Visor de Sucesos de Windows, el cual puede ser abierto desde el menú principal del Sistema de detección de Intrusos; la figura 3.12 muestra una alerta encontrada y registrada en el Visor.

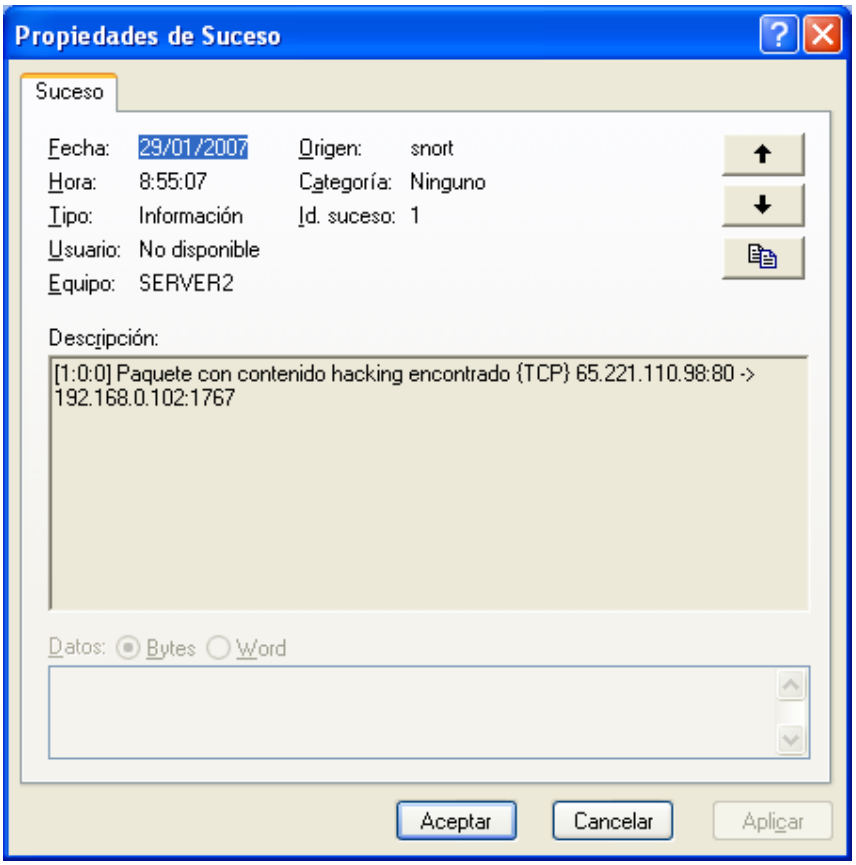

Fig. 3.12 Alarmas generadas en el *Log* de Windows

Como se mencionó antes, para detectar cuando se ocupa el servicio de Messenger se va a utilizar la ingeniería de análisis, para que mediante este proceso se pueda conocer los parámetros que utiliza *Windows Live Messenger* al conectarse al servicio; esto se lo hizo mediante el Sistema de Detección de Intrusos en su funcionamiento como *Sniffer,* analizando todo el tráfico de una máquina mientras quiere conectarse al servicio. Lo descrito se indica a continuación.

En el menú principal se escoge el módulo *IDS*, se selecciona el modo de funcionamiento como *Sniffer* y se ingresa el número de la interfaz que para el caso del ejemplo es 2.

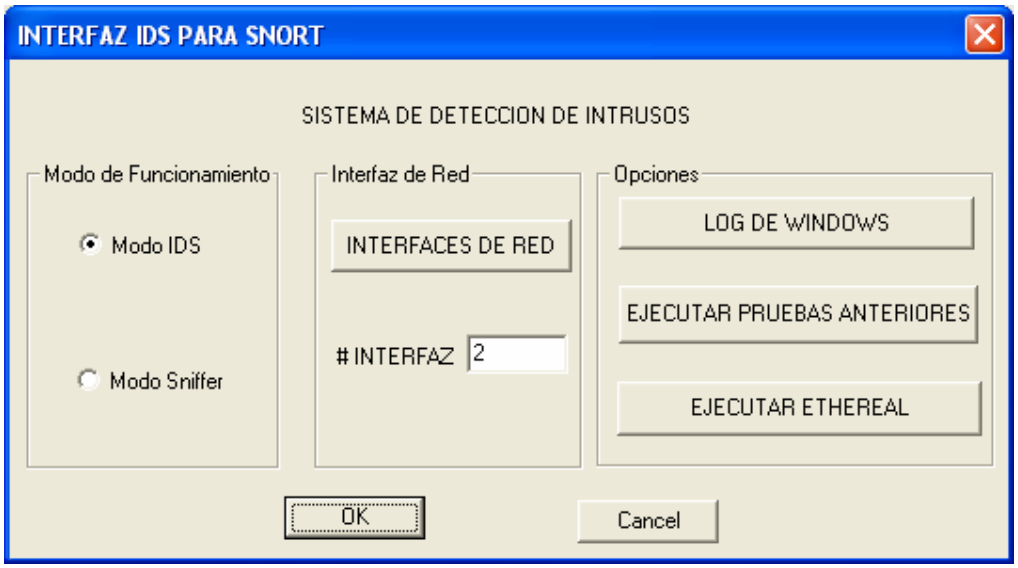

Fig. 3.13 Selección de modo *Sniffer*

Al presionar OK se presenta la pantalla de la figura 3.14, en la cual se tiene que poner el número de paquetes a analizar y el nombre del *Log* a generarse.

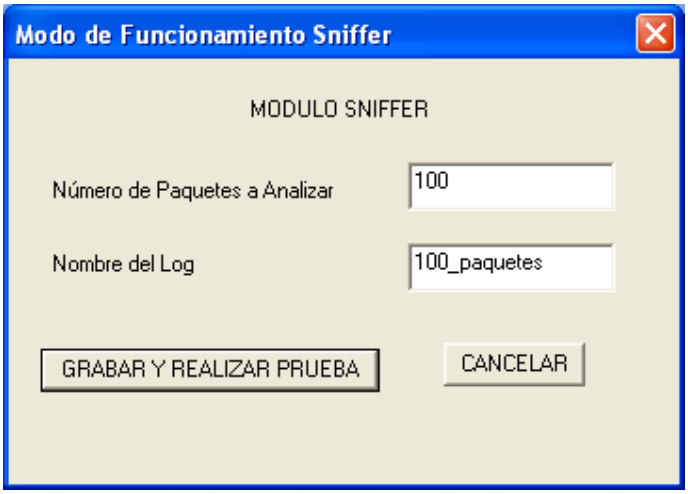

Fig. 3.14 Datos modo *Sniffer*

Luego de escribir el número de paquetes a analizar y el nombre del *Log* a generarse, se tiene que presionar el botón Grabar y Realizar Prueba, en donde el sistema pide al usuario grabar el archivo de ejecución por lotes para una próxima utilización. Lo descrito se muestra en la figura 3.15.

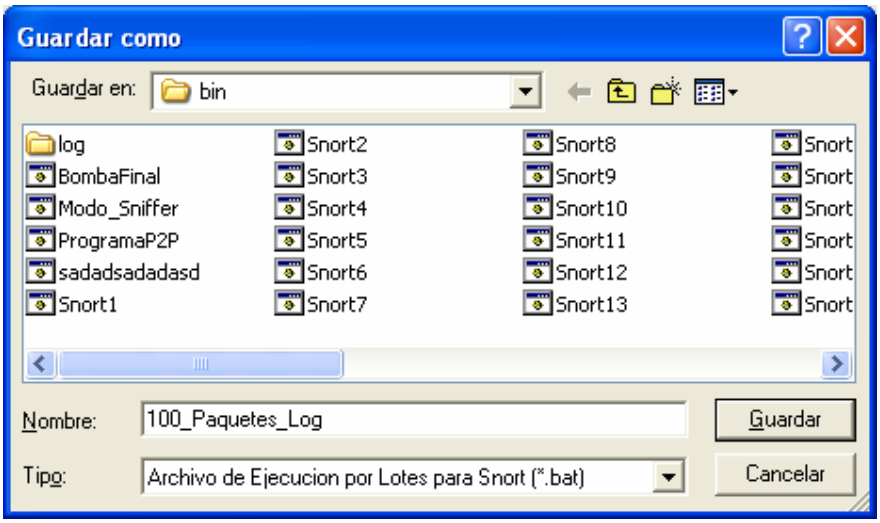

Fig. 3.15 Guardar archivo de ejecución por lotes para *Snort*

Al grabar el archivo se ejecuta el sistema de *DOS* registrando todo el tráfico generado en la red, tal como se lo indica en la figura 3.16.

Para la visualización del *Log* generado lo que se hace es abrir el archivo generado mediante el sistema Ethereal que debe estar instalado previamente.

Luego de abrir el *Log* generado y hacer el análisis de los contenidos de los paquetes, se puede determinar que cuando un equipo desea conectarse al servicio de Messenger envía paquetes *TCP* con contenidos de datos como se muestra en la figura 3.17

En los datos mostrados en la figura 3.17, se puede notar que se tienen textos CVR 238, 0x0c0a winnt 5.1 i386, MSNMSGR 8.0.0812 msmsgs sjopis1@hotmail.com, USR 239, TWNI sjopis1@hotmail.com.

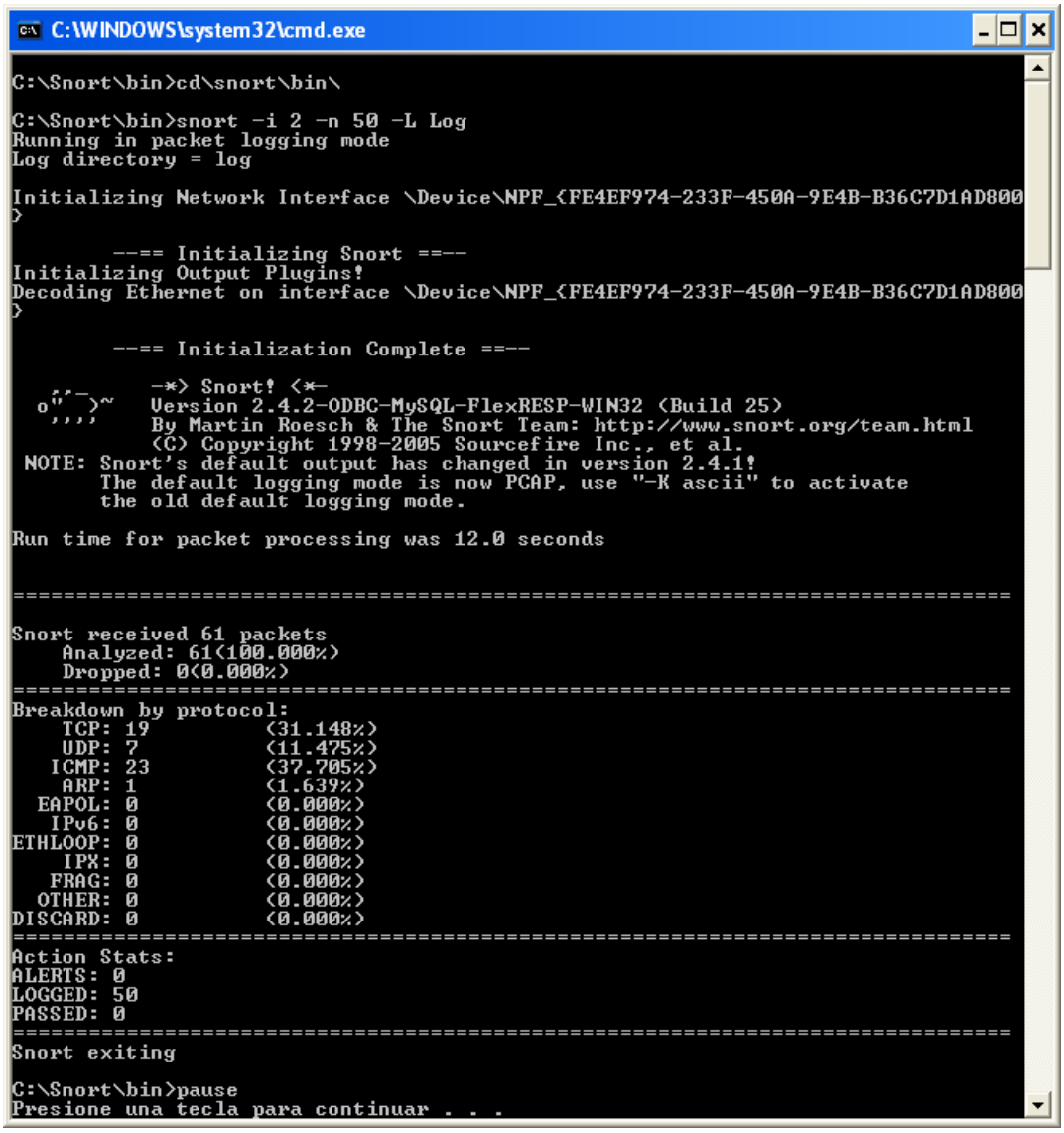

Fig. 3.16 Ejecución del sistema en modo *Sniffer.*

```
0000  00 11 95 34 fb 59 00 08  a1 64 08 6f 08 00 45 00   ...4.Y.. .d.o..E.
0010  00 96 33 66 40 00 80 06  1a 64 c0 a8 00 64 cf 2e   ..3f@... .d...d..
0020  1c 5d 0a db 07 47 33 e5  51 d4 08 2e 54 33 50 18   .]...G3. Q...T3P.
0030  ff e3 46 6b 00 00 43 56  52 20 32 33 38 20 30 78   ..Fk..CV R 238 0x
0040  30 63 30 61 20 77 69 6e  6e 74 20 35 2e 31 20 69   0c0a win nt 5.1 i
0050  33 38 36 20 4d 53 4e 4d  53 47 52 20 38 2e 30 2e   386 MSNM SGR 8.0.
0060  30 38 31 32 20 6d 73 6d  73 67 73 20 73 6a 6f 70   0812 msm sgs sjop
0070  69 73 31 40 68 6f 74 6d  61 69 6c 2e 63 6f 6d 0d   is1@hotm ail.com.
0080  0a 55 53 52 20 32 33 39  20 54 57 4e 20 49 20 73   .USR 239  TWN I s
0090  6a 6f 70 69 73 31 40 68  6f 74 6d 61 69 6c 2e 63   jopis1@h otmail.c
00a0  6f 6d 0d 0a                                       om..
```

Fig. 3.17 Paquete de Inicio de sesión de *Messenger*.

Luego de hacer varias pruebas con diferentes cuentas de usuario se determinó que lo que se mantiene constante son palabras reservadas como:

- CVR
- 0x0c0a winnt 5.1 i386
- MSNMSGR 8.0.0812 msmsgs
- USR
- TWNI

Entonces se llega a la conclusión que se puede crear una regla del Sistema de Detección de intrusos, que pueda discriminar los contenidos que se necesitan para el inicio de sesión de Messenger y así conocer qué usuarios están utilizando el servicio.

Lo realizado se muestra a continuación.

En el menú principal se escoge el módulo *IDS*; se selecciona el modo de funcionamiento como *IDS*, se ingresa el número de la interfaz que en el caso del ejemplo es 2.

Al poner OK se presenta la pantalla de la figura 3.18, en la cual se selecciona Imprimir los Payloads solo con datos no en Hexadecimal; como no se tiene todavía creada la regla, se tiene que presionar sobre el botón Crear nueva, donde se despliega la pantalla de la figura 3.19.

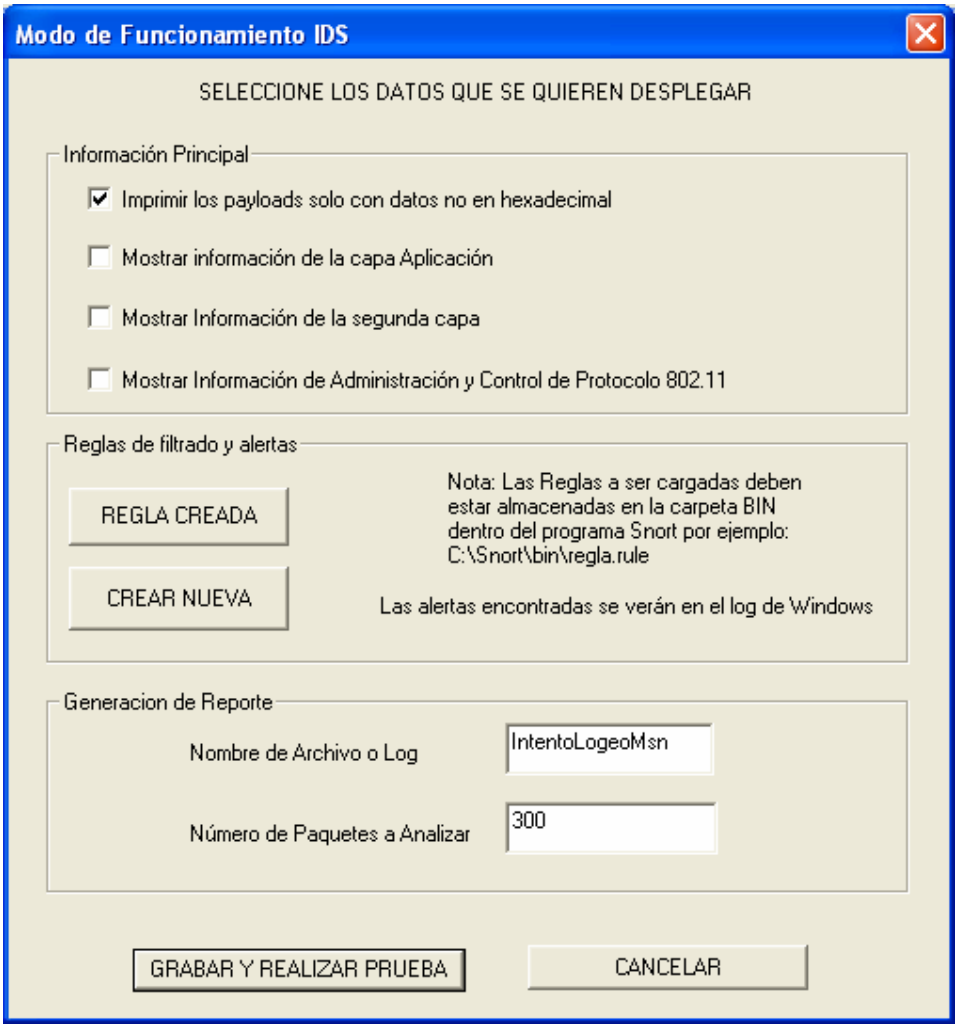

Fig. 3.18 Selección de Información, nombre del *Log* y número de paquetes a analizar.

Al igual que el ejemplo anterior, se quiere analizar todo el tráfico, en ambas direcciones y sin discriminación de aplicaciones; es por eso que se dejan los valores por defecto en las entradas correspondientes a dirección y números de puertos.

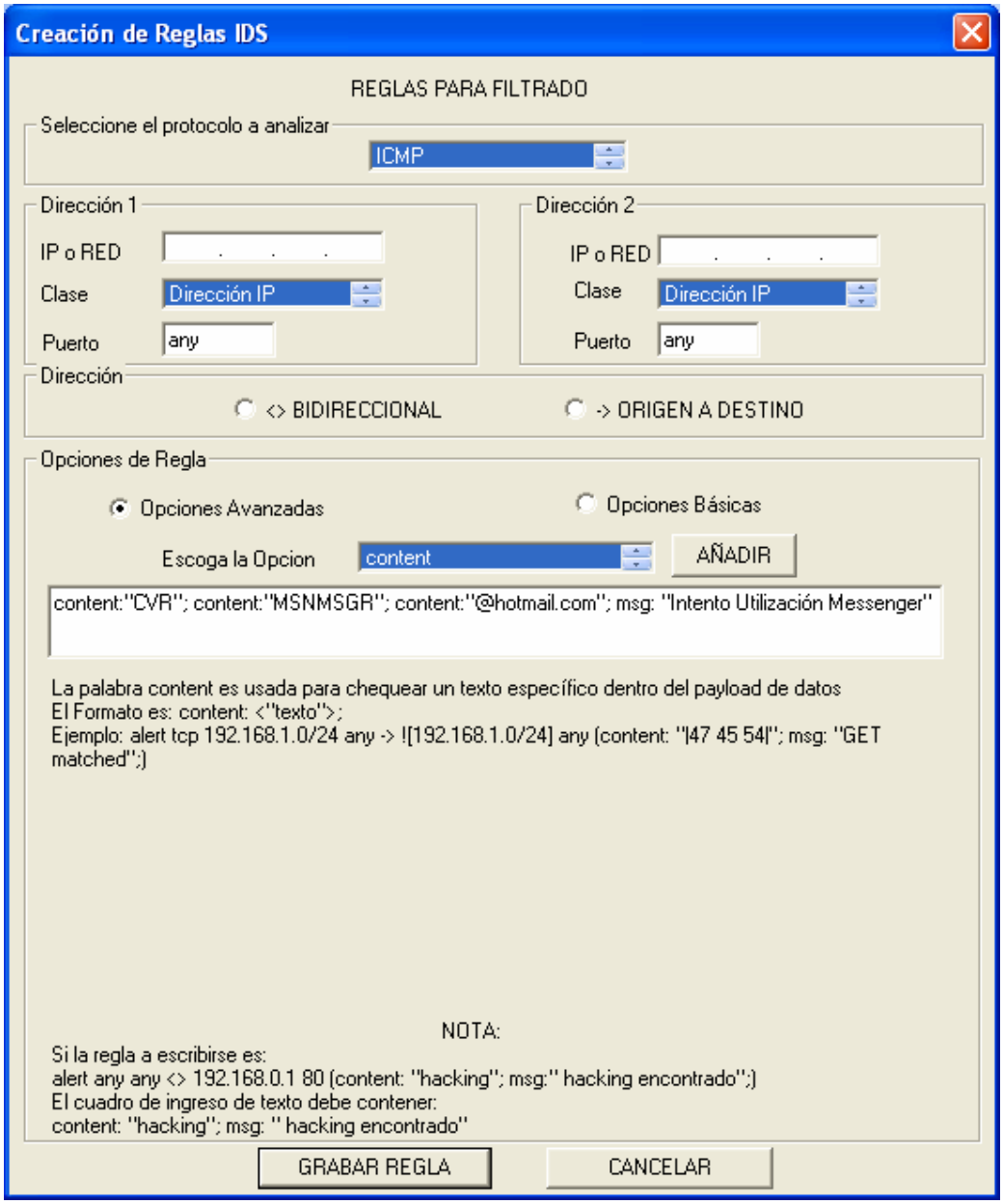

Fig. 3.19 Creación regla nueva para reconocer intentos de inicio de sesión *Windows Live Messenger.*

Para la creación del texto de la regla se selecciona opciones avanzadas, además se escogen las palabras reservadas en la lista y se adiciona el texto necesario para obtener el texto de la regla completa:

content: "CVR"; content: "MSNMSGR"; content:"@hotmail.com"; msg: "Intento Utilización Messenger"

Se escriben tres palabras “content” para dar alarmas cuando se tengan paquetes que contengan las tres palabras y no solo una; esto funciona como un operador lógico AND evitando así los falsos positivos.

Al presionar el botón Grabar Regla, se tiene que grabar la regla generada; en la figura 3.20 se muestra que se la llamó IntentoLogeoMessenger. Al presionar guardar se presenta el texto de la regla creada que se muestra en la figura 3.21.

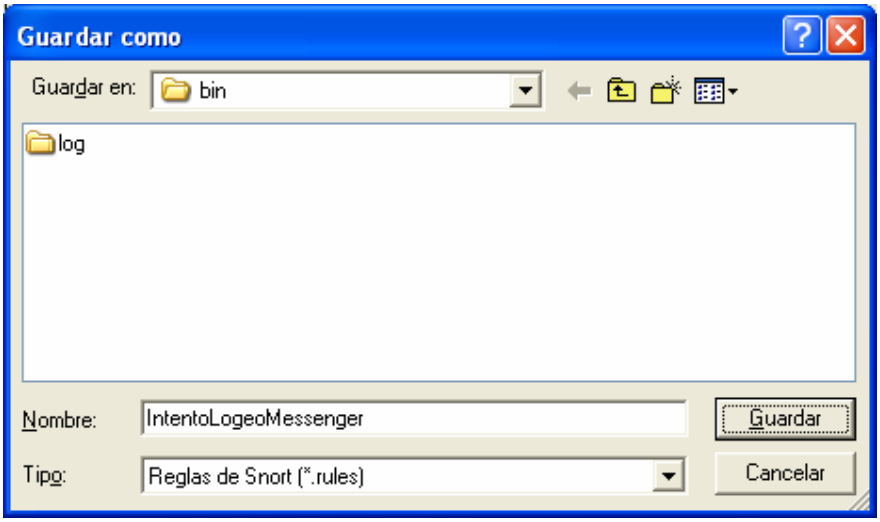

Fig. 3.20 Guardar regla generada.

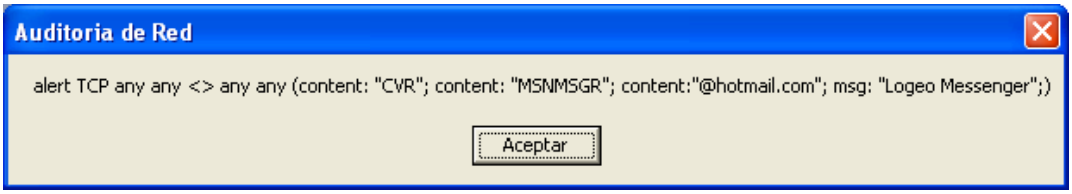

Fig. 3.21 Texto final de la regla generada.

Al presionar Aceptar, se guarda el nombre del archivo de ejecución por lotes de *Snort* tal como muestra la figura 3.22; automáticamente después de guardarse el archivo, éste se ejecuta para comenzar a obtener los datos (ver figura 3.23).

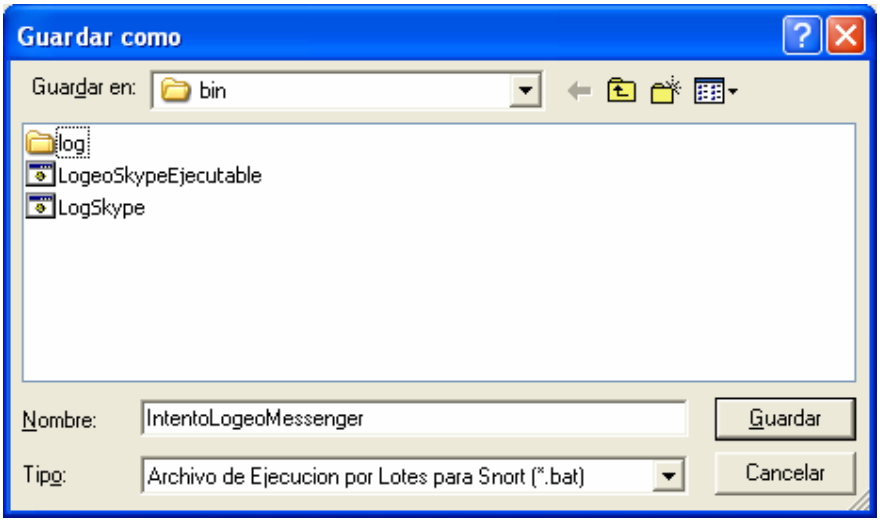

Fig. 3.22 Guardar archivo de ejecución por lotes para *Snort.*

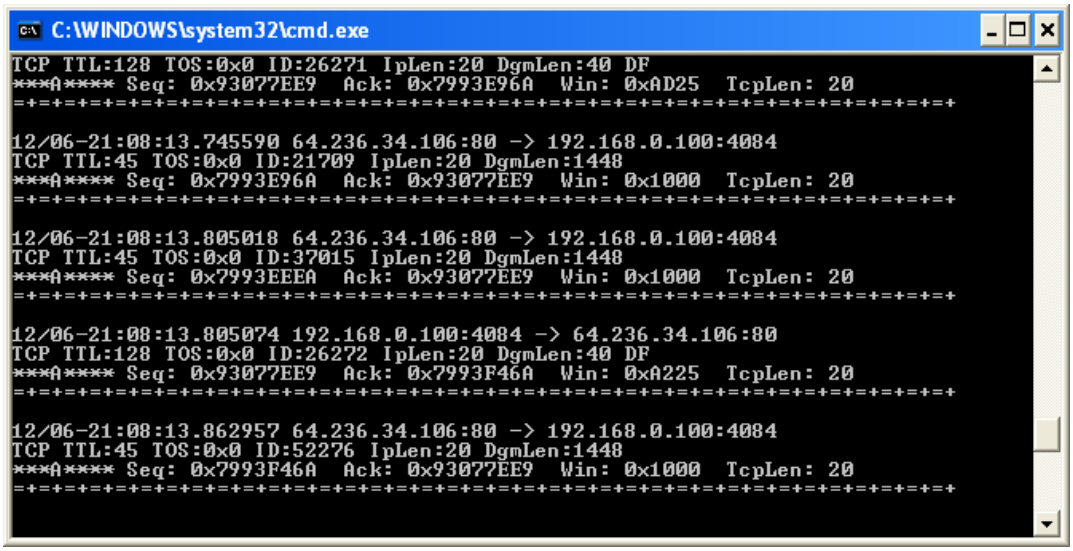

Fig. 3.23 Ejecución del comando creado.

Si es que hay algún intento de logeo al servicio de *Windows Live Messenger*, se tendrán las alarmas correspondientes en el Visor de sucesos de Windows tal como muestra la figura 3.24.

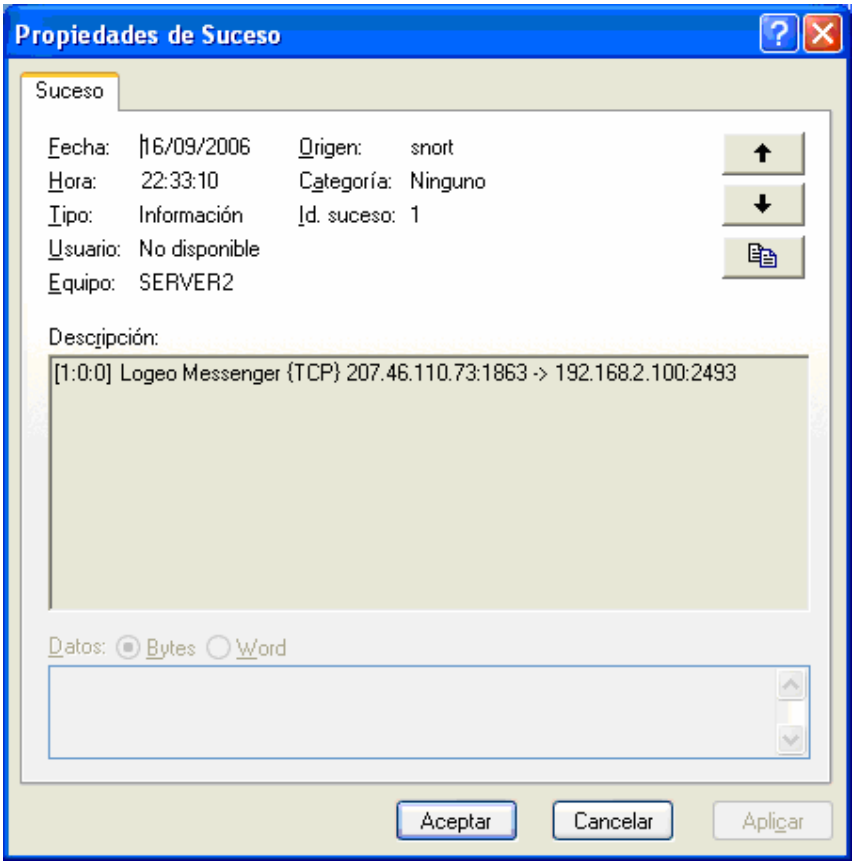

Fig. 3.24 Alarmas generadas en el *Log* de Windows

3.3 ESTUDIO Y ANÁLISIS DE LOS RESULTADOS

Para la definición de las políticas de seguridad de la organización, se tiene que estudiar todos los resultados obtenidos de las pruebas realizadas, auditorías de *Hardware y Software*, así como entrevistar a los usuarios de los terminales de datos y a los encargados de las áreas relacionadas con los sistemas informáticos.

Para esto se va a enlistar todos los puntos importantes que se encontraron y se deducen de las pruebas.

Como se puede observar en las tablas 3.2 y 3.3 relacionadas con la prueba de puertos “bien conocidos”, en los servidores de aplicaciones *Mail* y *Web* respectivamente, se puede notar que se encuentran abiertos los puertos

necesarios para las aplicaciones respectivas, lo cual es lógico para que se tenga un funcionamiento correcto de los servicios configurados en cada servidor.

En el caso del servidor de correo electrónico en donde los puertos 135 (*RPC*) y 139 (*NetBios*) están abiertos, se tiene un problema contra la seguridad de la organización, ya que hacia éstos se pudieran dar intentos de conexión y provocar ataques; es recomendable en el caso de este servidor se cierren dichos puertos para evitar problemas.

De las tablas 3.4 y 3.5 relacionadas a la prueba de puertos “bien conocidos” en los clientes, se nota que los puertos relacionados a las aplicaciones *HTTP* puerto 80 y puerto 443 (*SSL*) se encuentran abiertos, lo que hace que se tengan vulnerabilidades que pueden ser explotadas por un atacante. Como estos puertos en algunos casos deben estar abiertos para que permitan la navegación, se debe normar que se encuentren las actualizaciones al día para evitar cualquier problema.

De las tablas 3.9 y 3.10 relacionadas a los puertos en servidores, se puede notar que no se tienen instaladas aplicaciones como programas P2P, sistemas de Chat, ya que no se detecta que se están utilizando puertos para su funcionamiento, lo que no se da en las máquinas de los clientes, tablas 3.11 y 3.12, en donde se detecta que se encuentran abiertos varios puertos necesarios para el funcionamiento de las aplicaciones citadas anteriormente.

De pruebas de *Sniffing* realizadas en horarios aleatorios, para poder analizar el tráfico circundante en la red, se pudo determinar que en ocasiones, los usuarios o empleados de la organización hacen búsquedas de temas ajenos a los que necesita la organización; esto es un punto importante que se debería tomar en cuenta en la definición de las políticas de seguridad a reglamentarse mas adelante.

Igualmente se puede concluir de los resultados de las tablas 3.11 y 3.12, que los usuarios pueden instalar programas nocivos para la operatibilidad de la

organización; es por eso que se debe reglamentar que ninguno de los usuarios comunes de la organización tenga privilegios de administración en los equipos, evitando así la instalación de cualquier tipo de programas que no sean convenientes y la modificación de configuraciones.

De las pruebas y resultados del análisis de inicio de sesión con *Windows Live Messenger*, se pudo determinar que algunas máquinas de la organización utilizaban dicho servicio, por lo que se debe determinar que este servicio no es apropiado para la organización ya que desperdicia ancho de banda de acceso a Internet y además ocasiona el descuido de los empleados en las labores para las cuales fueron contratados.

Los usuarios ocupan el recurso de Internet de manera no apropiada, ya que pueden acceder a contenidos prohibidos que no favorecen a la organización sino al usuario; es por eso que se debe tener políticas para una prohibición específica de estos casos.

Todos los sistemas operativos y software no se encuentran actualizados con los últimos “parches” de seguridad del fabricante, lo cual produce una grave falla de seguridad ya que se tienen “huecos” que pueden ser utilizados por gente mal intencionada para dejar fuera de operación la organización.

Las personas que dirigen o administran la red de datos de la organización no se encuentran preparadas o bien enteradas del tema de redes de datos, por lo que son susceptibles a la ingeniería social para obtener datos importantes que pueden dañar la seguridad del sistema informático.

3.4 ESTABLECIMIENTO DE NORMAS, PROCEDIMIENTOS Y POLÍTICAS DE SEGURIDAD

Este documento tiene la finalidad de presentar de manera clara y sencilla las políticas de uso y acceso a los recursos informáticos para los usuarios de equipos de computación de la organización.

Políticas para la definición de *password*

Objetivo

El uso del *password* es una parte delicada para la organización ya que si se llega a perder o ser conocido por personas ajenas podría causar problemas de operación; es por eso que el propósito de esta política de seguridad es que ningún usuario no autorizado pueda descubrir el *password* personal que permita el ingreso a la red de datos y al recurso informático.

Definición de la Política

El *password* debe contener por lo menos ocho (8) caracteres alfanuméricos, de los cuales al menos tres (3) serán números. No se debe emplear una contraseña parecida al nombre de usuario utilizado, para acceder al sistema o servicio; estas contraseñas no se deben utilizar para todo el sistema. Al cambiar de contraseña no emplear una similar a la antigua.

No se puede emplear sólo palabras o sólo números, ni utilizar palabras que se puedan encontrar en el diccionario, no se puede ocupar palabras en otros idiomas o escritas en orden inverso, no utilizar información personal (CI, número del seguro social) ni relacionar de alguna forma con el dueño (nombre de la mascota, fechas importantes, lugar de vacaciones, etc).

Cobertura de la Política

Esta política de seguridad debe ser acatada por todas las personas de los departamentos de la organización.

Cumplimiento de la Política

Se considera que el empleado ha cubierto esta política de seguridad, si todo el tiempo se mantiene con un *password* de acceso a la red con 8 caracteres alfanuméricos y es cambiada por lo menos tres (3) veces al año.

Penalidad

En caso de no cumplirse con esta política de seguridad, el usuario no podrá acceder al sistema, hasta que su *password* cumpla con los requerimientos exigidos.

Política para la utilización de correo electrónico

Objetivo

El Correo Electrónico es un servicio que permite mantener comunicados a los usuarios internos entre sí, así como con el mundo exterior, permitiendo que los usuarios desarrollen sus actividades de trabajo.

El uso indebido de este recurso afecta a los usuarios que lo necesitan, razón por la cual se deben indicar las normas de buen uso del servicio.

Definición de la Política

Está permitido

El uso del correo electrónico para actividades de trabajo.

Enviar archivos adjuntos de información con un tamaño máximo de 2 Megabytes (MB); en caso de requerir enviar información de mayor tamaño, se deberá empaquetar con cualquier software especializado para ello, por ejemplo Winzip, que existe en el computador. Si a pesar de la compresión el archivo es mayor a 2 MB se deberá solicitar de manera escrita al área de tecnología y administración de redes, solicitando la ampliación temporal de los datos adjuntos.

Enviar correos masivos dentro de la organización, únicamente desde las cuentas de correo relacionadas con la parte administrativa, como es el departamento de Administración de Sistemas.

No está permitido

Acceder a cuentas de otros usuarios.

Difundir o transmitir información que invada la privacidad de otro funcionario de la organización, así como comunicaciones con contenido de pornografía, actos racistas, informaciones o mensajes de carácter ofensivo a la dignidad de las personas.

Transmitir mensajes que sean similares en su contenido, es decir, correos considerados como cadena.

Difundir o transmitir hacia el exterior de la organización cualquier información que sea de uso exclusivo de los empleados de la compañía.

Dar la dirección de correo electrónico del usuario o de otros colaboradores de la organización, a ciertas páginas del Internet, desde las cuales se envía publicidad.

Abrir mensajes de remitentes desconocidos, porque puede tratarse de algún tipo de virus que provoque daños a la red de la organización.

Aspectos Importantes

Cabe recalcar que se realizará el seguimiento de las políticas indicadas e informará a la Gerencia General sobre su cumplimiento.

Una disposición enviada por el servicio de correo electrónico tiene igual validez que un memorando.

Cobertura de la Política

Esta política de seguridad debe ser acatada por todas las personas de los departamentos de la organización.

Penalidad

Si no se cumple con esta política se informará a la Gerencia General sobre su incumplimiento.

Políticas para la Utilización de Internet

Objetivo

El Internet que se utiliza en la organización es un recurso limitado y costoso que debe estar disponible y de forma oportuna a los usuarios para que desarrollen sus actividades laborables; el uso indebido de este recurso afecta a los usuarios que lo necesitan, razón por la cual se deben priorizar las necesidades y racionalizar el uso.

Definición de la Política

Las actividades laborales (investigaciones, prácticas, búsquedas de material, etc.) que se realizan en el Internet tienen prioridad de uso.

Las actividades laborables primordiales se realizan en las primeras horas del día, es por eso que se prioriza la utilización del Internet en el horario 8:00 hasta las 14:00, exclusivamente para las actividades mencionadas anteriormente.

En caso de requerir utilizar el servicio para consultas o búsqueda de información relacionada a la actividad laboral y que no sea prioritaria, se debe utilizar en horas diferentes a las horas pico.

No es permitido

Navegar en páginas *Web* con contenido pornográfico, de violencia, de drogas, entre otros, por cuanto a más de distraer a los usuarios en temas ajenos a la organización, retrasa el trabajo de éstos y consume recursos de la organización.

Usar programas, tales como el servicio de Chat, salvo en casos de estricta necesidad para el cumplimiento de tareas relacionadas con la organización, debido al gran consumo de ancho de banda y distracción de las actividades normales de trabajo.

Descargar programas, música, videos ya que se desperdicia el ancho de banda de acceso a Internet.

Cobertura de la Política

Esta política de seguridad debe ser acatada por todos los funcionarios de los departamentos de la organización.

Cumplimiento de la Política

Se considera que el empleado ha cubierto esta política de seguridad si todo el tiempo de uso de Internet lo hace dentro del horario establecido, no accede a las páginas mencionadas en definición de la misma y no descarga archivos innecesarios.

Penalidades

En caso de no cumplirse con esta política de seguridad el usuario será amonestado verbalmente.

Políticas para la utilización de las Computadoras, Impresoras, Faxes

Objetivo

Los empleados de la organización son dependientes del hardware, es por eso que si existe alguna falla en los equipos, la eficiencia y operatibilidad de la organización corren peligro; el objetivo de la política es evitar el desperdicio de horas de trabajo en la reconfiguración, reparación de los equipos o servicios que se puedan ver afectados.

Definición de la Política

Los usuarios podrán acceder al computador sólo como usuario sin privilegios de administrador para evitar que puedan instalar cualquier tipo de aplicación ni puedan desconfigurar o configurar otros servicios en el sistema operativo, que alteren a la organización.

Los programas obtenidos desde el Internet directamente por parte de los usuarios, no podrán ser instalados para evitar problemas de virus informáticos.

En caso de requerir algún software nuevo o la modificación de alguno ya instalado en la computadora del usuario, se debe hacer una solicitud a la Unidad encargada.

Queda terminantemente prohibido a los usuarios tratar de hacer *hacking* interno hacia los servidores y computadoras personales de otros usuarios.

Sólo se permiten las impresiones de documentos relacionados con la organización.

La Unidad de Administración de Sistemas realizará la revisión esporádica de la utilización del hardware entregado a los empleados.

Cobertura de la Política

Esta política de seguridad debe ser acatada por todos los funcionarios de los departamentos de la organización.

Cumplimiento de la Política

Se considera que el empleado ha cubierto esta política de seguridad, si en la revisión casual de las computadoras no se encuentra otro software instalado a más de los que constan en la base de datos.

Penalidades

En caso de no cumplirse con esta política de seguridad el usuario será amonestado verbalmente; si es reincidente se informará a la Gerencia General sobre el uso del recurso informático otorgado.

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

1. Existen diversas formas de realizar una auditoría de seguridad, pese a que la solución ideal debe ser la realización de un proyecto de auditoría completo, con varias herramientas en donde se incluyan todos y cada uno de los parámetros y conceptos mencionados. Existen algunas organizaciones, que dependiendo de sus necesidades y del presupuesto, solicitan únicamente realizar algunas de ellas.

2. Pese a la existencia de metodologías concretas para el desarrollo de auditorías de redes informáticas, el factor clave que sin duda hará que los resultados de las mismas respondan a las exigencias de las empresas, se encuentra principalmente en la experiencia y conocimiento de los auditores.

3. Un problema importante en las auditorías de redes, es que el presupuesto asignado para esta área generalmente es bajo, por tal razón no se puede hacer un trabajo completo y de primera.

4. En un sistema de detección de intrusos, se debe evitar tener falsos positivos y en lo posible falsos negativos, ya que con uno sólo de éstos, se tiene una alta posibilidad de que exista algún tipo de problema; esto dará una mala impresión al área administrativa, dificultando la justificación de una inversión más grande y hasta tener problemas de ataques y suplantación de identidades.

5. El lenguaje *UML*, se ha adoptado como estándar por los ingenieros de desarrollo de software, por lo que es de suma importancia su difusión, y debería ser estudiado y fomentado en los estudiantes, cuyas ramas de especialización están relacionadas con la programación y desarrollo de

software, conllevando esto a que las aplicaciones desarrolladas tengan una estructura lógica, un correcto funcionamiento y un alto desempeño.

6. Con los diversos diagramas que se elaboran en *UML*, se puede de manera rápida interpretar, codificar y programar los diversos módulos del sistema a crearse, llevando a una ganancia de tiempo en la programación, optimización de recursos y sobre todo dar soluciones rápidas para los problemas que se pueden presentar.

7. Los sistemas a ser creados, deben cumplir todas las necesidades del usuario, las cuales generalmente son: ser lo más amigables posibles, solicitar la información necesaria para las pruebas, entregar resultados esperados y sobre todas las cosas apegados a la realidad, evitando en lo posible los falsos positivos y falsos negativos.

8. Un sistema debe ser creado de manera que ocupe los mínimos recursos de hardware, sin que esto conlleve a que el rendimiento de la aplicación disminuya y peor aún que genere resultados erróneos.

9. La utilización de librerías estáticas para la compilación de la aplicación desarrollada, ayuda a que ésta pueda ejecutarse sin ningún inconveniente en cualquier plataforma con sistema Operativo Windows 98 o superior, la desventaja de esta opción de compilación es que ejecutable creado ocupa más tamaño.

10. No se tienen las facilidades necesarias al realizar las auditorías de seguridad informática, sea porque los usuarios desconocen del tema o porque no contemplan el beneficio que van a tener en un corto plazo; todo esto hace que el tiempo estimado para la ejecución de las mismas se extienda y que los costos aumenten.

11. Los equipos de interconexión sean éstos *switches, routers*, *access points*, en la mayoría de las organizaciones son utilizados tal y como se los adquirió, es decir con las configuraciones por defecto, dando altas

probabilidades de ataques, ya que dichas configuraciones pueden ser conocidas por cualquier persona extraña a la organización.

12. Los equipos de computación como computadoras personales o *laptops* pueden presentar problemas de contagio de virus, troyanos o gusanos informáticos, ya que por defecto las actualizaciones de los sistemas operativos no están funcionando y se encuentran puertos abiertos innecesariamente, lo que implica tener un nivel de defensa bajo, si un equipo presenta estos problemas es conectado a una red hostil como Internet, no podrá protegerse como debería hacerlo.

13. Los departamentos administrativos solo observan la operatibilidad de los sistemas, esto quiere decir que sólo interpretan el funcionamiento con un correcto desempeño; aseguran que si está funcionando el sistema, no hay ni existirán problemas. Pero la realidad es otra, si un sistema operativo o una red de datos tiene falencias en la seguridad, pueden existir personas que lo estén monitoreando pasivamente, tomando datos hasta lograr suficiente información y comenzar a atacar, causando problemas graves como fraudes o suplantaciones de identidad.

14. En auditorías de seguridad, se pueden considerar dos tipos de aspectos que se manipulan, uno llamado controlado, relacionado con aplicaciones informáticas y malas configuraciones de equipos, que pueden ser manejados mediante las herramientas informáticas y conocimiento de causa. El otro tipo no controlado relacionado con aspectos humanos, en donde debe concientizar a personas de los problemas que existen como el de la ingeniería social explicado anteriormente,

15. Siendo Internet el lugar donde se encuentra la mayor cantidad de información y además como éste es un recurso limitado, se debe tener controlado su acceso y su utilización, ya que según las estadísticas y pruebas realizadas en los equipos, se concluye y se observa que el Internet es ocupado para aplicaciones innecesarias como son: descarga de archivos de música, videos y hasta en búsquedas de contenido

prohibido, lo que conlleva a poder descargas archivos o navegar por páginas peligrosas que no favorecen a la seguridad informática de la organización.

16. Se entiende que un *IDS,* es un complemento de un *firewall,* es decir no implica que un Sistema de detección de intrusos sea suficiente para brindar un excelente nivel de seguridad a una organización, pero si ayuda a mejorarlo.

4.2 RECOMENDACIONES

Se recomienda:

1. La utilización de modelos de programación*,* para la creación o desarrollo de software, ya que cumpliendo con un proceso de elaboración, se puede optimizar tiempo de diseño, código de programación, y búsqueda de errores; además de que son mucho más adaptables a cambios que se pueden presentar en un futuro.

2. Que para generar aplicaciones informáticas de cualquier fin, se lo haga de una manera bastante amigable, no tan pesada en relación a utilización de recursos del sistema y sea lo bastante documentada, para que pueda ocuparse de una manera sencilla.

3. Capacitar al área encargada del sistema informático y redes de datos, en los campos de seguridad informática para que comprendiendo todos los posibles problemas, sean éstos de desactualizaciones de equipos, de virus, troyanos, gusanos, de instalación de software pirata, etc, puedan manejarlo y actuar de manera rápida para poder solucionarlo de una manera óptima.

4. Hablar con los responsables de la parte administrativa de las organizaciones, acerca de los problemas informáticos y de las posibles

consecuencias que se puede tener a corto y largo plazo, resaltando que la mejor defensa es la prevención.

5. Las políticas de seguridad definidas, no pueden quedarse solamente escritas en papel, es por eso que se recomienda que éstas deban ser difundidas en toda la organización sea por medio de *e-mails*, o de una exposición global, en la cual se puedan realizar preguntas que aclaren todas las dudas e inquietudes que se puedan tener. Siendo un tema importante, si no se lo cumple se tendrían implicaciones serias.

6. Actualizar el sistema operativo y aplicaciones importantes de las computadoras, por medio de los "parches" de seguridad que publican los fabricantes, principalmente en aquellas que se encuentren conectadas a Internet; ya que éstos equipos son mucho más vulnerables que otros que no trabajen en una red insegura como la red Internet.

7. En organizaciones grandes tener un Servidor de actualizaciones como *WSUS,* con esto se descargará toda la información necesaria de los "parches" y actualizaciones una sola vez por cada nueva actualización que el fabricante publique. Esto implica que la información repetitiva (actualizaciones y "parches" para equipos con sistema operativo y programas iguales) se descargue desde el servidor y no ocupando Internet, ahorrando considerablemente el ancho de banda de conexión.

8. Que los equipos utilizados para interconexión no sean instalados con las configuraciones por defecto, y que en cambio se hagan las modificaciones necesarias de los parámetros sean éstos *password*, políticas, direcciones, etc.

9. La utilización combinada del *firewall* y el sistema de detección de intrusos, ya que son un complemento que favorece a la seguridad de las organizaciones.

10. La instalación y utilización de antivirus y *firewalls* de casas comerciales conocidas, ya que si se utiliza software no conocido y difundido, en vez de favorecer a la seguridad de la organización se estaría perjudicando ya que podrían tratarse de aplicaciones maliciosas o modificadas que tengan escondidos programas que atenten contra la seguridad de la red.

11. Si se requiere bajar programas fundamentales y necesarios para la organización, se recomienda que se lo haga desde páginas *Web* o *urls* seguros; que se compruebe la integridad del archivo descargado mediante la utilización de funciones *hash*, todo esto para evitar instalaciones de archivos modificados que son perjudiciales.

12. Para la parte inalámbrica de la red de datos, configurar claves de encripción o seguridades inalámbricas (como son W*EP, WPA, WPA2*) y además se configure una lista de acceso controlando las direcciones *MAC* de las tarjetas, permitiendo así sólo el ingreso a equipos registrados y evitar una posible intrusión de un equipo que logró conseguir la clave.

13. Que las revisiones en los equipos para observar el cumplimiento o no de las políticas de seguridad sean de dos maneras, programadas y no programadas; las programadas deben definirse con un número de ocasiones mensuales y las no programadas, de una manera esporádica cuando el área lo crea conveniente. Con esto se obliga al usuario a permanecer sin ningún tipo de alteraciones en el software y hardware asignado.

14. Tener una base de datos de los programas y de las configuraciones, que pueda ser accedida vía *Web*, para informar a los usuarios del inventario de su computador y evitar posibles negaciones de responsabilidad en caso de cambios de configuración e instalación de programas.

REFERENCIAS BIBLIOGRÁFICAS

[1] Redes de Computadoras, Andrew S. Tanenbaum

[2] Comunicaciones y Redes de Computadoras, William Stallings

[3] http://isecom.securenetltd.com/OSSTMM.es.2.1.pdf

[4] Repaso de redes de Datos, Edwin Montoya

[5] http://www.monografias.com/trabajos13/trmetod/trmetod.shtml

[6] http://www.iworld.com.mx/iw_specialreport_read.asp?iwid=4184&back=1

[7] http://www.cert.org/stats/

[8] Intrusion Detection with Snort, Bruce Perens' Open Source Series

[9] Redes Trampa y Detección de Intrusos en entornos cifrados, Jose M. Duart Lorente

[10] Utilización de UML en Ingeniería de Software con Objetos y Componentes, Booch, Jacobson, Rumbaugh, Adisson Wesley

[11] http://www.creangel.com/uml/intro.php

[12] http://www.ilustrados.com/publicaciones/EpypFyupkEoXKQKYcV.php

[13] http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci511650,00.html

[14] Computerworld [nº:1012] [pag:32] [16/04/2004]

[15] http://www.auditoriasistemas.com/politicas_de_seguridad.htm

ANEXOS

ANEXO 1

INSTALACIÓN Y CONFIGURACIÓN DEL SOFTWARE DE AUDITORÍA

INSTALACIÓN Y CONFIGURACIÓN DEL SOFTWARE DE AUDITORÍA DE RED

Como se mencionó en la introducción de este proyecto de titulación, el software consta de dos módulos, uno es el escáner de puertos y el segundo la interfaz gráfica para la herramienta *SNORT*; para un correcto funcionamiento del segundo módulo, se tiene que instalar software *freeware* como se indica a continuación.

WinPcap 3.1

WinPcap es la herramienta estándar de la industria para acceder a la conexión entre capas de red en entornos Windows. Esta herramienta permite a las aplicaciones capturar y transmitir los paquetes de red, puenteando la pila de protocolos; tiene características útiles adicionales, incluyendo filtrado de paquetes a nivel del núcleo, un motor de generación de estadísticas de red y soporte para captura de paquetes remotos.

WinPcap consiste de un controlador, que extiende el sistema operativo para proveer acceso de red a bajo nivel, y una librería que es utilizada para acceder fácilmente a las capas de red de bajo nivel.

Gracias a este conjunto de características, WinPcap es el motor de captura de paquetes y filtrado de muchas de las herramientas de red comerciales y de código abierto, incluyendo analizadores de protocolos, monitores de red, sistemas de detección de intrusos, *sniffers*, generadores de tráfico y *network testers*. Algunas de estas herramientas *como Ethereal, Nmap, Snort y Stop* son conocidas y usadas comúnmente.

Instalación de WinPcap

Al ejecutar el archivo ejecutable se tiene la pantalla de la figura A1-1, donde se explica el tipo de herramienta; se tiene que seleccionar *next* para proceder con la instalación.

Fig. A1-1 Instalación de WinPcap

Luego de presionar la tecla mencionada, se tiene acceso al contrato de licencia de uso del programa (Figura A1-2); después de leer todos los términos y condiciones de la herramienta se tiene que seleccionar *I Agree*, aceptándolos.

Al aceptar la licencia se comienza con la instalación propiamente dicha, en la que se graban en el sistema todos los archivos necesarios para el funcionamiento del software.

Cuando se copian todos los archivos se muestra un diálogo, indicando que la instalación fue concluida satisfactoriamente, tal como se muestra en las figuras A1-3 y A1-4 respectivamente.

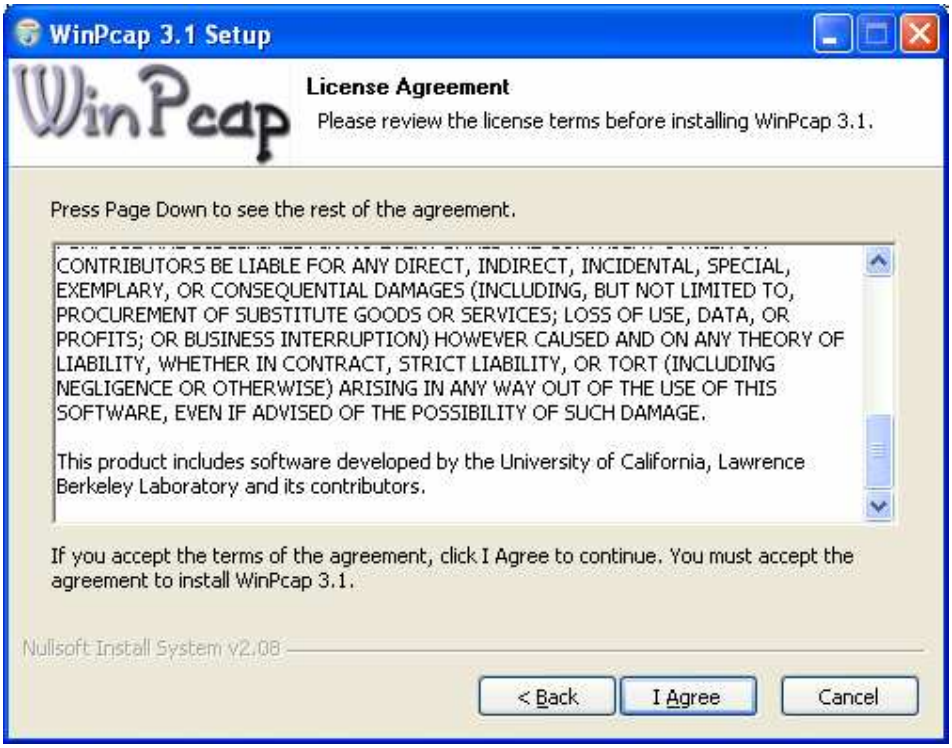

Fig. A1-2 Contrato de Licencia WinPcap

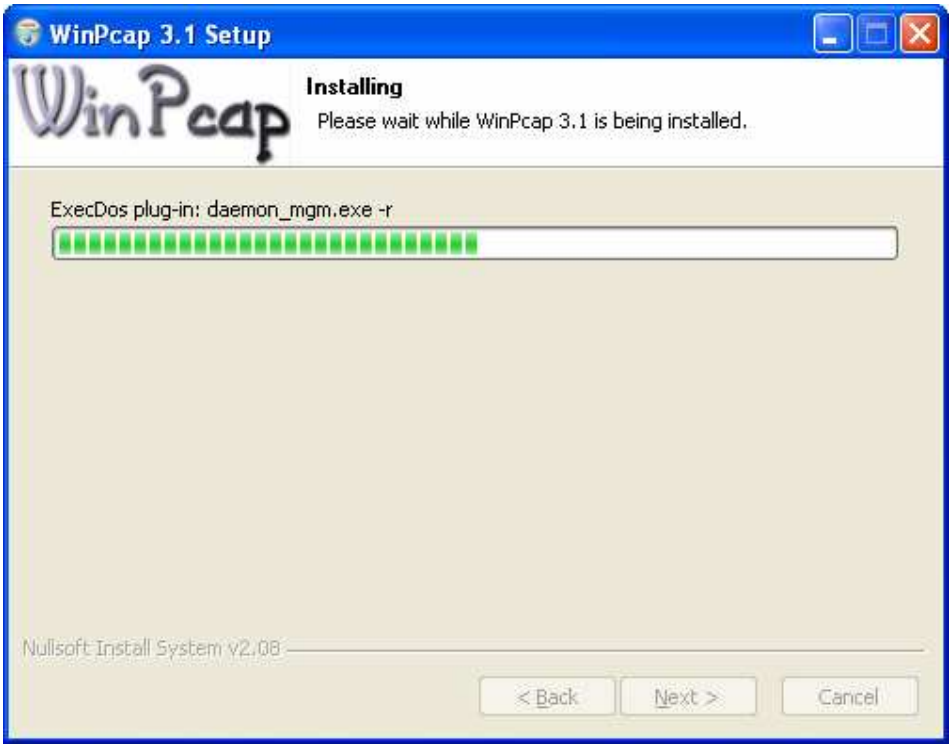

Fig. A1-3 Instalación de la herramienta

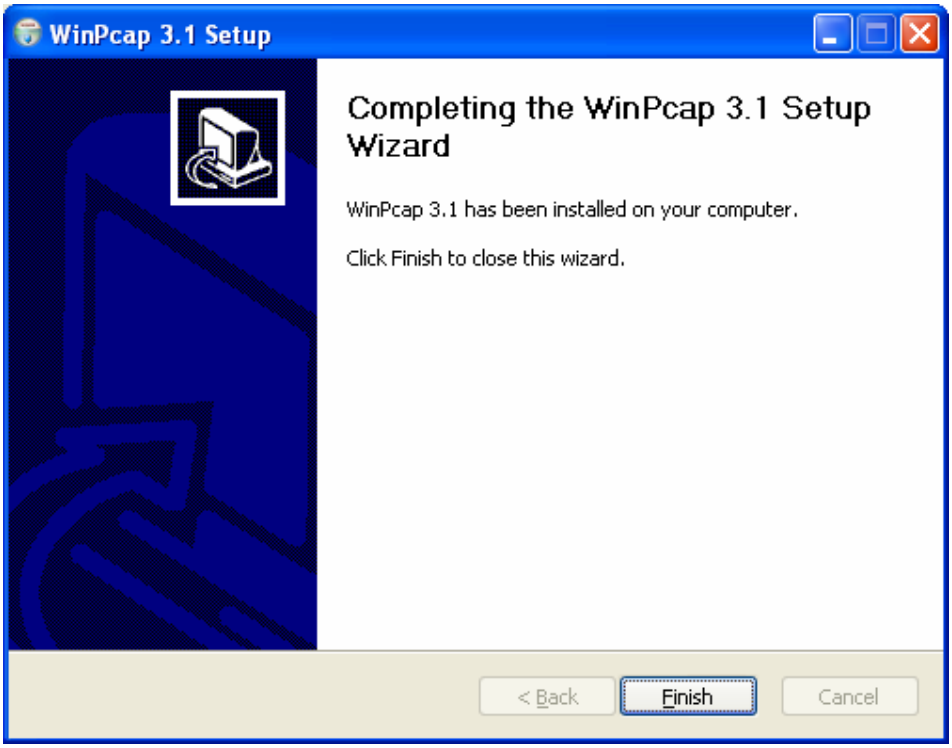

Fig. A1-4 Finalización exitosa de WinPcap

Snort 2.4 para DOS

Luego de instalar el software WinPcap, es posible instalar el Sistema de Detección de Intrusos *Snort* ejecutando el archivo snort_242_Installer.exe que se encuentra en el CD de instalación. Al ejecutar el archivo, se muestra la pantalla de la figura A1-5 donde indica el Contrato de Licencia, el cual debe aceptarse para proceder con la instalación.

Como no se planificó ocupar una base de datos para almacenar los logs generados por el sistema, se escoge la primera opción “I do not to log to a database or I am planning to log to one of the databases listed above”, tal como se muestra en la figura A1-6

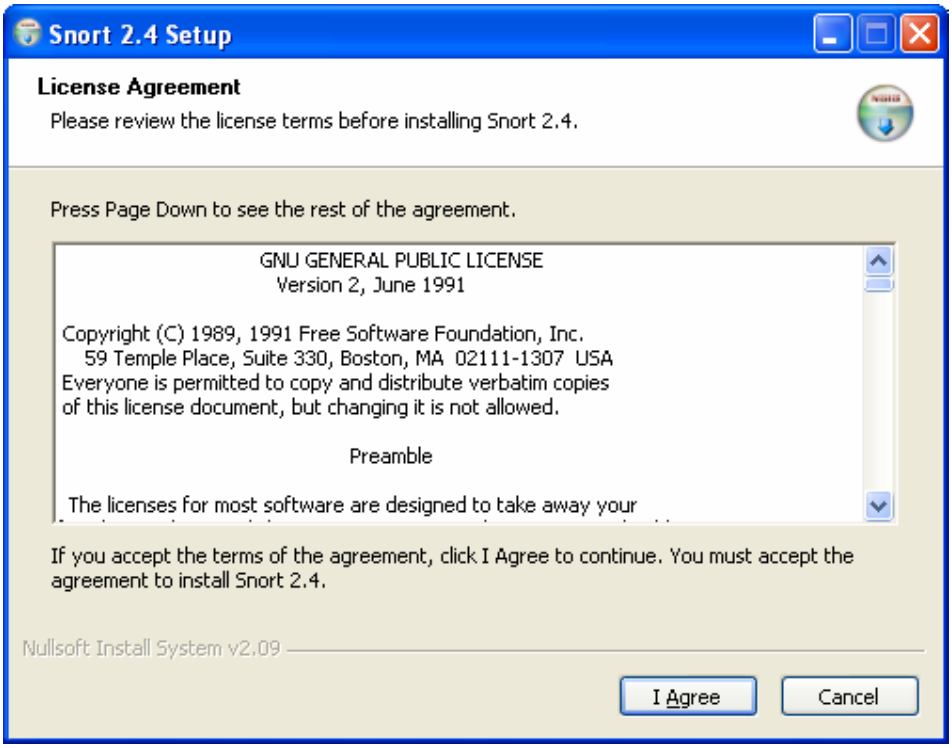

Fig. A1-5 Contrato de Licencia de Snort

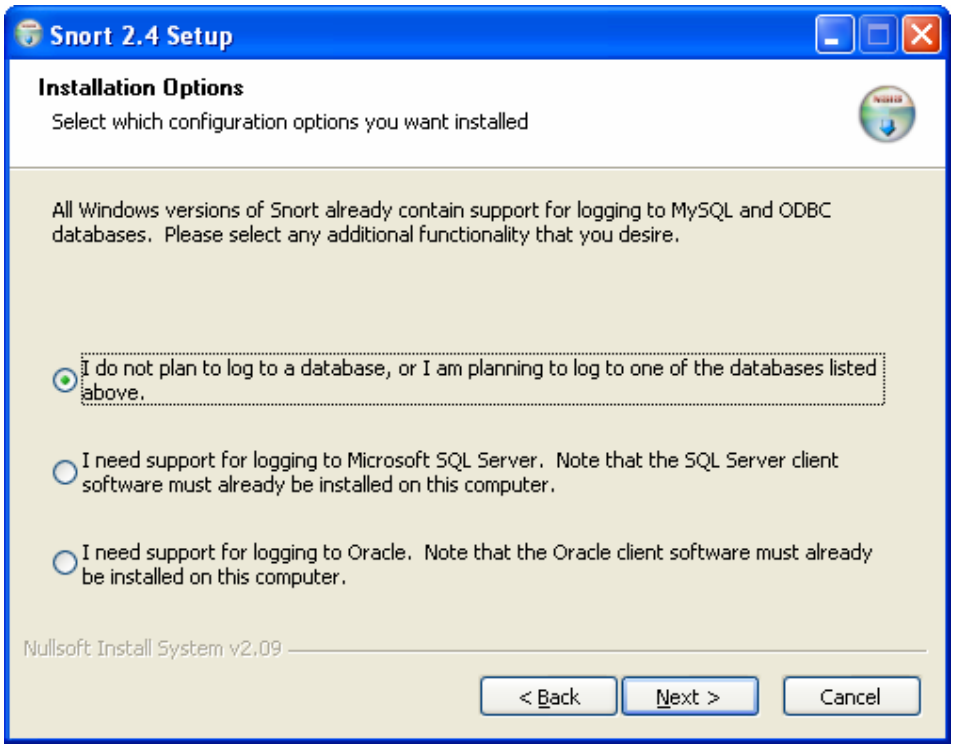

Fig. A1-6 Opciones de Instalación de Snort

Snort tiene la opción de grabar archivos correspondientes a documentación y esquemas al sistema; se puede seleccionar éstos para revisarlos posteriormente. Cabe recalcar que esto no es una obligación, pero se recomienda ocuparlos si es que se llegasen a tener dudas de uso en futuras pruebas.

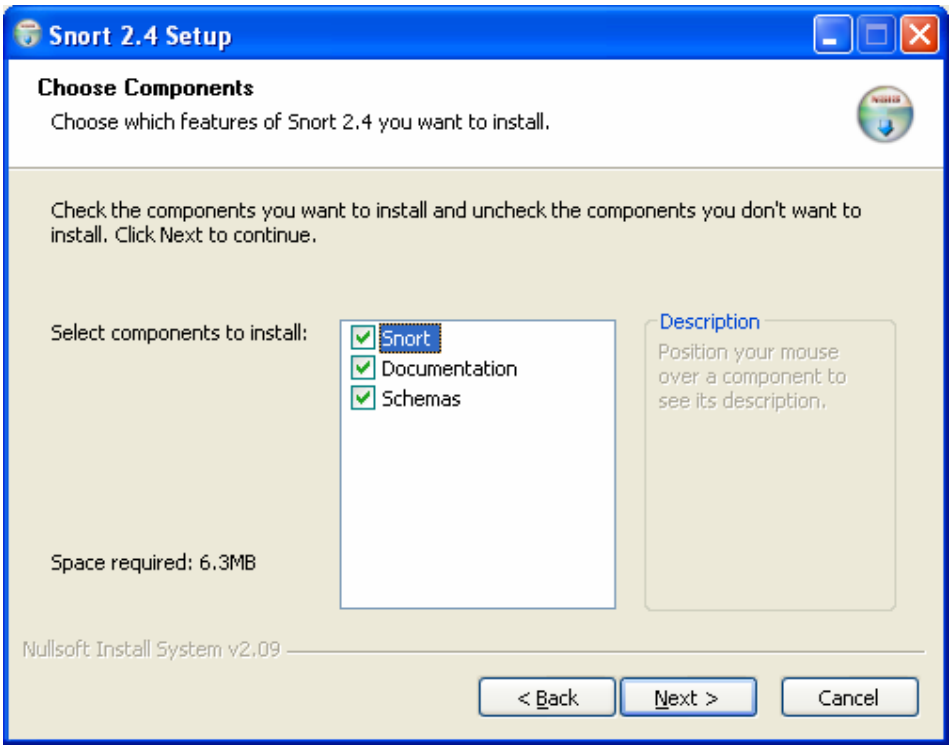

Fig. A1-7 Componentes de *SNORT*

A continuación se necesita poner la ubicación del directorio de instalación, el cual será C:\Snort como se indica en la figura A1-8, en el cual se copiarán los archivos necesarios luego de presionar *next* (ver figura A1-9).

Luego de tener instalado estas dos aplicaciones, se necesita reemplazar el archivo snort.conf incluido en el cd dentro de la ubicación C:\Snort\etc\, este archivo se encuentra modificado de tal manera que se pueda utilizar *Snort* según lo previsto.

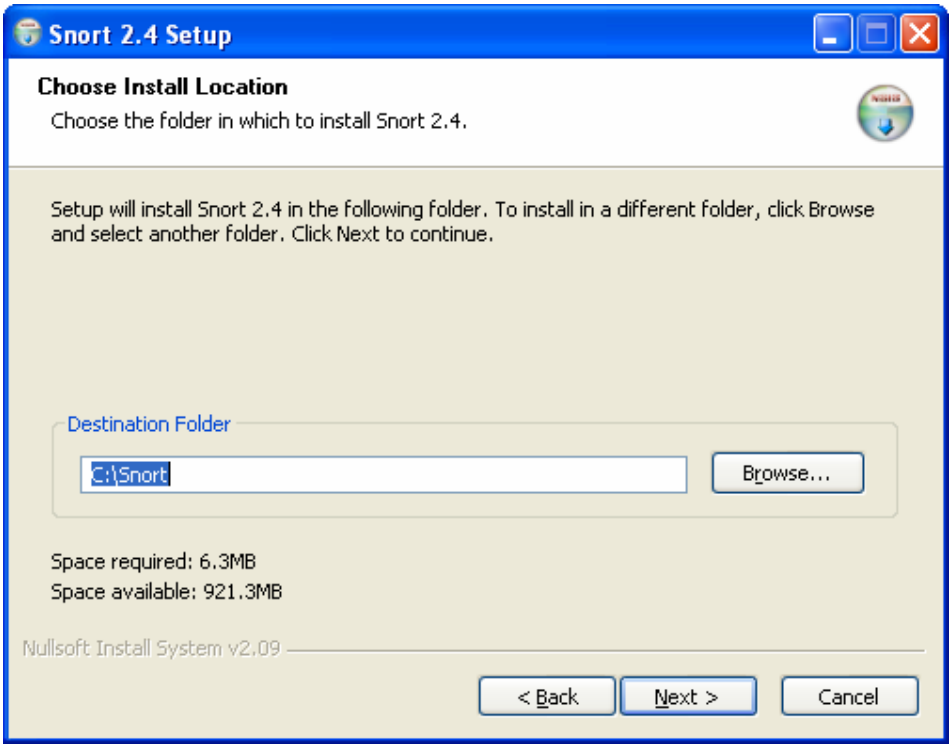

Fig. A1-8 Ubicación de la Instalación de Snort

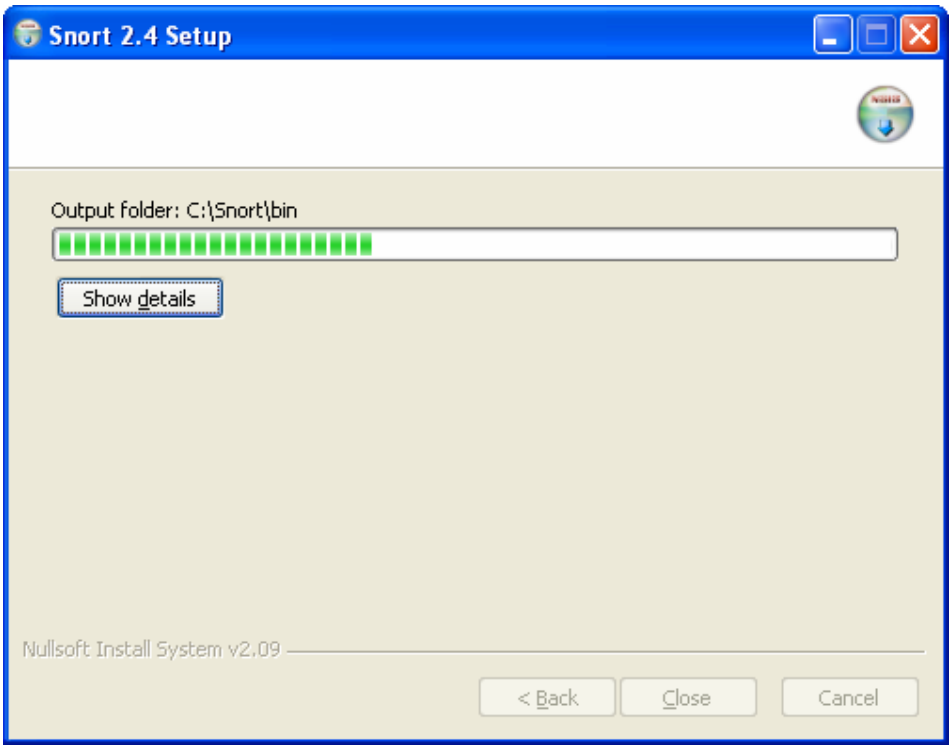

Fig. A1-9 Instalación de la Herramienta

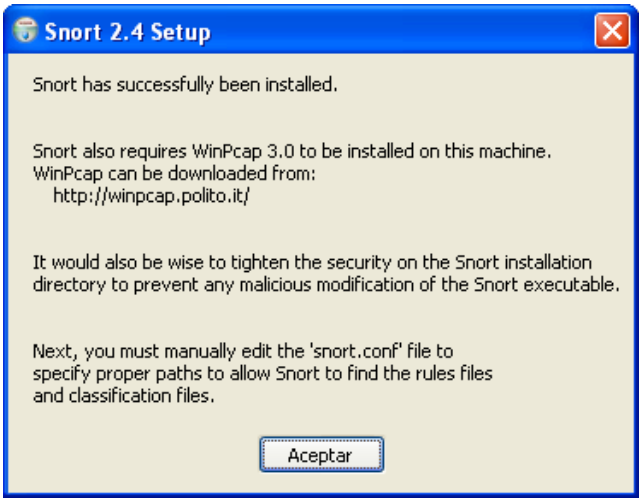

Fig. A1-10 Finalización exitosa de Snort

Ethereal

Para poder visualizar los *Logs* generados por el Sistema de detección de Intrusos *Snort,* se tiene que instalar un programa que lea los archivos generados; el programa escogido es el Ethereal que por ser bastante difundido y *FREEWARE* se adapta a los requerimientos de la herramienta.

Para instalar el programa, lo que se hace es dar doble clic en el instalador ethereal-setup-0.99.0.exe que se encuentra en el CD, con esto se va a ejecutar el asistente tal como muestra la figura A1-11.

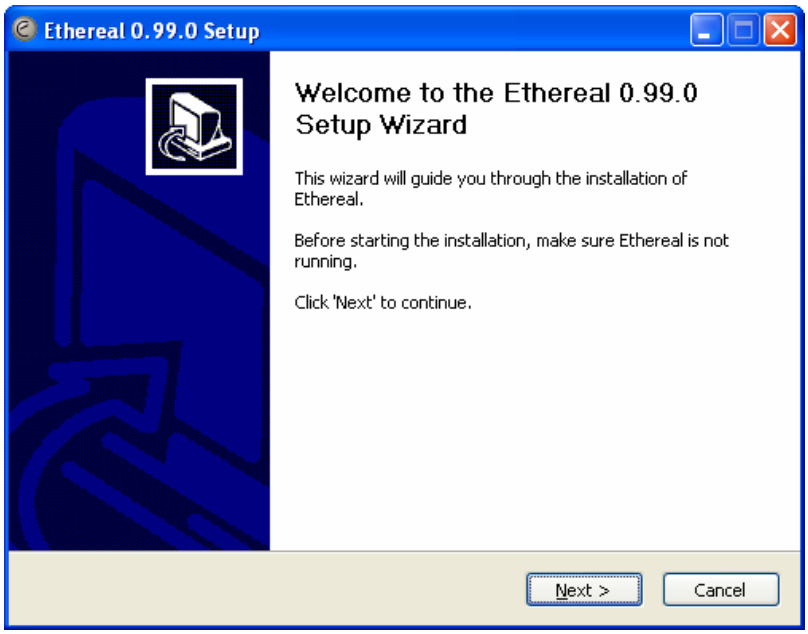

Fig. A1-11 Instalación de Ethereal 0.99.0

Al presionar “next” se despliega el Acuerdo de Licencia, el cual deberá aceptarse para continuar con la instalación (ver figura A1-12).

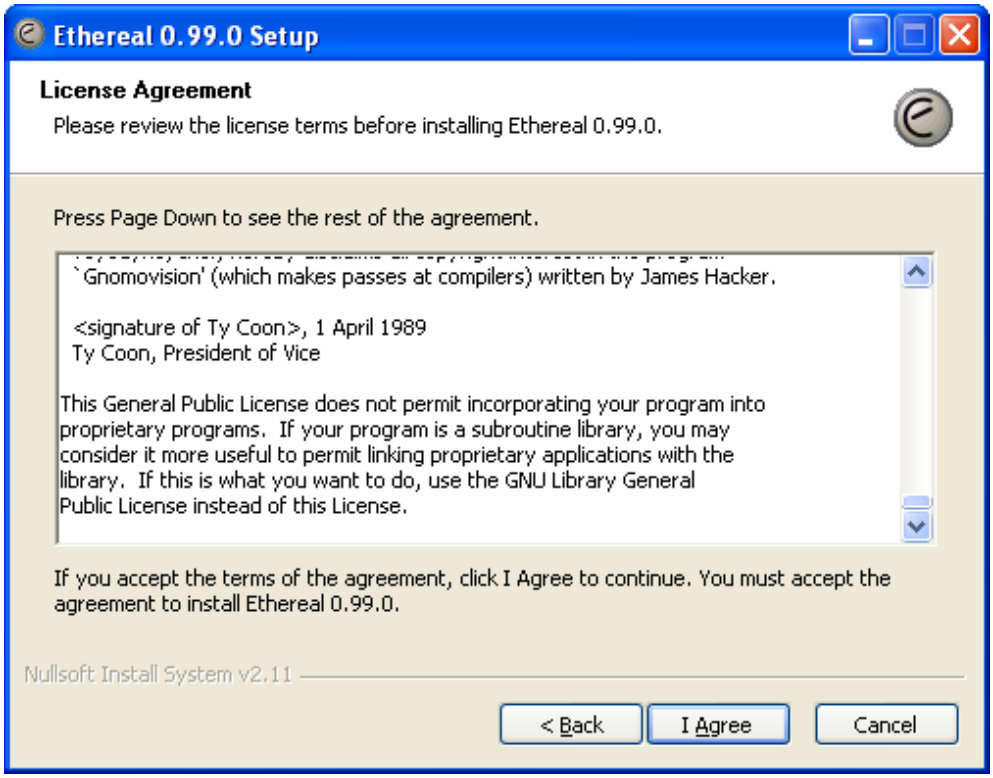

Fig. A1-12 Contrato de Licencia de Ethereal

Después se tiene que seleccionar los componentes a ser instalados; como se necesita la herramienta para poder analizar los *Logs* generados de manera gráfica se escogerá el Ethereal. El Tethereal que maneja lo mismo pero en forma de texto no es indispensable, por eso no se seleccionará. La opción “tools” si se instalará. Lo descrito se muestra en la figura A1-13.

Una vez seleccionados los componentes a ser instalados, se eligirá la creación de los accesos directos y la asociación de los archivos con la herramienta; es recomendable seleccionar todo para que el programa se encuentre a la mano del usuario.

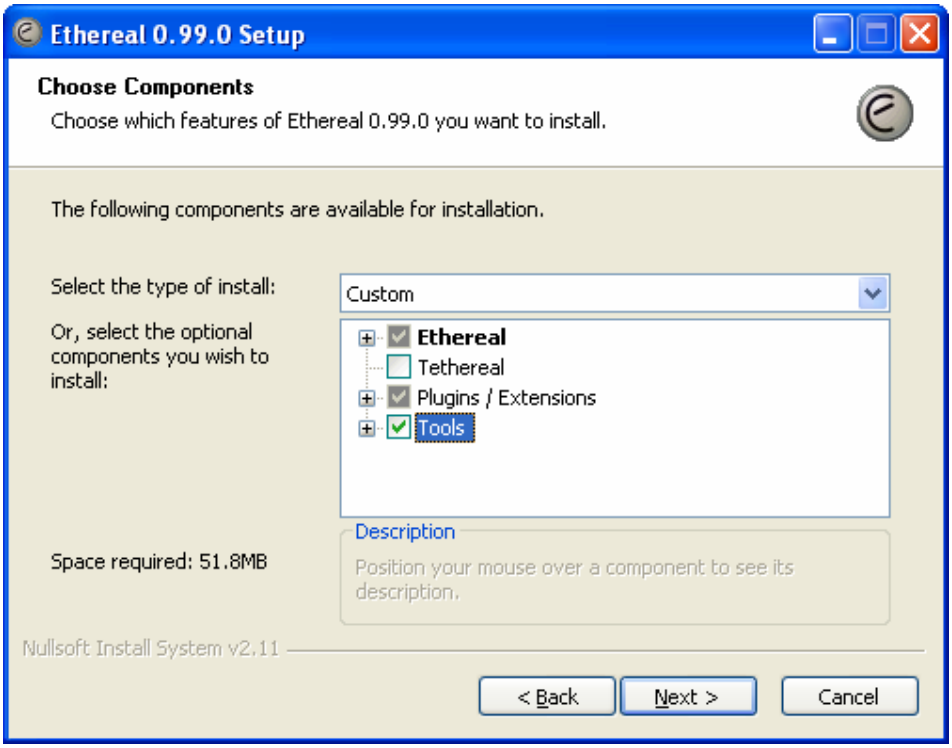

Fig. A1-13 Componentes a Instalar

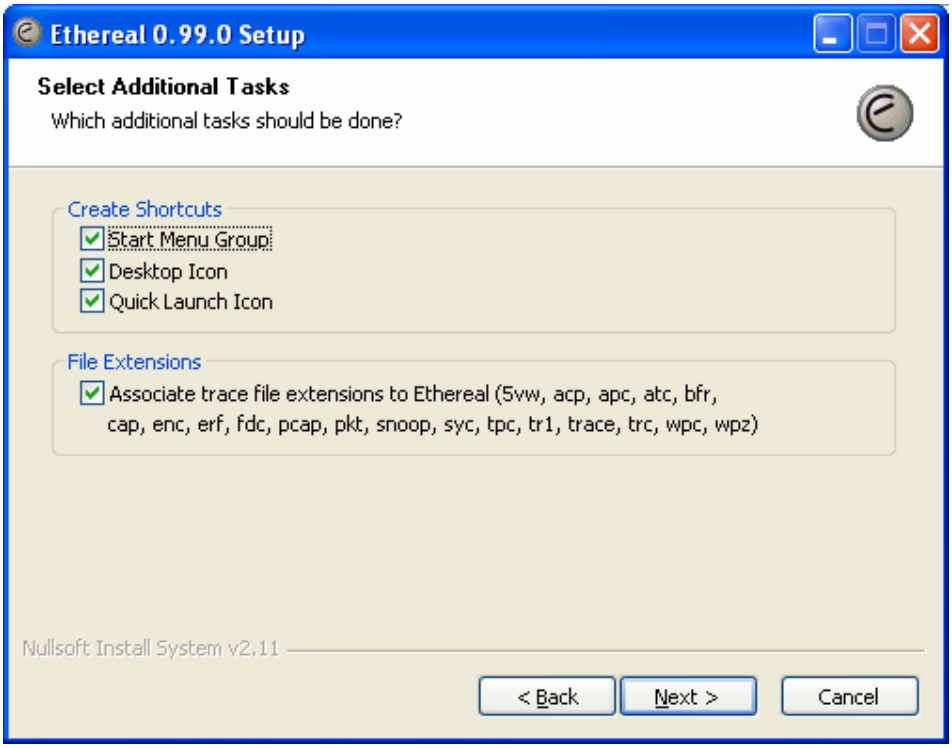

Fig. A1-14 Selección de tareas Adicionales

Una vez que se selecciona “next”, se tendrá que escoger la ubicación de la instalación; se escogerá la ubicación por defecto, pero cabe resaltar que la ubicación no altera la ejecución del programa (ver figura A1-15).

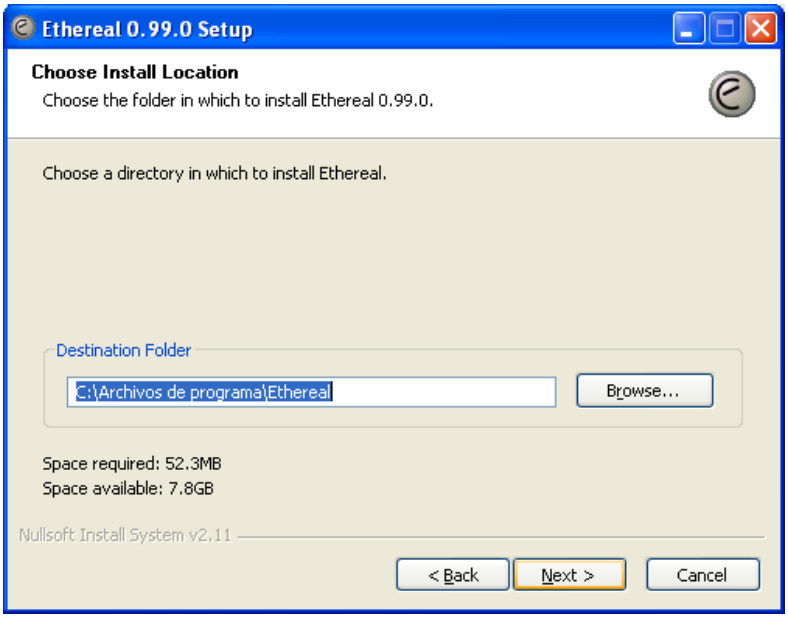

Fig. A1-15 Selección de ubicación de Instalación

Tal como se mencionó, el programa Ethereal depende del *FREEWARE* WinPCap, el cual si no se encuentra instalado se necesitaría instalarlo; como ya se lo dispone no se selecciona ninguna casilla y se hace clic en el botón “Install”

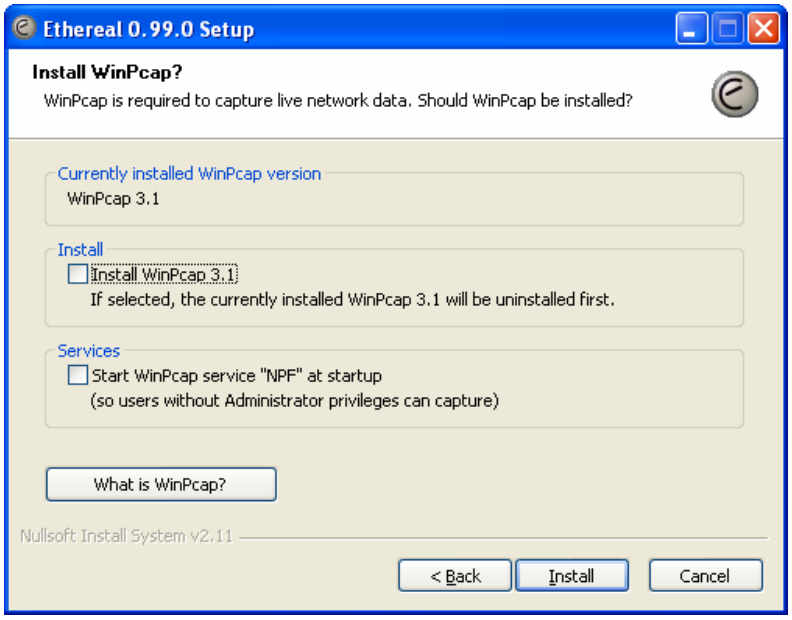

Fig. A1-16 Instalación de WinPCap

Al presionar el botón “install” se procede a la copia de archivos necesarios para la ejecución del programa, tal como se muestra en la figura A1-17. Si se copiaron todos los archivos correctamente y no existieron errores durante este proceso se presentará la pantalla de la figura A1-18 que indica que la instalación se realizó de manera satisfactoria.

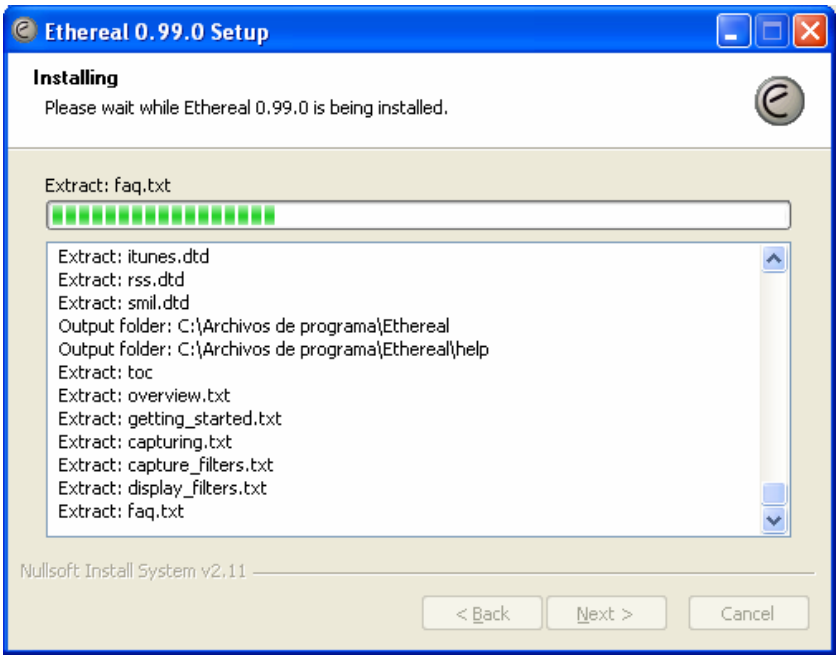

Fig. A1-17 Copia de archivos de Ethereal

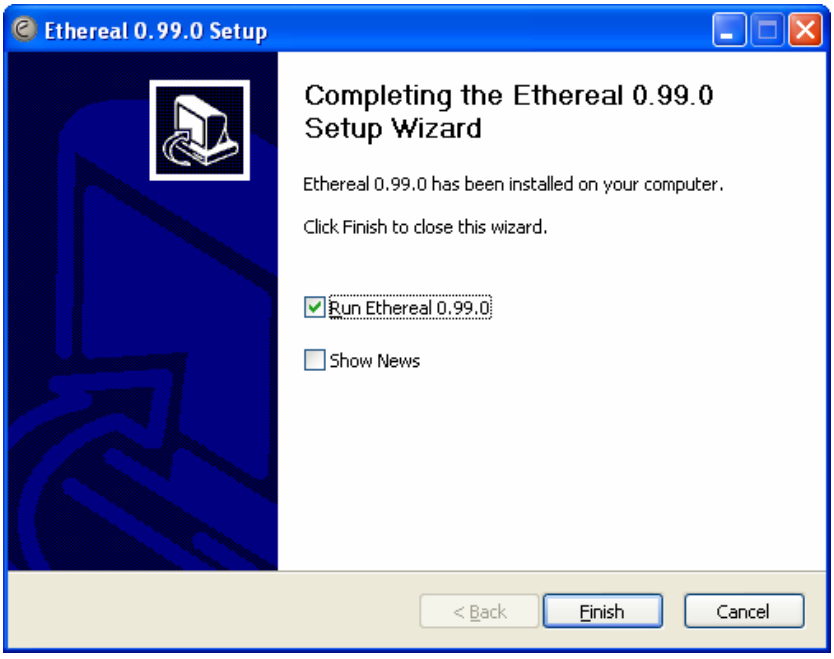

Fig. A1-18 Finalización de la instalación

La aplicación necesita que se cree una carpeta de nombre “log”, en la dirección c:\snort\bin\.

Por último la carpeta Auditoría de Red, correspondiente al programa y que se encuentra dentro del CD, debe copiarse en cualquier directorio de la computadora, y se ejecutará el archivo Auditoria.exe que se encuentra dentro de la carpeta.

ANEXO 2

MANUAL DE UTILIZACIÓN DEL SOFTWARE DE AUDITORÍA

MANUAL DE UTILIZACIÓN DEL SOFTWARE DE AUDITORIA DE RED

El software desarrollado tiene dos módulos principales, los cuales pueden ser escogidos en la primera pantalla que aparece cuando se inicializa la aplicación, tal como muestra la figura A2-1.

Cada uno de estos módulos pueden ser accesados presionando el botón "INGRESAR" debajo de cada uno de los nombres, así como desde los íconos de la barra de herramientas.

Fig. A2-1 Menú Principal del Software

A2.1 Módulo de escáner de puertos

Al escoger el módulo escáner de puertos aparecerá la pantalla mostrada en la Figura A2-2. Los datos y manera de llenar el diálogo es como se menciona a continuación.

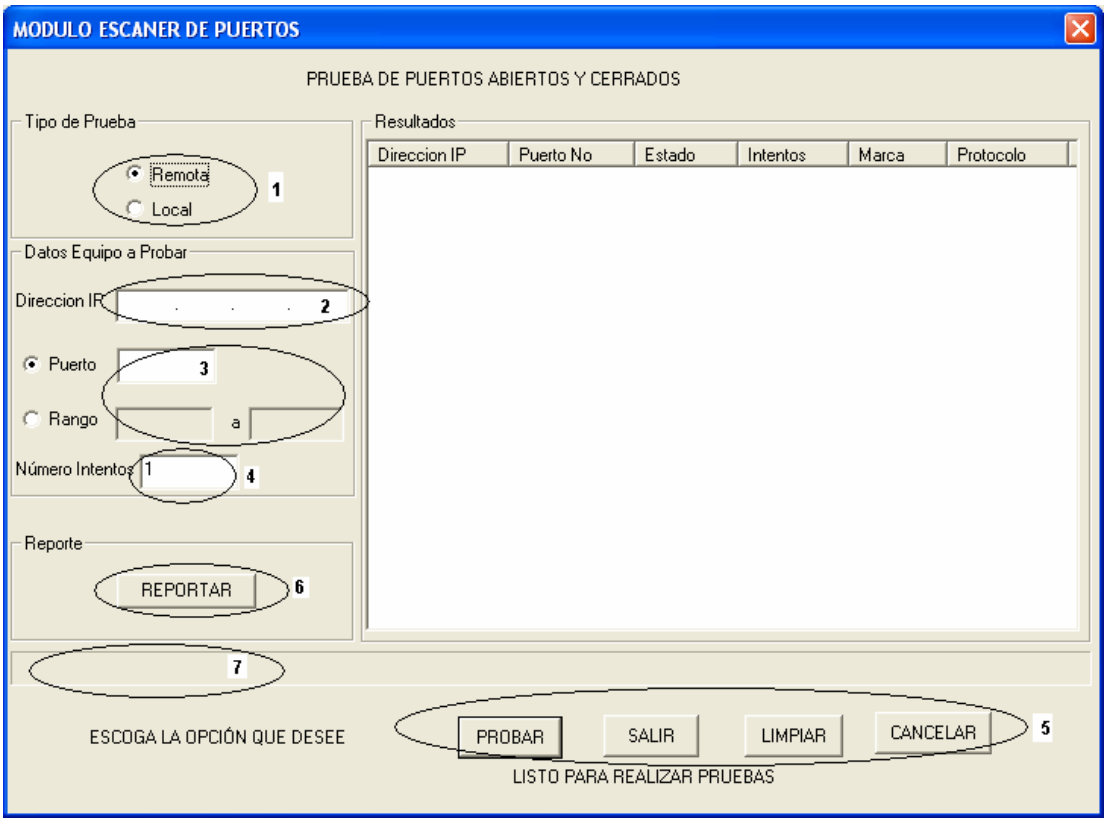

Fig. A2-2 Pantalla de Escaneo de Puertos

Tipo de Prueba

Se debe escoger si la prueba es local o remota. Por defecto la prueba es remota y está habilitado el ingreso de la dirección *IP* del equipo remoto; si se escogiese prueba local se deshabilitará el campo para el ingreso de la dirección *IP*.

Dirección *IP* del equipo Remoto

Debe ser ingresada de forma numérica, no se aceptan letras y los valores pueden ir de 0 a 255, excepto en el último campo que puede tener valores de 1 a 254.

Puerto o Rango de Puertos

La prueba puede hacerse a un puerto determinado o a un rango de puertos; si es a un rango de puertos se deberá ingresar el número del puerto inferior y del puerto superior; el tiempo de respuesta depende del número de puertos que se estén analizando.

Número de Intentos

El número de intentos por defecto es 1; en caso de que salga el resultado “CERRADO” se tomará en cuenta el número de intentos. Este valor debe ser puesto para obtener un mejor resultado; del número colocado en esta casilla depende el tiempo de procesamiento. Es recomendable poner un valor entre 1 y 5, el programa no limita los valores que pueden ser ingresados.

Botones Probar, Cancelar, Salir y Limpiar

Al presionar el botón Probar, se ejecuta el programa desplegando los resultados en la lista del lado derecho.

Al presionar el botón Cancelar, se cancela el proceso presentando los resultados parciales de las pruebas que se lograron realizar.

Al presionar el botón Limpiar, se borra la lista de resultados para la realización de nuevas pruebas.

Al presionar el botón Salir, se sale del programa sin poder presentar resultados parciales.

Botón Reportar

Este botón está asociado con el reporte de los datos hacia un archivo de excel (*.xls), el cual puede guardarse en cualquier directorio o ubicación del computador.

Errores que pueden presentarse al realizar la prueba

Los errores que se pueden tener en el módulo de Escáner de puertos, se dan por: dejar en blanco la dirección *IP* de elemento de red, dejar en blanco el puerto de prueba, dejar en blanco uno de los dos valores de puertos del rango que se quiere analizar.

Dirección *IP* en blanco

Cuando se quiere realizar una prueba remota y no se ingresa una dirección *IP*, el programa detectará este problema y presentará un mensaje indicando dicho inconveniente.

Fig. A2-3 Error Dirección *IP* en Blanco

Dirección *IP* inconsistente

Esta advertencia se dará cuando en la dirección *IP* se quiera ingresar una dirección de Loopback o cuando se quiera ingresar en el último octeto de la dirección valores mayores o iguales a 255.

Fig. A2-4 Error Dirección *IP* no lógica

Puerto en blanco

Esta advertencia se dará cuando no se ingresa el puerto correspondiente para la prueba.

Fig. A2-5 Error Puerto en Blanco

Rango de Puertos en blanco o datos Inconsistentes

Esta advertencia se dará cuando los valores de los puertos sean menores a 0, o que no se cumpla que el valor del puerto inferior sea menor del superior.

Fig. A2-6 Error en Rango de Puertos

Realizar el reporte sin datos obtenidos previamente

Esta advertencia se dará cuando se quiera realizar un reporte de resultados sin haber obtenido datos previamente.

Fig. A2-7 Error de no datos

A2.2 Módulo de interfaz para snort

Al escoger el módulo de Sistema de Detección de Intrusos, aparecerá la pantalla de la figura A2-8, donde se tiene que seleccionar el modo de funcionamiento, ingresar el número de interfaz de red y escoger entre cuatro botones que tienen la siguiente utilidad.

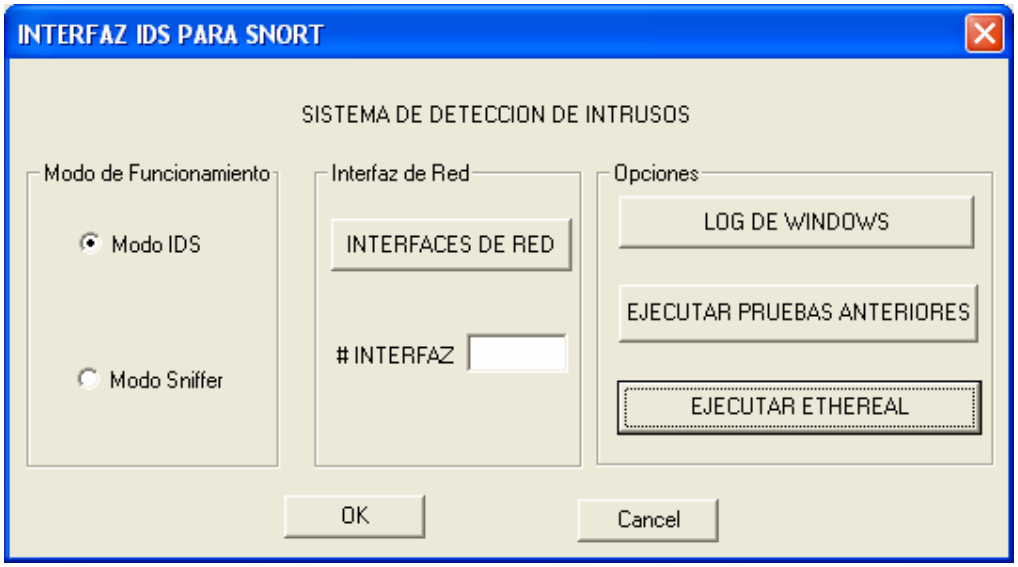

Fig. A2-8 Diálogo para escoger módulo de funcionamiento e Interfaz.

Botón Interfaces de Red

Como un equipo puede tener más de una tarjeta de conexión a redes, sea ésta Ethernet, Inalámbrica, Fax modem; se debe especificar a *Snort* cuál de éstas se quiere analizar. Es por eso que para conocer el número que *Snort* tiene asociado a la interfaz, se tiene que presionar este botón, el cual accionará un archivo de ejecución por lotes para *DOS* indicando todas las tarjetas de red del equipo seguidas del número correspondiente.

Botón Ejecutar Pruebas Anteriores

Mediante este botón, se tiene la opción de ejecutar directamente un archivo de ejecución por lotes antes creado, optimizando así el tiempo en la creación de reglas y configuración de parámetros que van a ser ocupados de manera similar en una red de datos.

Botón Log de Windows

Mediante este botón, se ingresa al Visor de Sucesos de Windows (Windows Xp y Windows 2000), para observar las alertas y alarmas creadas por *Snort,* si es que existieran.

Botón Ejecutar Ethereal

El cual sirve para ejecutar el programa Ethereal y poder analizar los *logs* generados por *Snort.*

Radio Button Modo de funcionamiento

Mediante el *radio button,* se podrá seleccionar la utilización de cualquiera de los dos modos de funcionamiento; cabe recalcar que si no se ingresa el número de interfaz, el programa no podrá continuar.

Selección de modo de funcionamiento *IDS*

En este modo, se presenta la pantalla de la figura A2-9, en la cual se puede visualizar tres regiones importantes; la primera región (información principal) que permite escoger todos los datos que se desplieguen, la segunda (reglas de filtrado y alertas) que sirve para seleccionar si se usan reglas almacenadas o nuevas y la tercera (generación de reporte) que está relacionada con el nombre del *Log* de datos a generarse y el número de paquetes a analizar.

Información Principal

Todas estas opciones están relacionadas con las funcionalidades de *Snort* de *DOS.*

Reglas de filtrado y alertas

Aquí se tendrá la opción de crear u ocupar las reglas de filtrado de tráfico para *Snort,* que ayudarán a discriminar el tráfico de datos.

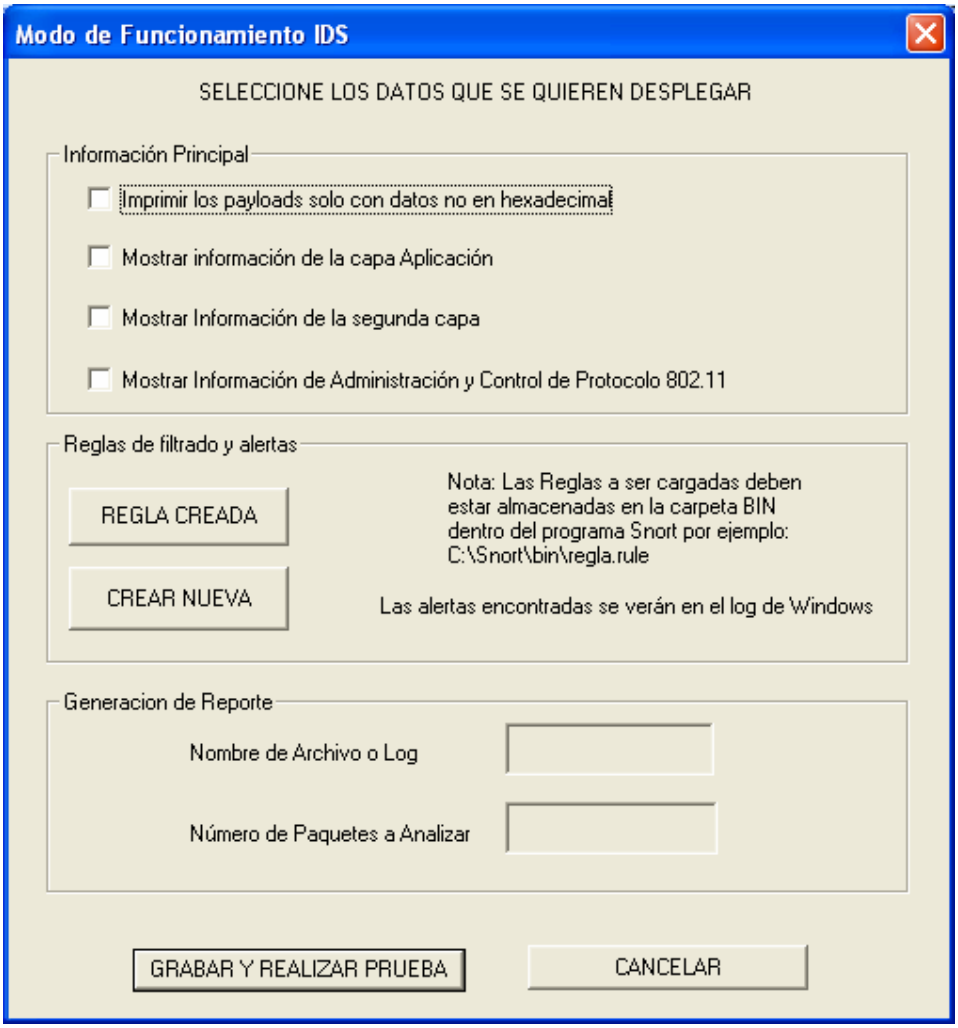

Fig. A2-9 Interfaz de modo de funcionamiento *IDS*

Creación de una nueva regla

Para la creación de una regla, se tendrá que seleccionar el botón correspondiente, en donde al presionarlo se desplegará el diálogo de la figura A2-10; los controles que se tienen en este diálogo, se presentan a continuación.

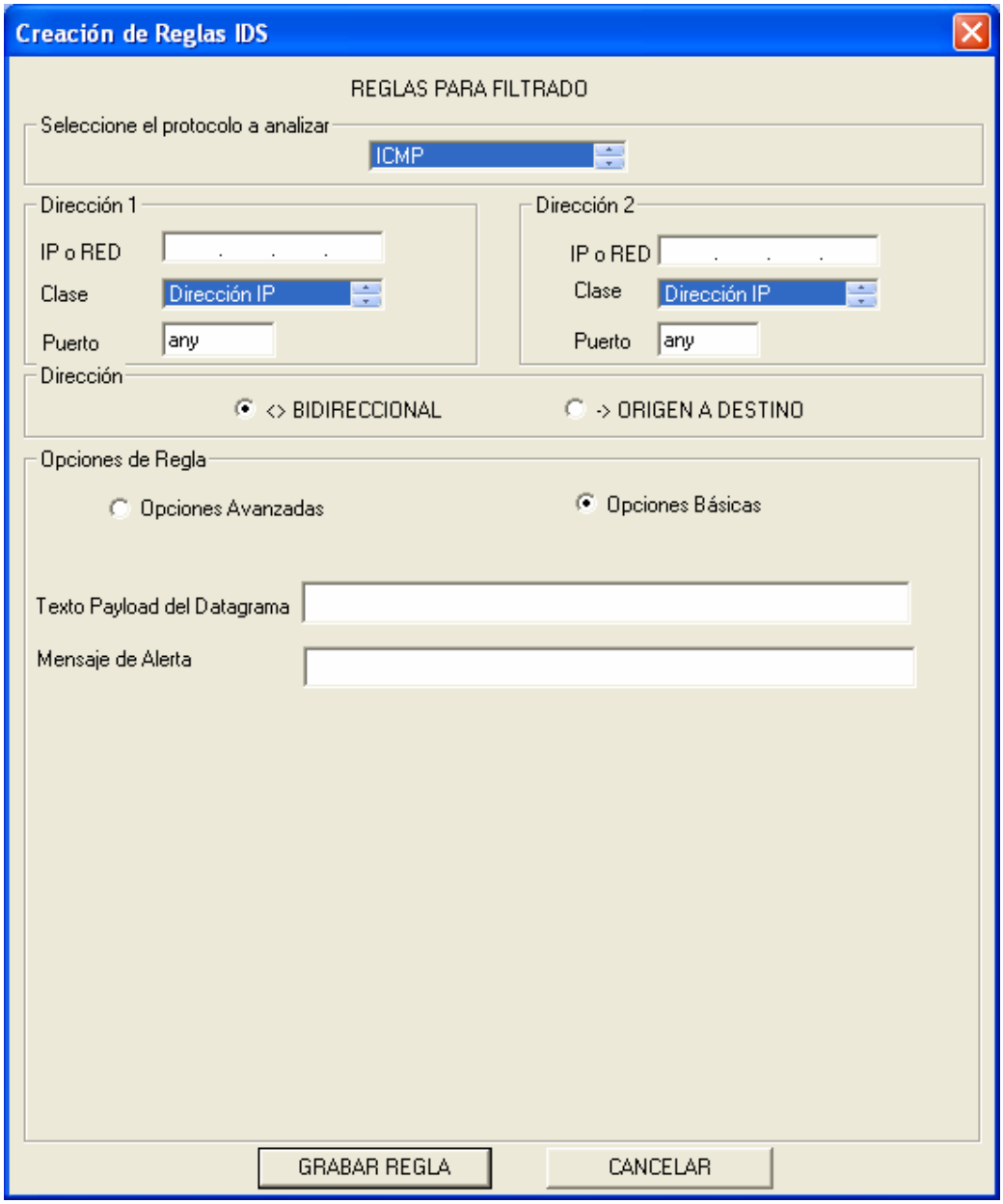

Fig. A2-10 Creación de una nueva regla para alarma

Protocolo

Es el protocolo que va a ser seleccionado, éste se tiene que elegir de la lista. Por defecto se tiene escogido el *ICMP* pero podría seleccionarse cualquier otro dando un clic en la lista y cerciorándose que esté resaltado de color azul.

Una de las funcionalidades de *Snort* es que se puede analizar pequeñas proporciones de tráfico (de máquina a máquina), medianas proporciones de tráfico (de máquina a red) o grandes proporciones de tráfico (toda la red). Para esto se tienen las opciones que se indican a continuación.

Dirección 1

Es origen de tráfico.

Si se deja en blanco la dirección *IP*, el software interpreta que se quiere analizar todo el tráfico desde cualquier dirección.

Si se ingresa una dirección *IP*, se podrá seleccionar en la lista si la dirección ingresada corresponde a un host, una red o una subred con máscara /8, /16 o /24.

El espacio correspondiente a puerto por defecto tiene la palabra “any”, la cual analizará todo tipo de aplicaciones (sin discriminarlas), si se pone un número, se analizará la aplicación correspondiente al puerto escrito.

Dirección 2

Es el destino de tráfico.

Con similares características a la explicadas en Dirección 1.

Discriminación de dirección

La dirección *IP* y el número de puerto del lado izquierdo del operador de dirección (->) son considerados tráfico de máquina fuente; la dirección *IP* y el número de puerto del lado derecho del operador son considerados tráfico para máquina destino. Existe también el operador (<>) que considera a la pareja de dirección y puerto en ambos sentidos, esto es conveniente para analizar ambos lados del intercambio de información.

Opciones Básicas

Mediante esta opción se puede analizar el contenido de los paquetes que los usuarios están enviando o recibiendo en la red, de una manera sencilla, ya que solo se pide el texto que se quiere observar y el texto de la alarma para el Visor de Sucesos de Windows, obviando las palabras reservadas de *Snort*, las cuales el programa las escribe automáticamente.

Fig. A2-11 Opciones Básicas de Reglas

Texto Payload del Datagrama: Es el texto a analizar, el cual puede estar enviándose o recibiéndose en la parte de datos de los paquetes.

Mensaje de alerta: Es el mensaje de alarma que se pondrá en el Visor de Sucesos de Windows.

Opciones Avanzadas

Con esta opción, el usuario puede examinar más a fondo el protocolo escogido, partes de la trama, banderas a nivel de bits; se necesitan conocimientos avanzados de redes de datos para poder ocupar de manera óptima esta opción.

Todas las palabras reservadas como msg, ttl, dsize, etc, pueden ser escritas o escogidas de la lista que se encuentra en los controles de opciones avanzadas,

lo importante es que se deba mantener la sintaxis para no producir errores en la ejecución de los archivos de *Snort*, cada palabra reservada que puede ser escogida viene acompañada de su respectiva explicación, formato de escritura y un ejemplo.

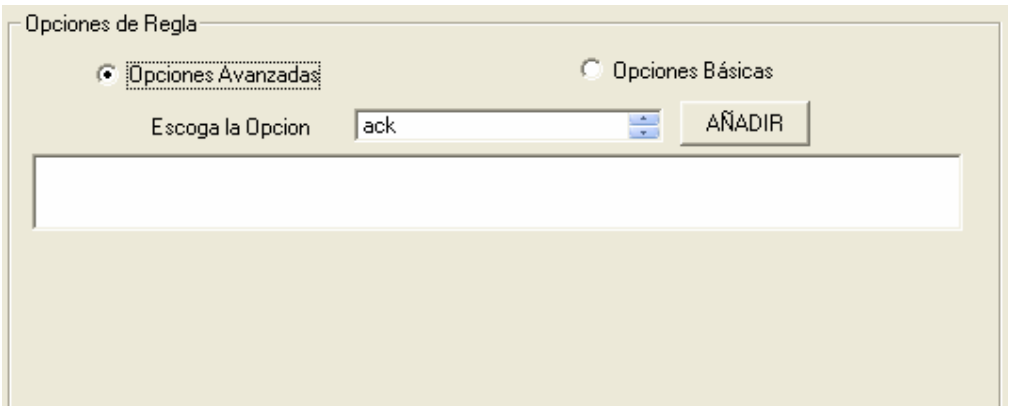

Fig. A2-12 Opciones Avanzadas de Reglas

Utilizar una regla escrita previamente

Para utilizar una regla escrita previamente, se tendrá que seleccionar el botón correspondiente, en donde al presionarlo se desplegará el diálogo para abrir archivos y se tendrá que escoger la regla requerida.

Selección de Modo de funcionamiento *SNIFFER*

Un *sniffer* es un es un programa de captura de paquetes de red, generalmente utilizado con fines maliciosos ya que intercepta información en tránsito, pero también puede usarse en forma responsable con fines técnicos de auditoría y análisis de vulnerabilidades como es el presente caso.

Al escoger el modo de funcionamiento *Sniffer*, solo se tiene que ingresar el nombre del log a ser generado y el número de paquetes que se quiere analizar (ver figura A2-13).

Como se explicó, para la visualización de los *logs* generados por la herramienta *Snort* se utilizará el programa Ethereal.

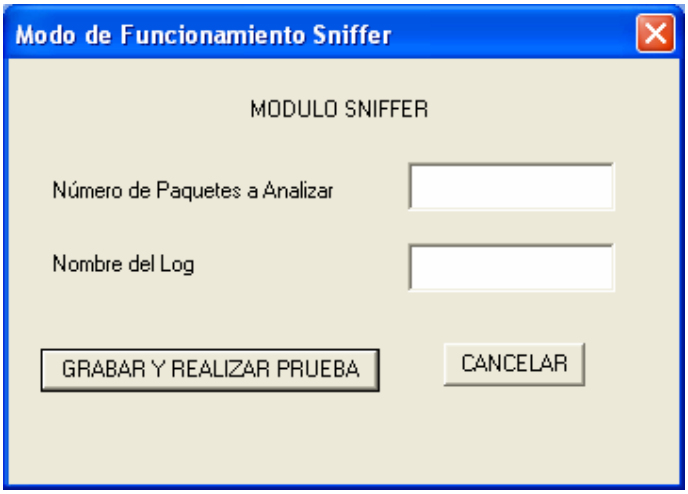

Fig. A2-13 Modo de funcionamiento Sniffer

Errores que se pueden presentar al realizar una prueba

Cuando no se ingresa el número de paquetes a analizar, se informará al usuario de dicho error.

Fig. A2-14 Error de no ingreso de número de paquetes a analizar

Cuando no se ingresa el nombre del log a generarse, el sistema informa de dicho error, como se indica a continuación.

Fig. A2-15 Error de no ingreso de Nombre de *log* a generarse

Ejecución de un archivo relacionado a una prueba anterior guardada

Cada vez que se hace una prueba del *IDS*, sea en modo *Sniffer* o Sistema de Detección de Intrusos se genera un archivo de ejecución por lotes, el cual puede ser ocupado posteriormente.

Para realizar esta tarea, lo que se tiene que hacer es escoger en el menú principal la opción de Sistema de detección de Intrusos, luego en la siguiente pantalla seleccionar el botón de Ejecutar pruebas anteriores como se muestra en la figura A2-16.

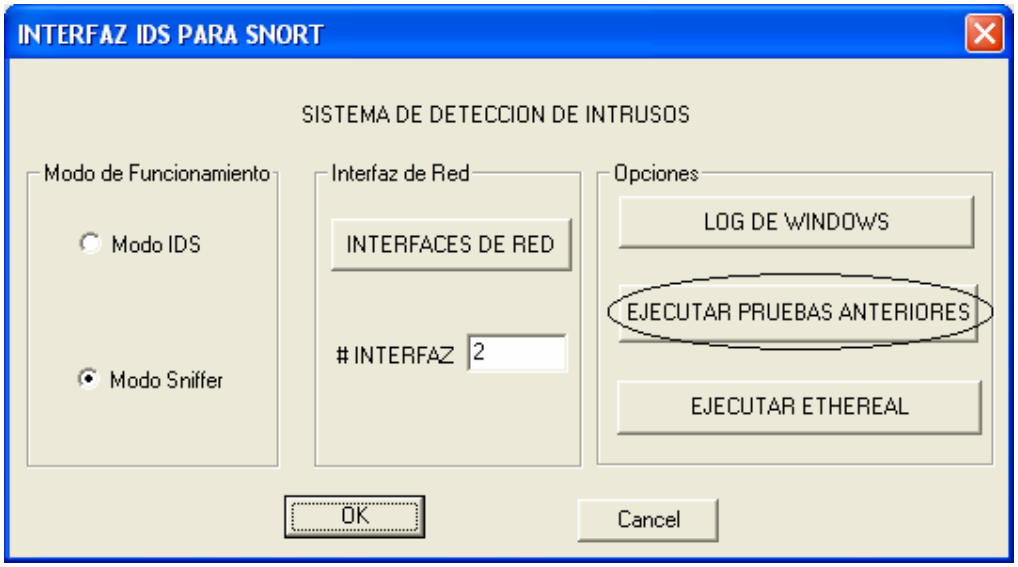

Fig. A2-16 Selección de ejecución de prueba anterior

Donde al presionarlo, se deberá escoger el archivo que corresponda a la prueba realizada anteriormente, una vez después de que se lo seleccione se ejecutará en DOS, comenzando a captar los datos y dar las alarmas que hayan estado configuradas.

ANEXO 3

EJEMPLO DE PRUEBAS Y RESULTADOS OBTENIDOS

EJEMPLO DE PRUEBAS Y RESULTADOS OBTENIDOS

Ejemplo de comprobación del estado del puerto 80 de la máquina de dirección *IP* 192.168.2.1, mediante el módulo de Escáner de Puertos.

Para comprobar el estado de un puerto de una máquina, lo que se tiene que hacer es seleccionar en el menú principal de la aplicación el módulo Escáner de Puertos, luego en la interfaz presentada, se tendrá que llenar los datos necesarios para la prueba, que para el caso presentado en la figura A3-1 son: Prueba remota, Dirección *IP*: 192.168.2.1, número de puerto 80.

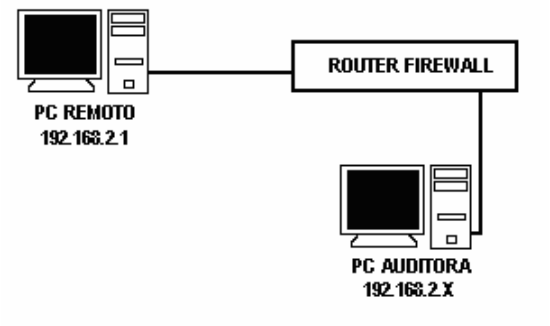

Fig. A3-1 Esquema de prueba de un puerto en una máquina remota

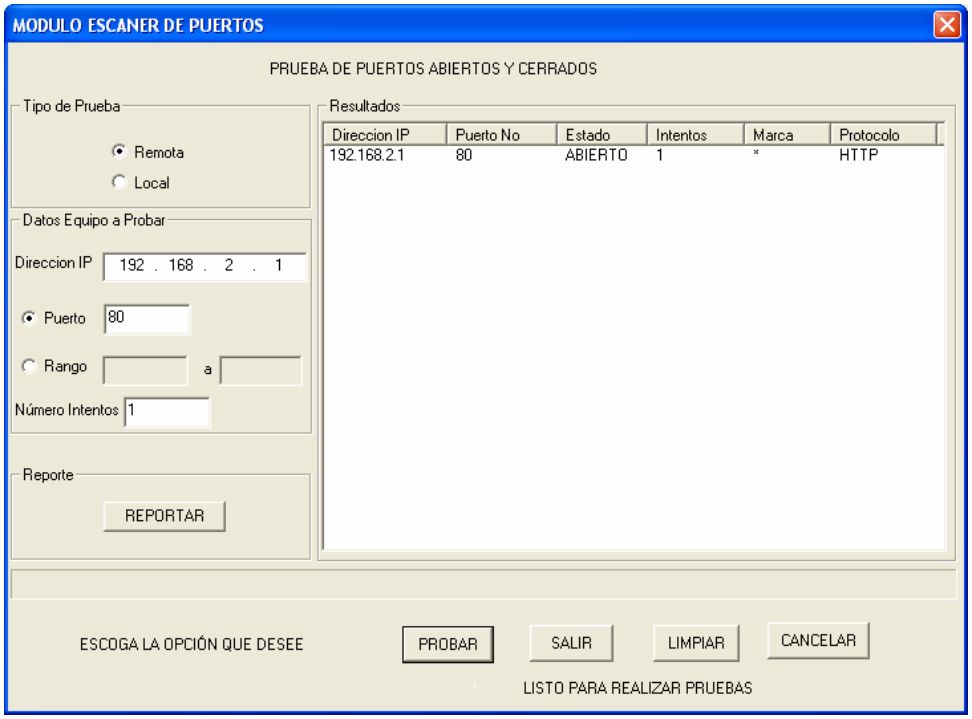

Fig. A3-2 Prueba remota del puerto 80 en la máquina 192.168.2.1

Al presionar el botón Probar, el programa comienza a hacer la prueba y despliega el resultado en la lista del lado derecho, en esta lista constan la dirección *IP* del elemento de Red, el número del puerto probado, el estado del puerto, el número de intentos, la marca la cual será "*" cuando el puerto este abierto y en blanco en caso de estar cerrado, acompañado del nombre de protocolo si es que es conocido.

Para crear un reporte de datos, se presiona sobre el botón "REPORTAR", el cual muestra el diálogo de la figura A3-3, en donde se solicita el nombre del archivo de Reporte de puertos a generarse.

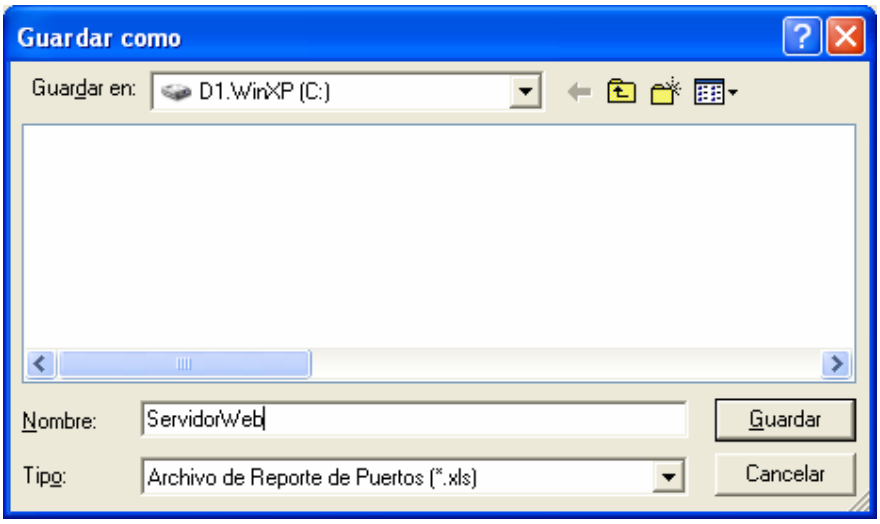

Fig. A3-3 Nombre de Archivo de reporte

Ejemplo de utilización del Sistema de Detección de intrusos con opciones básicas, analizando un cierto número de paquetes, que tenga entre sus datos la palabra "BOMBA"

En el menú principal se escoge el módulo *IDS*, se selecciona el modo de funcionamiento como Sistema de Detección de Intrusos, se ingresa el número de la interfaz, si no se lo conoce se ocupa el botón correspondiente, en el caso del ejemplo el número es 2 (dos) como se indica en la figura A3-4.

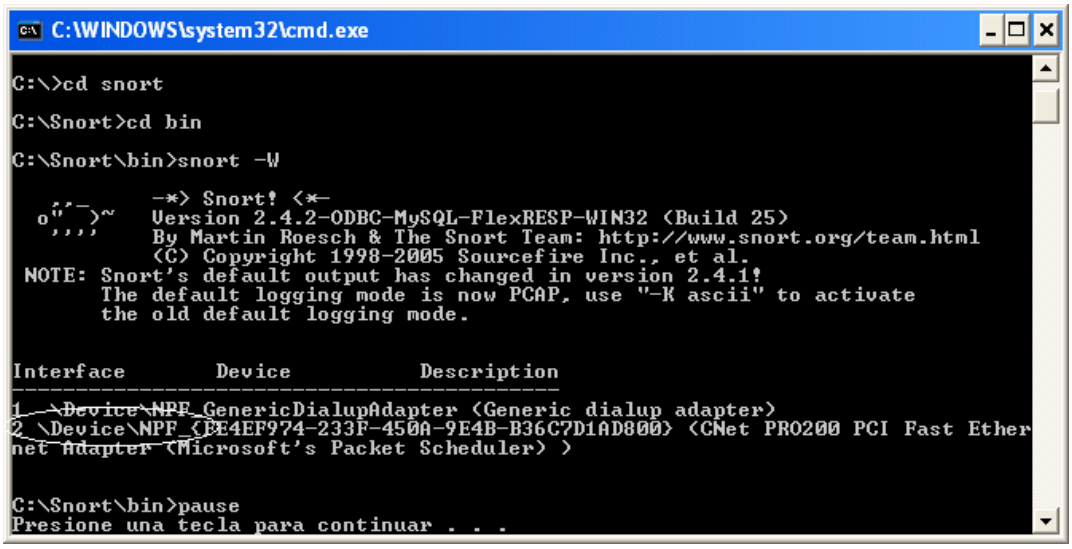

Fig. A3-4 Interfaces de red

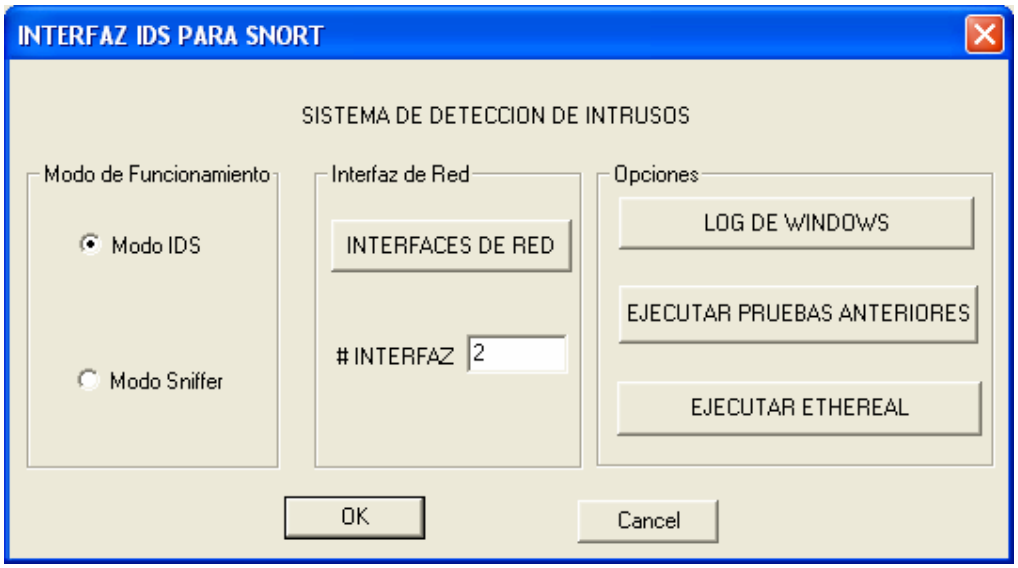

Fig. A3-5 Selección de Modo de Funcionamiento e ingreso de número de interfaz

Escogido el modo de funcionamiento y puesto el número de interfaz, se presiona el botón OK, el cuál presenta el diálogo de la figura A3-6, en donde se va a seleccionar las tres primeras opciones: imprimir los payloads solo con datos no hexadecimal, mostrar información de capa aplicación y mostrar información de la segunda capa; los que facilitarán el análisis posterior del log generado.

Como no se tiene una regla creada para dicho propósito, se debe dar un clic sobre el botón crear nueva, haciendo esto se va a presentar la pantalla de la figura A3-7, como el contenido de los paquetes a analizar se encapsulan en tramas *TCP* se tendrá que escoger este tipo de protocolo, además como se quiere analizar todo el tráfico, las Dirección 1 y 2 se las dejará en blanco (por defecto) al igual que los valores de los puertos, que quedarán con las palabras "any"; se analizará bidireccionalmente seleccionando "<>". Por último se tendrá que poner en los controles de ingreso de datos:

Texto Payload del Datagrama: Bomba.
Mensaje de Alerta: Paquete con contenido Bomba encontrado.

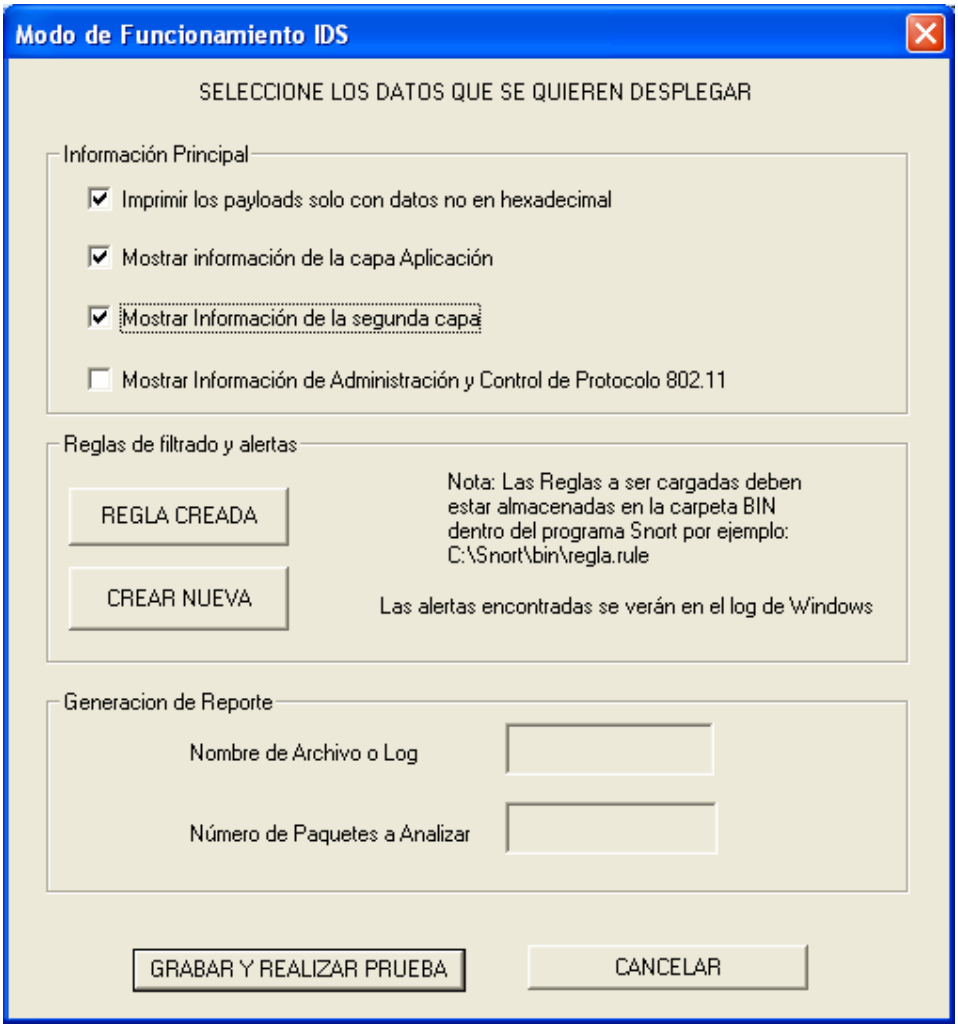

Fig. A3-6 Datos para funcionamiento IDS

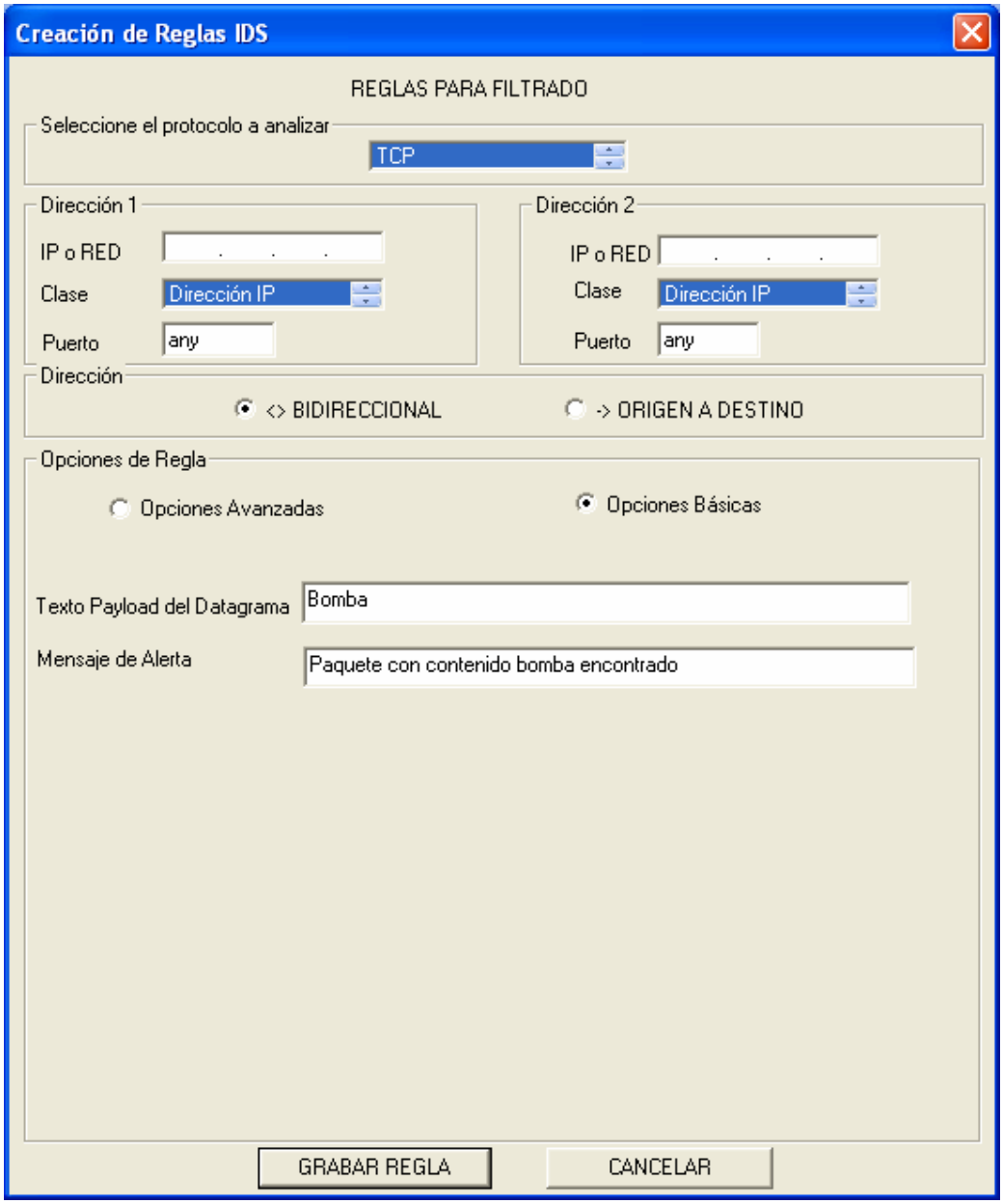

Fig. A3-7 Nueva Regla de Filtrado

Al presionar el botón Grabar Regla, se tendrá que guardar la regla creada, la extensión de la regla será escrita automáticamente, es por eso que solo se tiene que poner el nombre de la misma, como se indica a continuación.

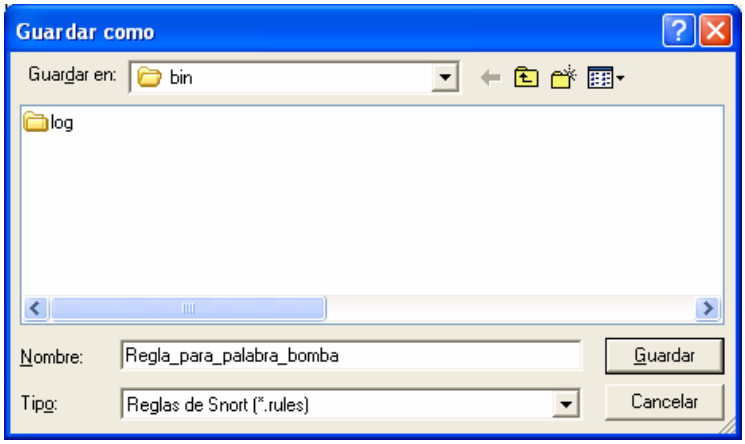

Fig. A3-8 Guardar una regla de Snort

Guardada la regla, se muestra un cuadro de mensaje indicando el texto de la regla, para cerciorase de que fue escrita correctamente.

Fig. A3-9 Texto de Regla Creada

Cerciorado el texto de la regla, se debe llenar la última parte del modo de funcionamiento *IDS,* como es el nombre del *Log* a ser generado y el número de paquetes que se quieren analizar; al aceptar todos los parámetros escritos y escogidos se crea el archivo de ejecución por lotes para *Snort,* que como se indicó puede ser utilizado posteriormente en otros análisis similares (ver figura A3-10).

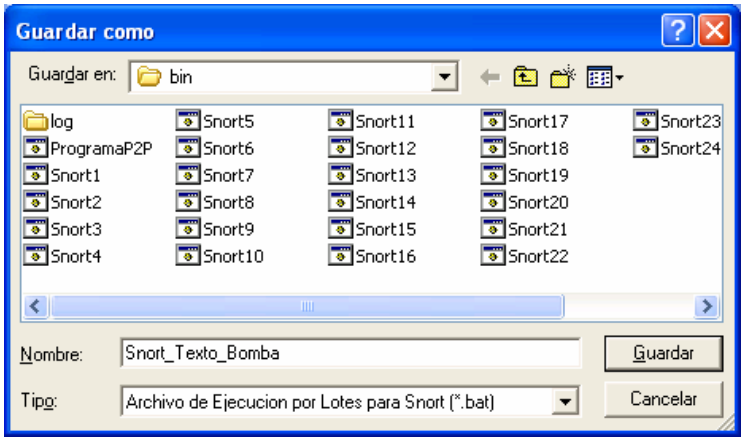

Fig. A3-10 Guardar archivo de ejecución por Lotes para Snort

Después de guardar el archivo de ejecución por lotes y si este no tiene errores el programa llama al símbolo del sistema para la ejecución del archivo de *Snort* en *DOS*. Tal y como se indica a continuación.

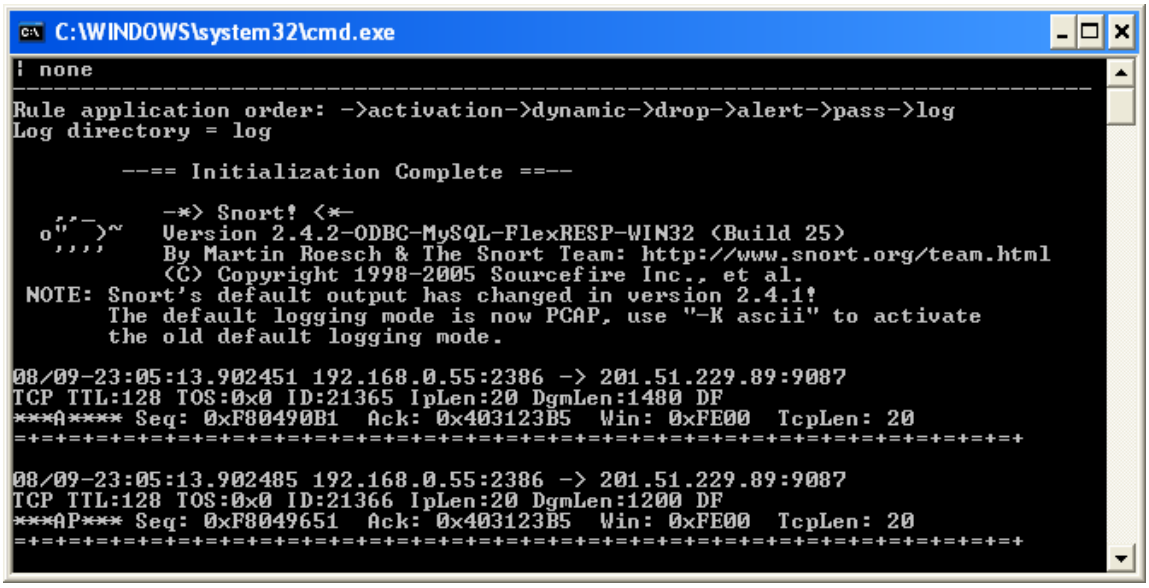

Fig. A3-11 Ejecución de *Snort* en modo DOS

Por último si se generaron alertas, estas serán mostradas en el Visor de Sucesos de Windows, el cual puede ser abierto desde el menú principal del Sistema de detección de Intrusos presionando el botón correspondiente, tal como se muestra a continuación.

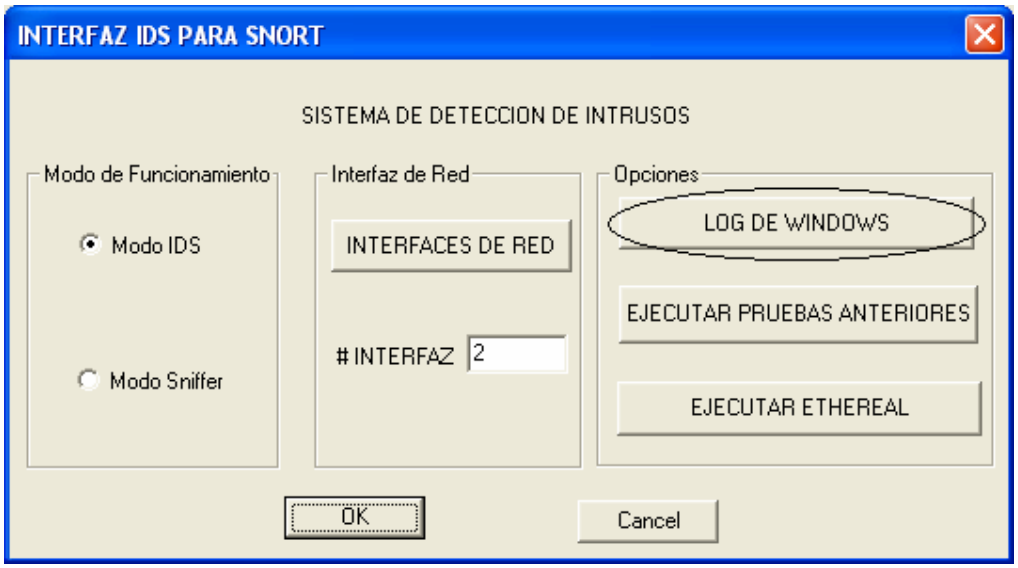

Fig. A3-12 Abrir el log de Windows

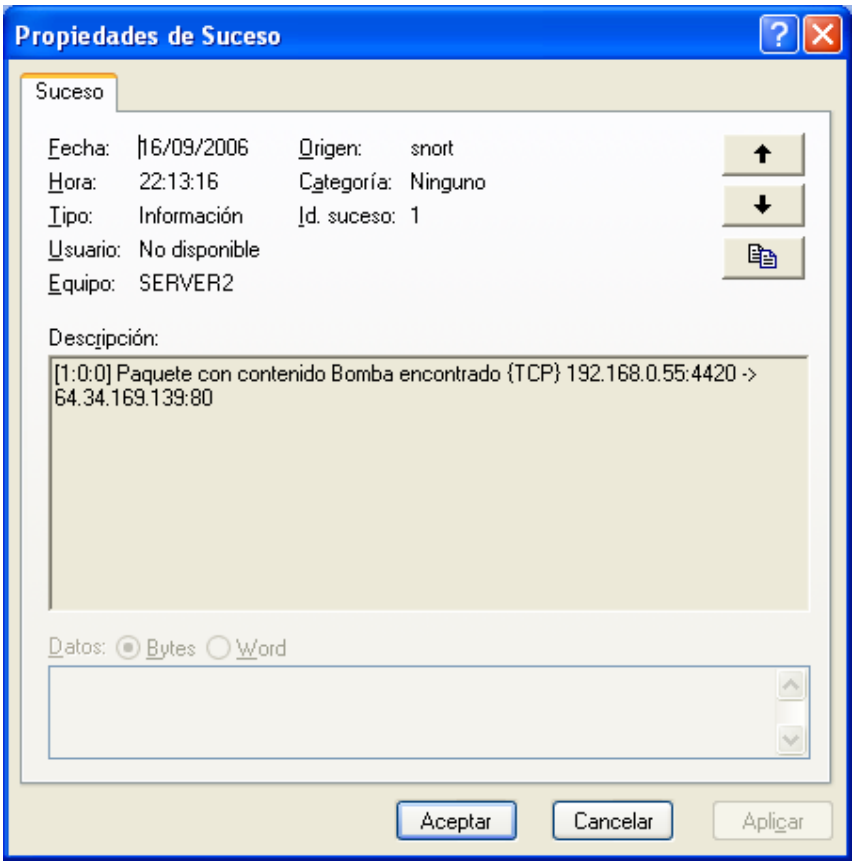

Fig. A3-13 Alarmas generadas en el Log de Windows

Ejemplo del sistema de de Detección de Intrusos con opciones avanzadas, analizando el paquetes *ICMP* que se envía desde una máquina cliente 192.168.2.5 hacia el servidor 192.168.2.1 y que tenga el tamaño en bytes mayor a 32 (tamaño común del ping normal).

En el menú principal se escoge el módulo IDS; se selecciona el modo de funcionamiento como Sistema de Detección de Intrusos, se ingresa el número de la interfaz, si no se lo conoce se ocupa el botón correspondiente que para el caso del ejemplo el número es 2, como se indica en la figura A3-14.

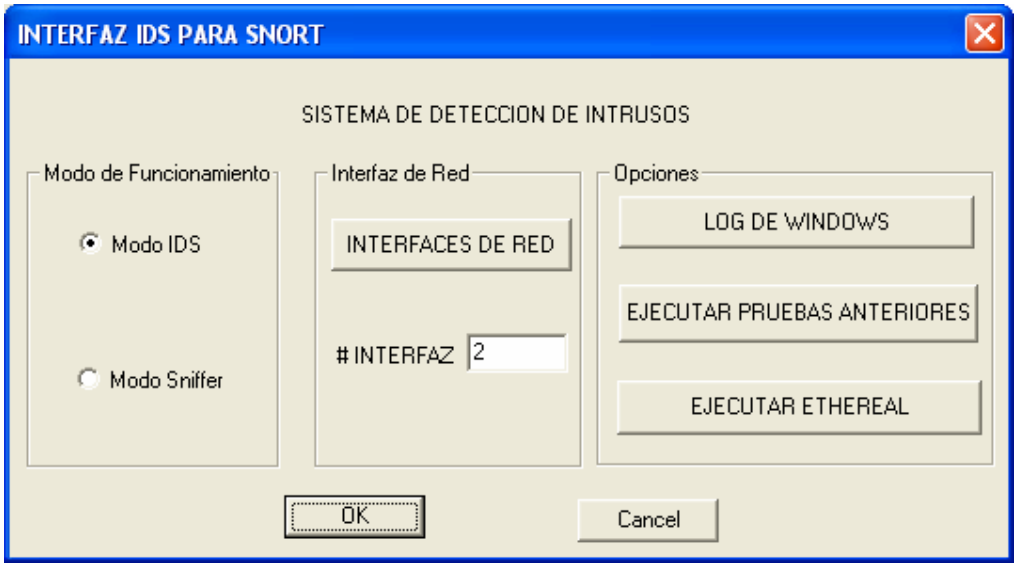

Fig. A3-14 Selección de Modo de Funcionamiento e ingreso de número de interfaz

Escogido el modo de funcionamiento y escrito el número de interfaz, se presiona el botón OK, donde se presenta el diálogo para ingreso de datos (ver figura A3-15), en el cual se va a seleccionar las tres primeras opciones: imprimir los payloads solo con datos no hexadecimal, mostrar información de capa aplicación y mostrar información de la segunda capa.

Como no se tiene una regla creada para dicho propósito se debe dar un clic sobre el botón crear nueva, haciendo esto se va a presentar la pantalla de la figura A3-16, donde se tendrá que escoger el tipo de protocolo que para el caso del ejemplo es *ICMP*, la Dirección 1 es 192.168.2.5 y la Dirección 2 es 192.168.2.1, los puertos quedarán con las palabras “any” para no discriminar por puerto; por último se analizará el ping en dirección (->) (origen a destino) ya que como se indicó se quiere desde la máquina 192.168.2.5 a la 192.168.2.1.

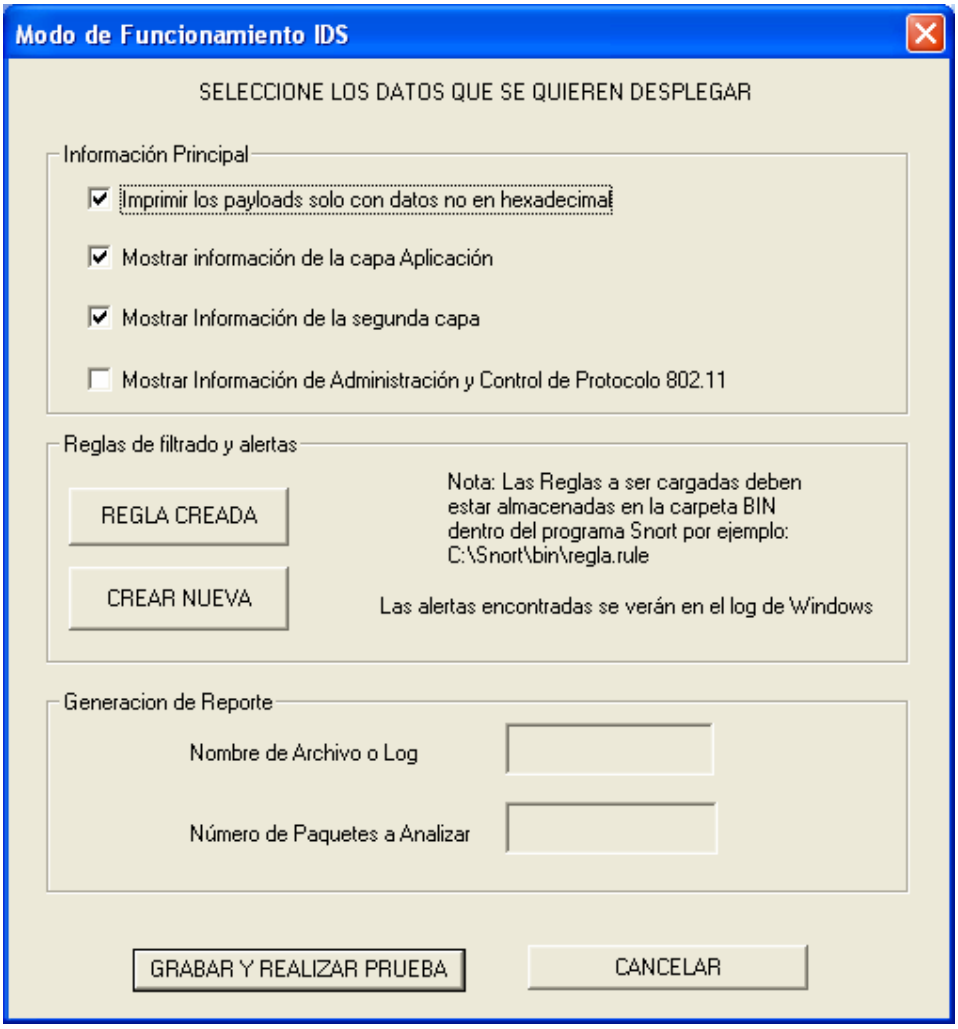

Fig. A3-15 Parámetros para generación del Log

Como se va a realizar una regla con formato avanzado se debe seleccionar el *radio button* correspondiente “Opciones Avanzadas”, en el cuadro de texto que se despliega, se deberá escribir el siguiente texto:

msg: "Ping con número de bytes mayor a 32”; dsize: >32

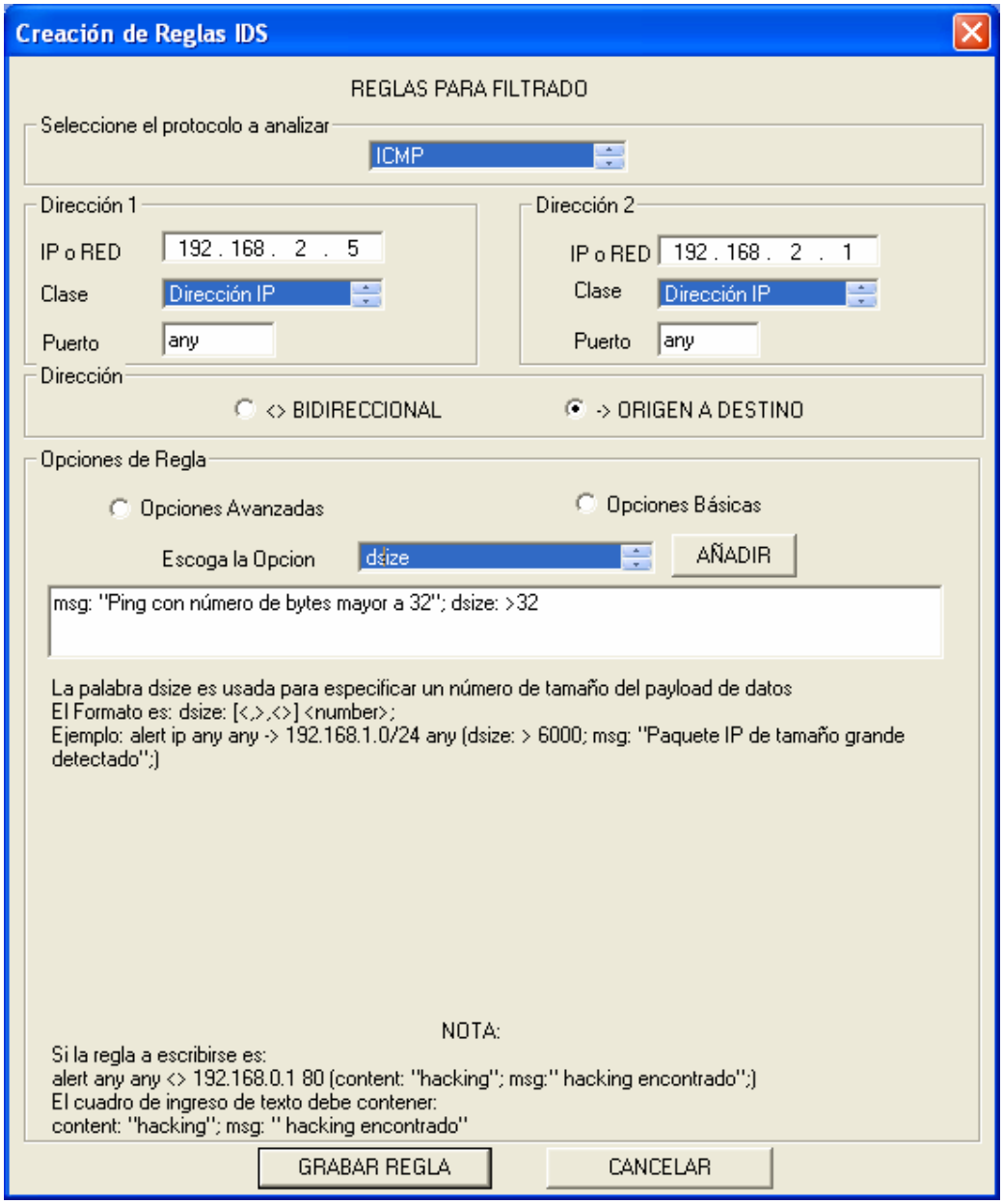

Fig. A3-16 Creación de una regla con Opciones Avanzadas

Al presionar el botón Grabar Regla, se tendrá que guardar la regla creada, la extensión de la regla será escrita automáticamente, es por eso que solo se tiene que poner el nombre de la misma (ver figura A3-17).

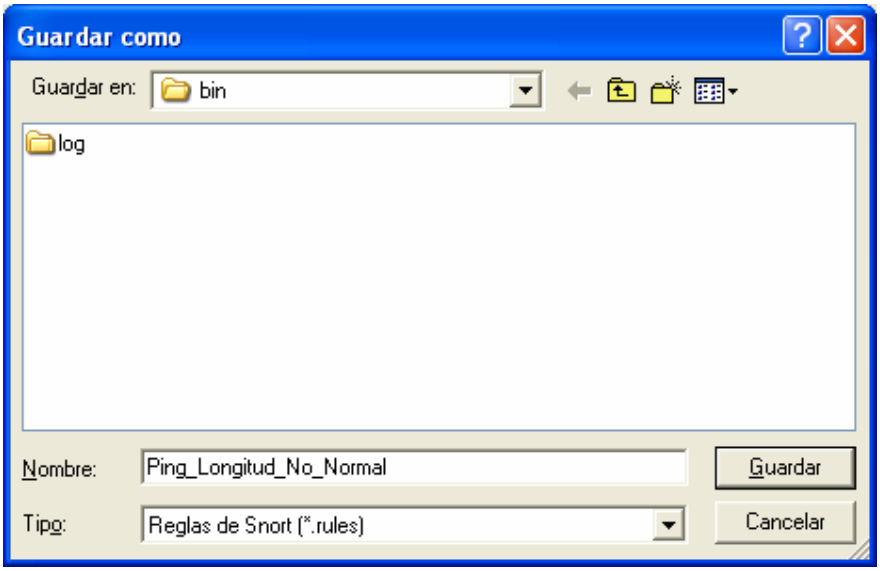

Fig. A3-17 Guardar una regla de Snort

Creada la regla, se muestra un cuadro de mensaje indicando el texto de la regla, para cerciorase de que fue escrita correctamente.

Fig. A3-18 Texto de Regla Creada

Cerciorado el texto de la regla se debe llenar la última parte del modo de funcionamiento IDS como es el nombre del *Log* a ser generado y el número de paquetes que se quieren analizar; al aceptar todos los parámetros escritos y escogidos se crea el archivo de ejecución por lotes para *Snort*, el cual puede ser utilizado posteriormente para otros análisis similares (ver figura A3-19).

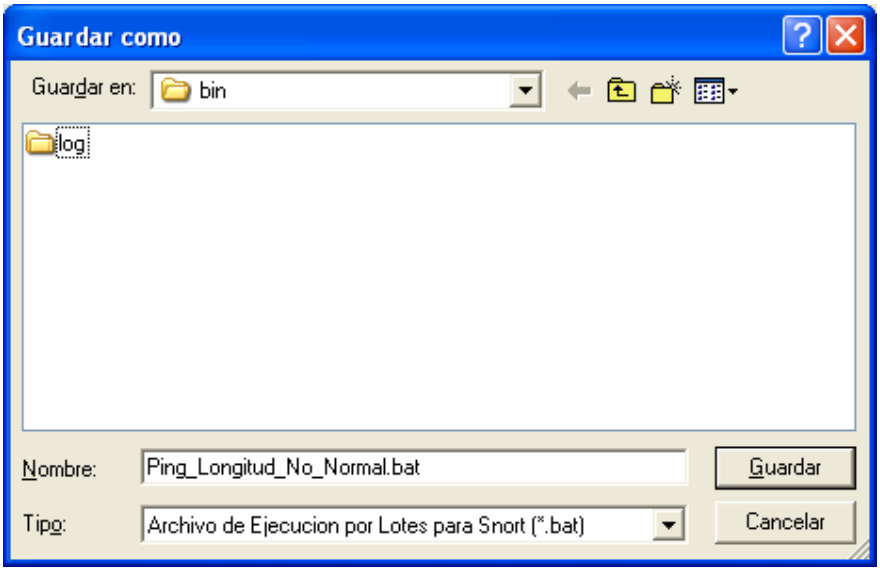

Fig. A3-19 Guardar archivo de ejecución por Lotes para Snort

Después de guardar el archivo de ejecución por lotes y si este no tiene errores el programa llama al símbolo del sistema para la ejecución del archivo generado de *Snort* en DOS. Tal y como se indica a continuación.

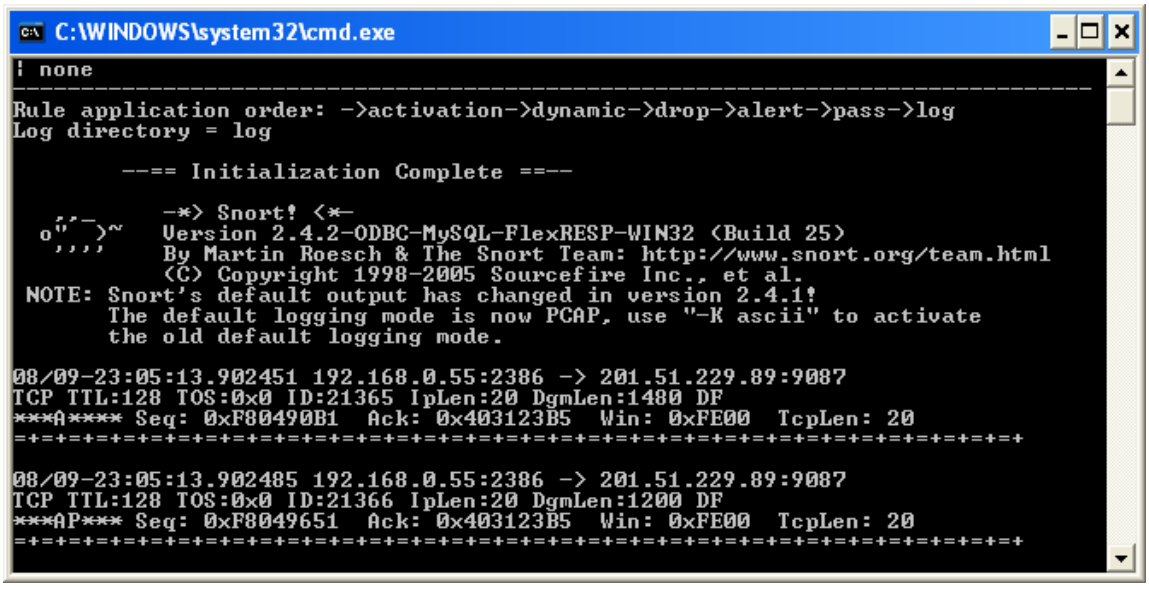

Fig. A3-20 Ejecución de Snort en modo DOS

Por último si se generan alertas estas serán mostradas en el Visor de Sucesos de Windows como se muestra a continuación.

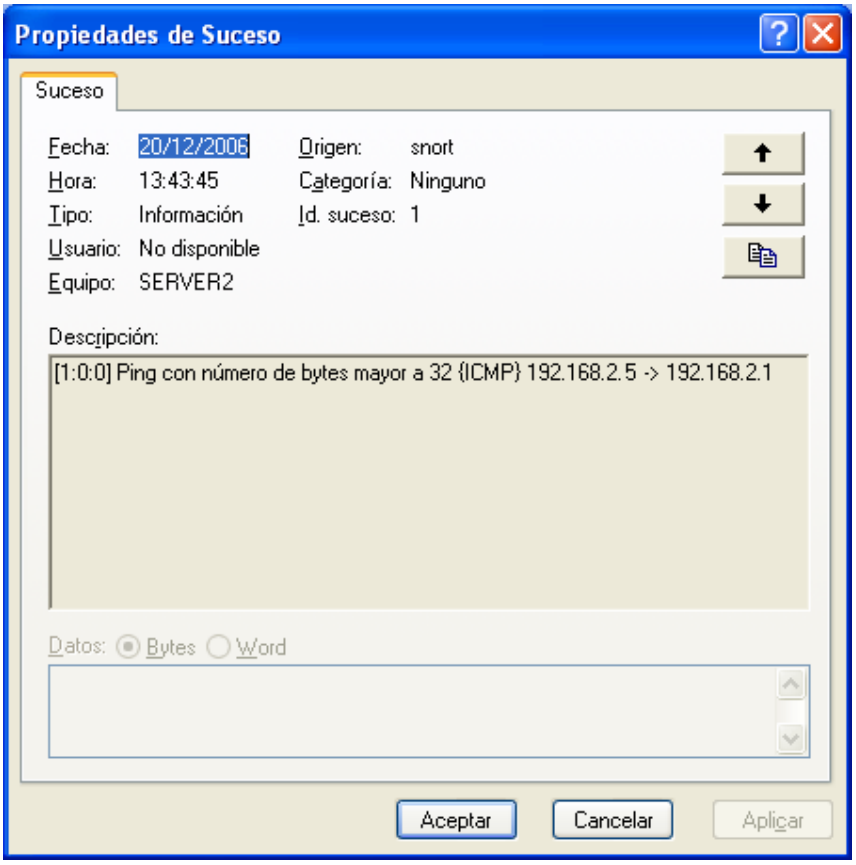

Fig. A3- 21 Alarmas generadas en el Log de Windows

Ejemplo del modo *Sniffer* analizando 100 paquetes y generando el Log con nombre 100_paquetes, mediante el modo del mismo nombre en el sistema de Detección de Intrusos *SNORT*.

En el menú principal se escoge el módulo *IDS*; después se selecciona el modo de funcionamiento como *Sniffer*, en donde se ingresa el número de la interfaz, si no se conoce este número se ocupa el botón correspondiente, en el caso del ejemplo el número es 2, ver figura A3-22.

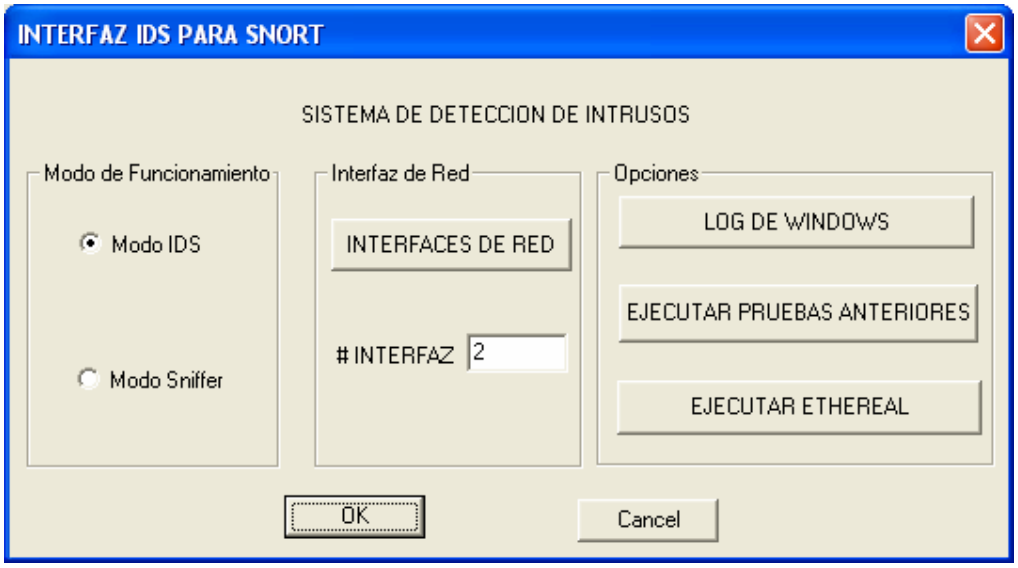

Fig. A3-22 Selección de modo Sniffer

Al poner OK, se presenta la pantalla de la figura A3-23, en la cual se deberá poner el número de paquetes a analizar y el nombre del log a generarse, que para el ejemplo son: 100 y 100_paquetes respectivamente.

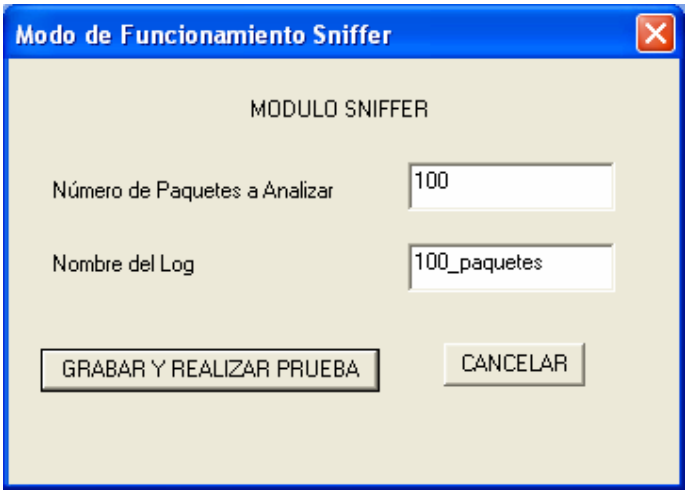

Fig. A3-23 Datos modo Sniffer

Al tener escrito el número de paquetes a analizar y el nombre del log generado lo que se tiene que hacer es presionar el botón Grabar y Realizar Prueba, que automáticamente pide al usuario grabar el archivo de ejecución por lotes (ver figura A3-24).

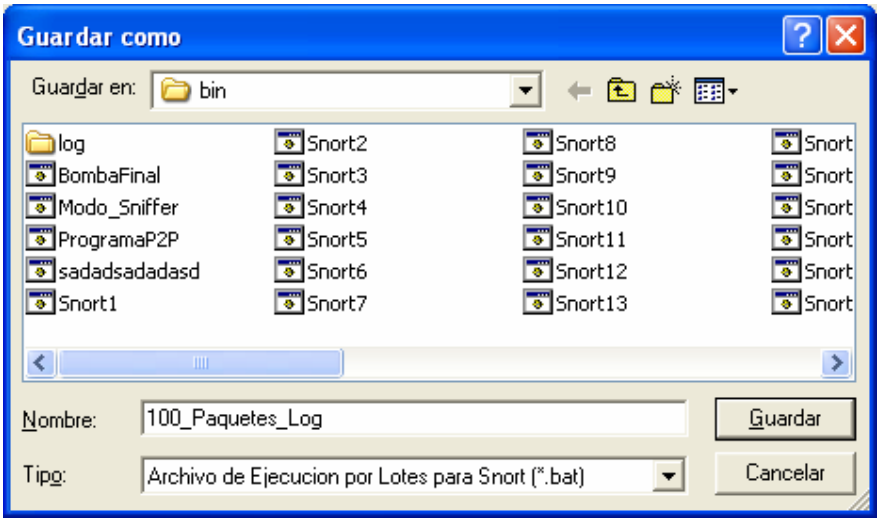

Fig. A3-24 Guardar archivo de ejecución por lotes para Snort

Al grabar el archivo, se ejecutará el sistema *DOS* registrando todo el tráfico generado en la red.

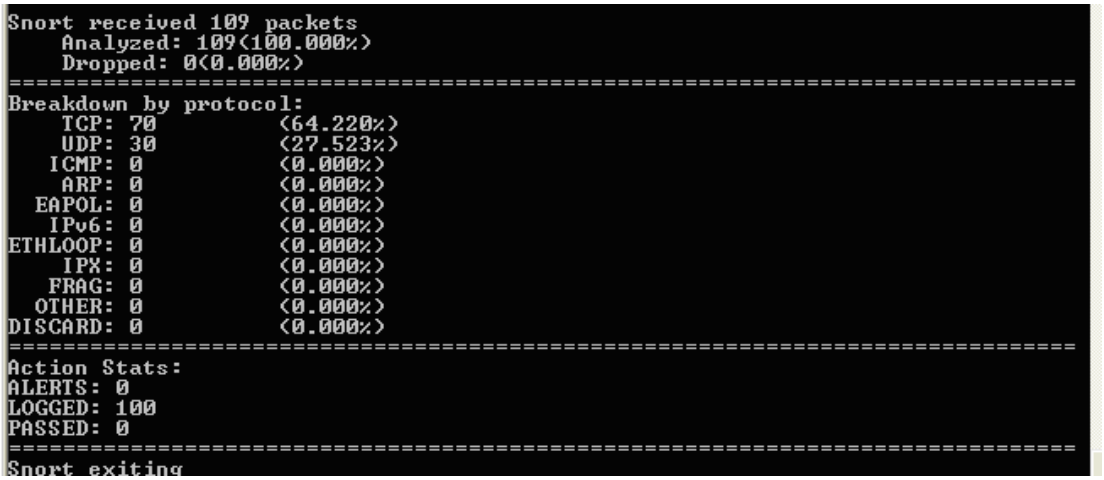

Fig. A3-25 Ejecución del sistema en modo Sniffer

Para la visualización del log generado, lo que se hará es abrir el archivo generado mediante el sistema Ethereal (instalado previamente), que estará grabado en c:\snort\bin\log; para poder abrir directamente el Ethereal se podrá utilizar el botón correspondiente en el menú principal.

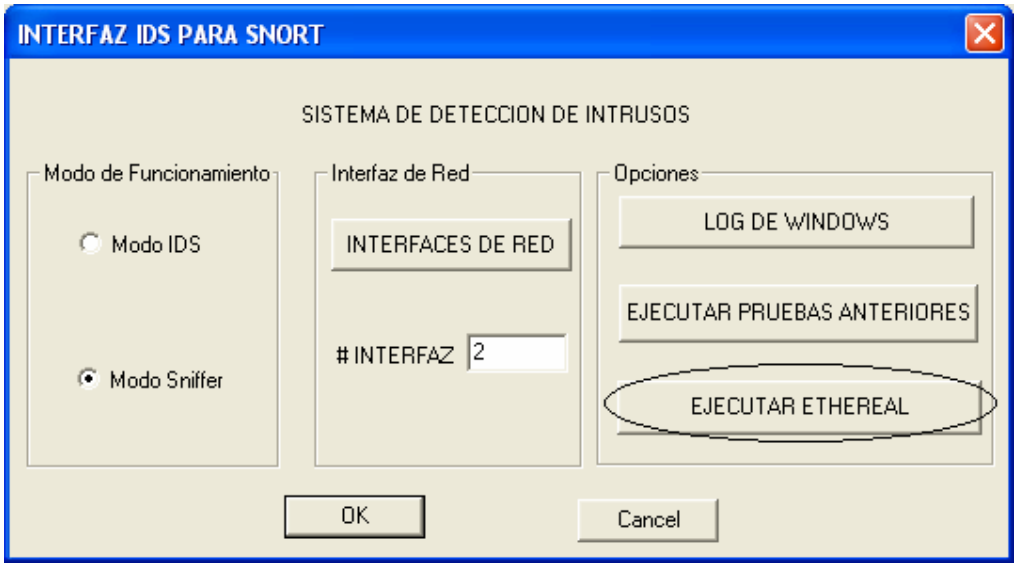

Fig. A3-26 Abrir Ethereal para analizar *logs* generados

Resultados de pruebas realizadas

Hechas las pruebas de la misma manera indicada en el inicio de este anexo, se presentan los resultados que se obtuvieron y que fueron los que se incluyeron en el capítulo correspondiente a las pruebas.

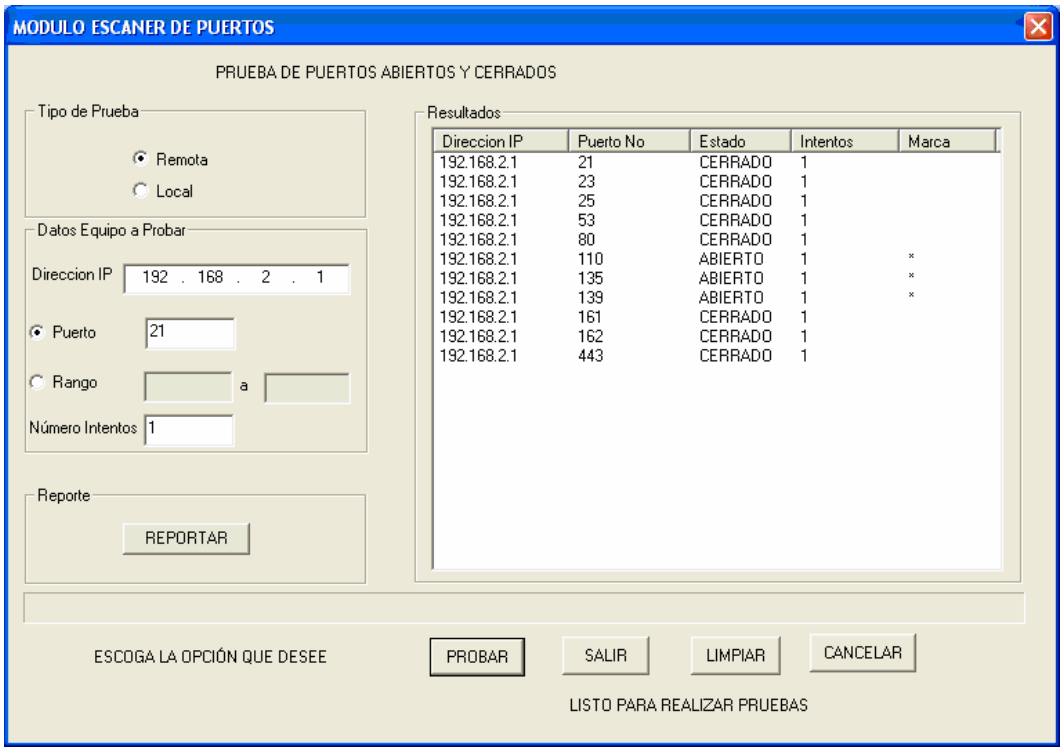

Fig. A3-27 Puertos bien conocidos en el Servidor de páginas *Web*

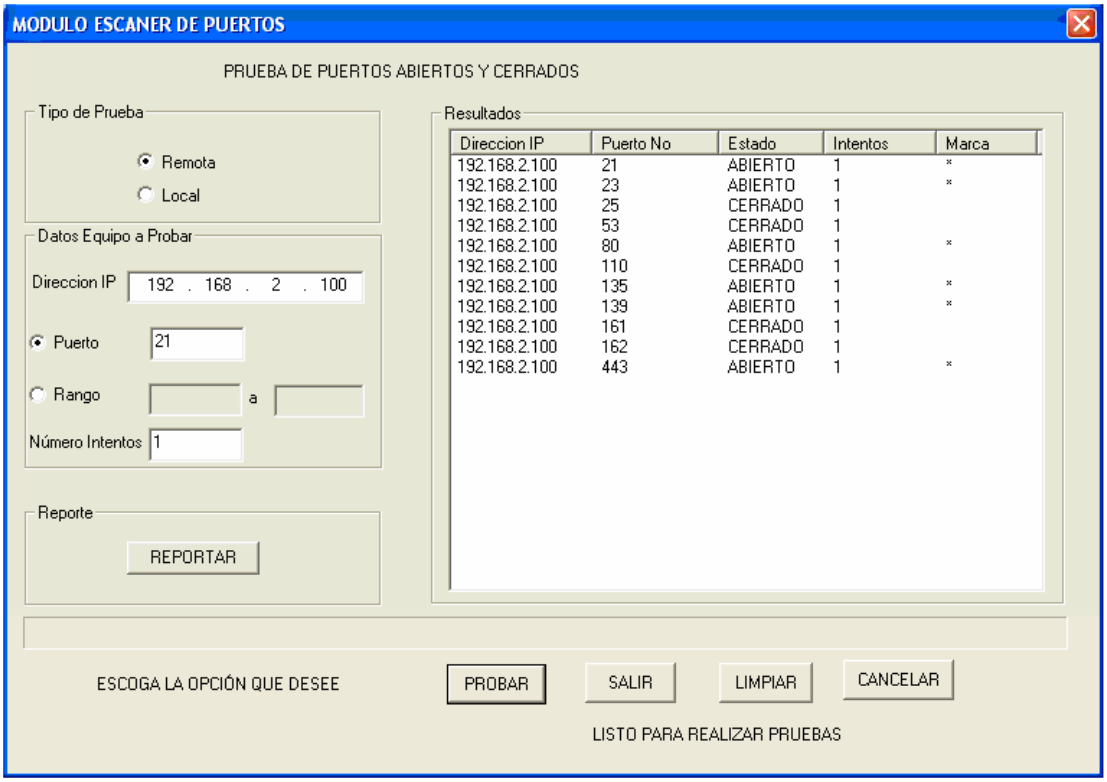

Fig. A3-28 Puertos bien conocidos en el Cliente 1

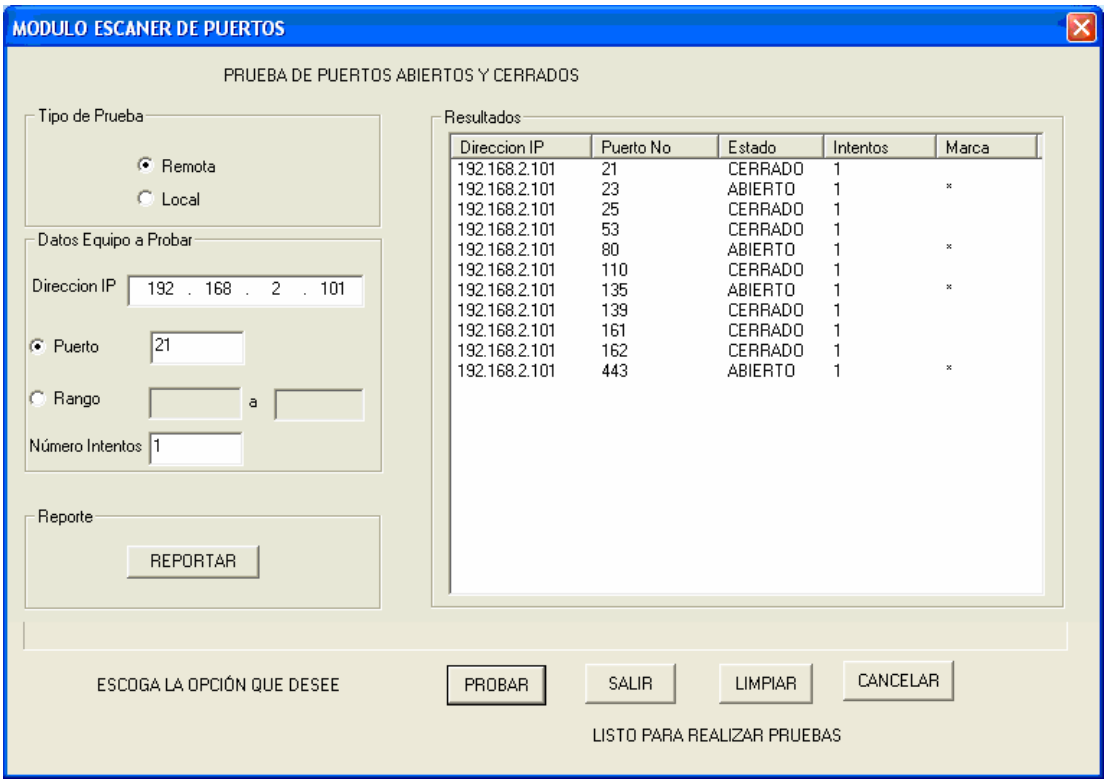

Fig. A3-29 Puertos bien conocidos en el Cliente 2

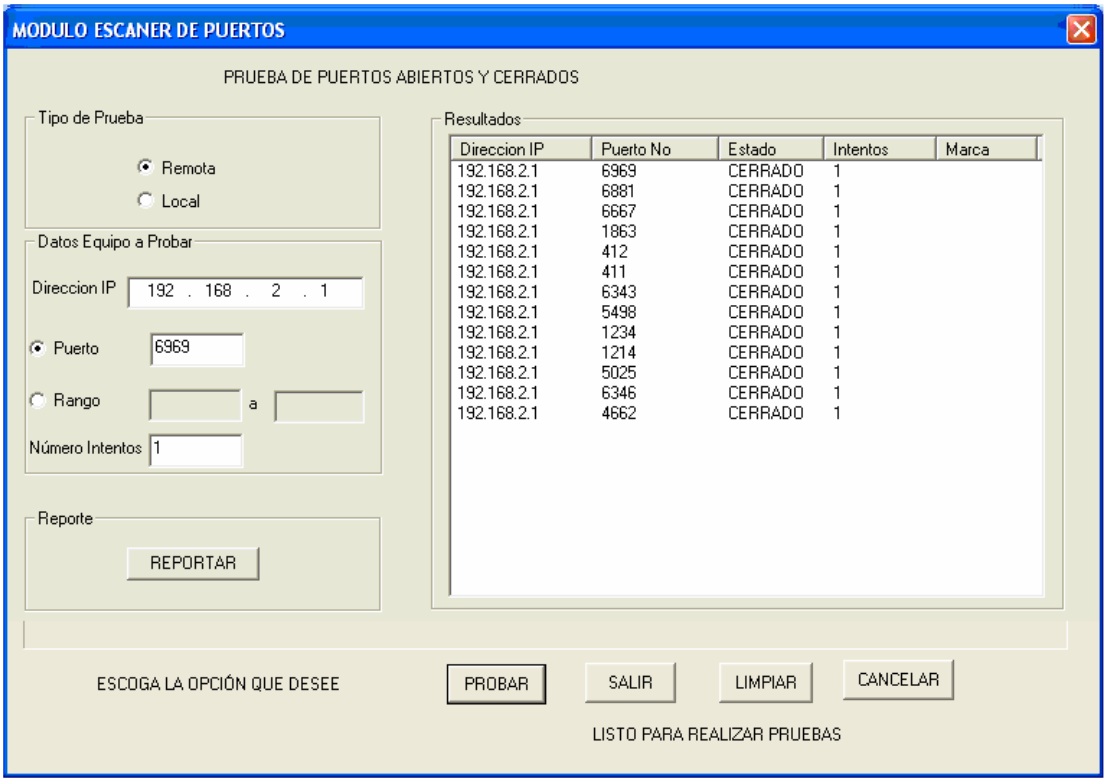

Fig. A3-30 Puertos de aplicaciones prohibidas en el Servidor de páginas *Web*

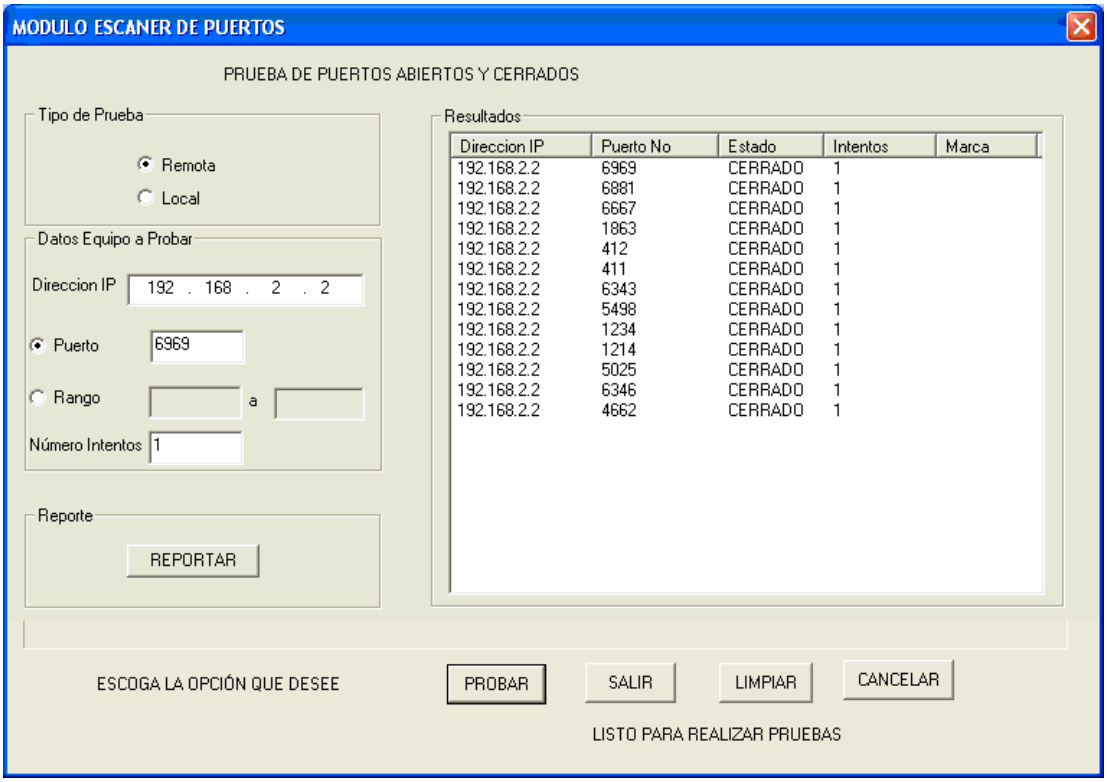

Fig. A3-31 Puertos de aplicaciones prohibidas en el Servidor *Mail*

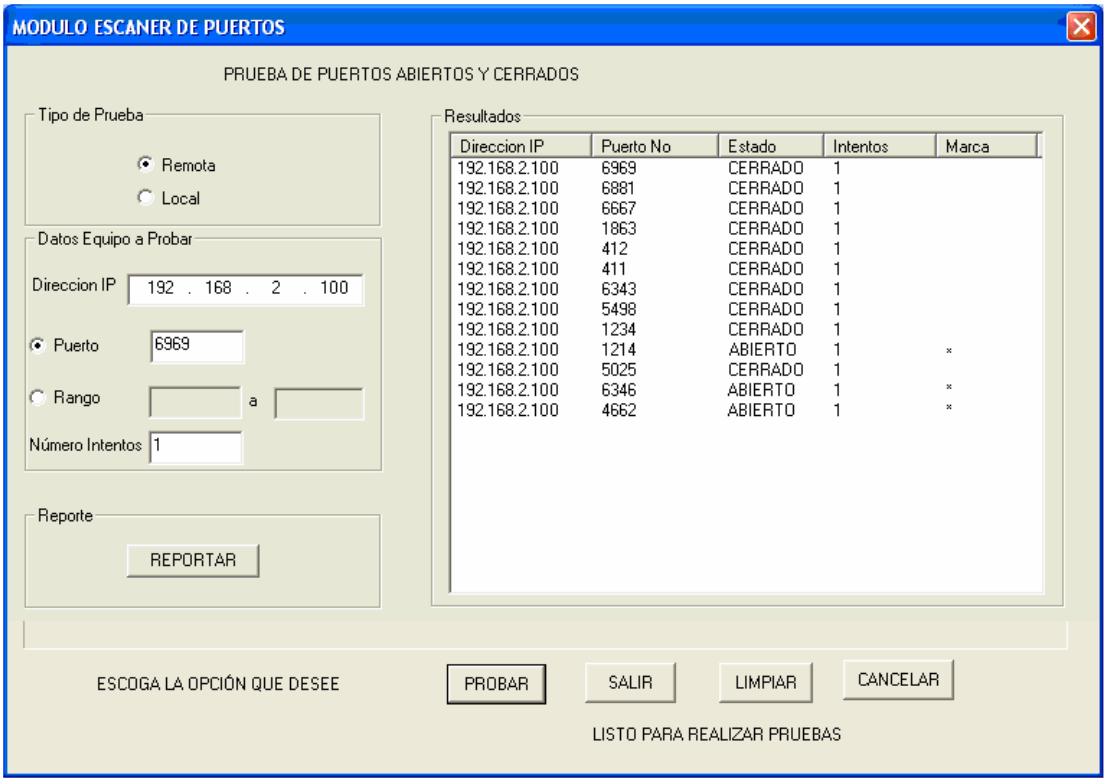

Fig. A3-32 Puertos de aplicaciones prohibidas en el Cliente 1

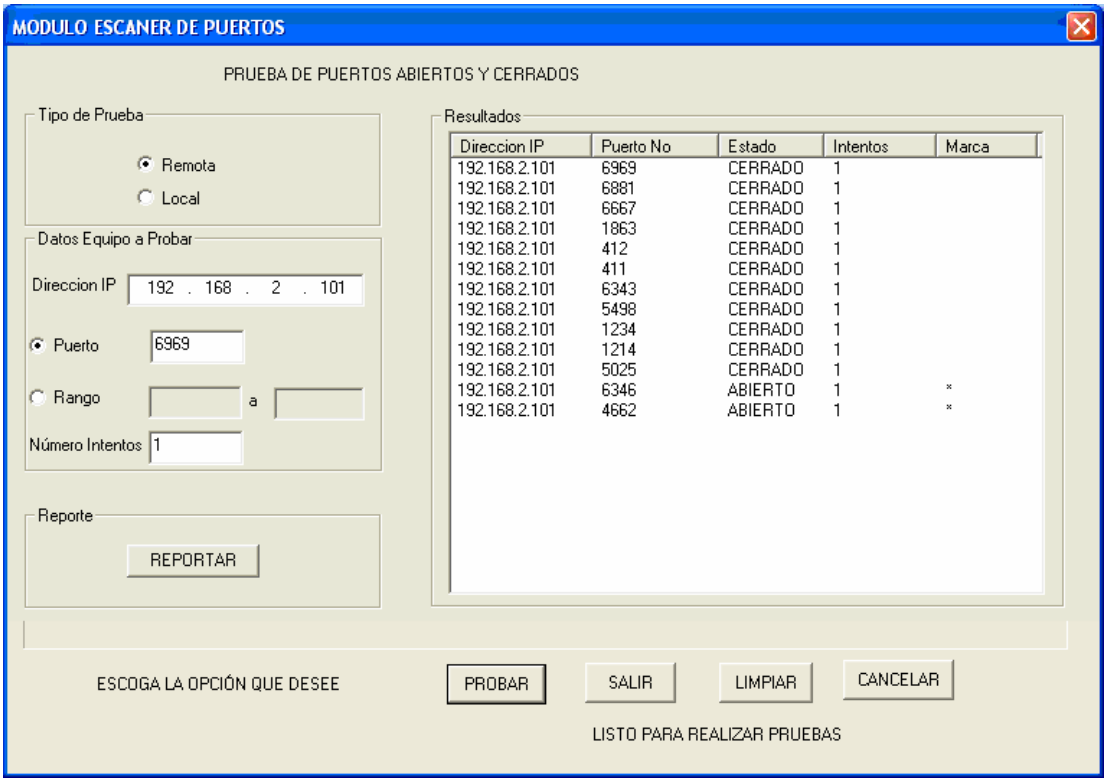

Fig. A3-33 Puertos de aplicaciones prohibidas en el Cliente 2

ANEXO 4

CODIGO FUENTE DE CLASES CREADAS

CreacionReglas.h

```
class CCreacionReglas : public CDialog
{// Construction
public:
	bool m_boolCancelar;
	CString CExplicar(int );
	CString m_strTextoReglaFinal;
	bool m_bMensaje;
	bool m_bEsAvanzada;
	bool m_bBiDireccional;
	int  m_iPosicion2;
	int	m_iPosicion1;
	bool m_bRedDestino;
	bool m_bRedOrigen;
	bool m_bDerecha;
	CString m_strRule;
	CString m_strNombreRegla;
	CString GrabarArchivo(CString buffer);
	CString m_strMensaje;
	CString m_strProtocolo;
	CString m_strTexto;
	CString m_strIpOrigen;
	CString m_strIpDestino;
	CString m_strPuertoOrigen;
	CString m_strPuertoDestino;
	CString m_strTextoRegla;
	CCreacionReglas(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CCreacionReglas)
	enum { IDD = IDD_REGLAS };
	CStatic m_ExplicacionAvanzada;
	CStatic m_Titulo;
	CStatic m_Explicacion;
	CStatic m_TextoOpcion;
	CStatic m_TextoPaquete;
	CStatic m_MensajePaquete;
	CEdit	m_TextoRegla;
	CButton		m_Anadir;
	CListBox		m_ListOpciones;
	CListBox		m_ListClases2;
	CListBox		m_ListClases1;
	CEdit	m_Mensaje;
	CIPAddressCtrl m_IpDestino;
	CIPAddressCtrl m_IpOrigen;
	CEdit	m_PuertoDestino;
	CEdit	m_PuertoOrigen;
	CListBox		m_ListProtocolos;
	CEdit	m_Texto;
	BOOL	m_bTexto;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CCreacionReglas)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL
```

```
// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CCreacionReglas)
	virtual BOOL OnInitDialog();
	virtual void OnOK();
	afx_msg void OnDireccionDer();
	virtual void OnCancel();
	afx_msg void OnBiDireccional();
	afx_msg void OnAnadir();
	afx_msg void OnAvanzadas();
	afx_msg void OnBasicas();
	afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
	afx_msg void OnCancelMode();
	afx_msg void OnEscogerItem();
	afx_msg void OnAceptar();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
```

CreacionReglas.cpp

```
#include "stdafx.h"
#include "AuditoriaRed.h"
#include "CreacionReglas.h"
#include "fcntl.h"
#include "io.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCreacionReglas dialog


CCreacionReglas::CCreacionReglas(CWnd* pParent /*=NULL*/)
	: CDialog(CCreacionReglas::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCreacionReglas)
	//}}AFX_DATA_INIT
}


void CCreacionReglas::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CCreacionReglas)
	DDX_Control(pDX, IDC_AVANZADOEXP, m_ExplicacionAvanzada);
	DDX_Control(pDX, IDC_TITULO, m_Titulo);
	DDX_Control(pDX, IDC_EXPLICACION, m_Explicacion);
	DDX_Control(pDX, IDC_AVANZADA, m_TextoOpcion);
	DDX_Control(pDX, IDC_TEXTOPAQUETE, m_TextoPaquete);
	DDX_Control(pDX, IDC_MENSAJEPAQUETE, m_MensajePaquete);
	DDX_Control(pDX, IDC_EDIT5, m_TextoRegla);
```

```
	DDX_Control(pDX, IDC_BUTTON1, m_Anadir);
	DDX_Control(pDX, IDC_LIST5, m_ListOpciones);
	DDX_Control(pDX, IDC_LIST3, m_ListClases2);
	DDX_Control(pDX, IDC_LIST2, m_ListClases1);
	DDX_Control(pDX, IDC_EDIT4, m_Mensaje);
	DDX_Control(pDX, IDC_IPADDRESS2, m_IpDestino);
	DDX_Control(pDX, IDC_IPADDRESS1, m_IpOrigen);
	DDX_Control(pDX, IDC_EDIT2, m_PuertoDestino);
	DDX_Control(pDX, IDC_EDIT1, m_PuertoOrigen);
	DDX_Control(pDX, IDC_LIST1, m_ListProtocolos);
	DDX_Control(pDX, IDC_EDIT3, m_Texto);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CCreacionReglas, CDialog)
	//{{AFX_MSG_MAP(CCreacionReglas)
	ON_BN_CLICKED(IDC_RADIO2, OnDireccionDer)
	ON_BN_CLICKED(IDC_RADIO1, OnBiDireccional)
	ON_BN_CLICKED(IDC_BUTTON1, OnAnadir)
	ON_BN_CLICKED(IDC_RADIO3, OnAvanzadas)
	ON_BN_CLICKED(IDC_RADIO4, OnBasicas)
	ON_WM_SHOWWINDOW()
	ON_WM_CANCELMODE()
	ON_LBN_SELCHANGE(IDC_LIST5, OnEscogerItem)
//	ON_BN_CLICKED(IDACEPTAR, OnAceptar)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCreacionReglas message handlers

BOOL CCreacionReglas::OnInitDialog()
{

	CDialog::OnInitDialog();
	m_boolCancelar=false;
	m_bEsAvanzada=false;
	m_bBiDireccional=true;
	m_bDerecha=false;
	m_ListOpciones.ShowWindow(false);
	m_Anadir.ShowWindow(false);
	m_TextoRegla.ShowWindow(false);
	m_TextoOpcion.ShowWindow(false);
	m_Explicacion.ShowWindow(false);
	m_ExplicacionAvanzada.ShowWindow(false);
	m_Texto.ShowWindow(true);
	m_TextoPaquete.ShowWindow(true);
	m_MensajePaquete.ShowWindow(true);
	m_Mensaje.ShowWindow(true);
	//Inicializar marcando el Radio Button relacionado con La direccion por defecto <>
	CheckRadioButton(IDC_RADIO1,IDC_RADIO2,IDC_RADIO1);
	//Inicializar marcando el Radio Button relacionado con las opciones báscas por defecto
	CheckRadioButton(IDC_RADIO3,IDC_RADIO4,IDC_RADIO4);
	//Al inicio los cuadros de texto tienen el texto "any"
	m_PuertoOrigen.SetWindowText("any");
	m_PuertoDestino.SetWindowText("any");
	m_ListProtocolos.AddString("IP");
	//Llenar la lista con los protocolos aceptados por SNORT
	m_ListProtocolos.AddString("ICMP");
```

```
	m_ListProtocolos.AddString("TCP");
	m_ListProtocolos.AddString("UDP");
	m_ListProtocolos.SetCurSel(0);
	//Llenar la lista con las posibles mascaras de red
	m_ListClases1.AddString("Dirección IP");
	m_ListClases1.AddString("Clase A /8");
	m_ListClases1.AddString("Clase B /16");
	m_ListClases1.AddString("Clase C /24");
	m_ListClases1.SetCurSel(3);
	//Llenar la lista con las posibles mascaras de red
	m_ListClases2.AddString("Dirección IP");
	m_ListClases2.AddString("Clase A /8");
	m_ListClases2.AddString("Clase B /16");
	m_ListClases2.AddString("Clase C /24");
	m_ListClases2.SetCurSel(3);
	//Llenar la lista con las posibles opciones para la edición de reglas
	m_ListOpciones.AddString("ttl");
	m_ListOpciones.AddString("tos");
	m_ListOpciones.AddString("id");
	m_ListOpciones.AddString("msg");
	m_ListOpciones.AddString("content");
	m_ListOpciones.AddString("ipopts");
	m_ListOpciones.AddString("fragbits");
	m_ListOpciones.AddString("dsize");
	m_ListOpciones.AddString("flags");
	m_ListOpciones.AddString("seq");
	m_ListOpciones.AddString("ack");
	m_ListOpciones.AddString("icmp_id");
	m_ListOpciones.AddString("icmp_seq");
	m_ListOpciones.AddString("offset");
	m_ListOpciones.AddString("depth");
	m_ListOpciones.AddString("nocase");
	m_ListOpciones.AddString("session");
	m_ListOpciones.AddString("rpc");
	m_ListOpciones.AddString("resp");
	m_ListOpciones.AddString("react");
	m_ListOpciones.AddString("reference");
	m_ListOpciones.AddString("sid");
	m_ListOpciones.AddString("rev");
	m_ListOpciones.AddString("priority");
	m_ListOpciones.AddString("uricontent");
	m_ListOpciones.AddString("tag");
	m_ListOpciones.AddString("ip_proto");
	m_ListOpciones.AddString("sameip");
	m_ListOpciones.AddString("byte_test");
	m_ListOpciones.AddString("distance");
	m_ListOpciones.AddString("within");
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

void CCreacionReglas::OnOK()
{
	//Variable temporal para tomar valor retornado por función GrabarArchivo
	CString strTemp;
	//Variable temporal para ingresar caracteres necesarios
	TCHAR temp[10]="";
	//Tomar los datos de los cuadros de texto y Lista del diálogo
	m_PuertoDestino.GetWindowText(m_strPuertoDestino);
	m_PuertoOrigen.GetWindowText(m_strPuertoOrigen);
```

```
m_IpOrigen.GetWindowText(m_strIpOrigen);
m_IpDestino.GetWindowText(m_strIpDestino);
m_Texto.GetWindowText(m_strTexto);
m_Mensaje.GetWindowText(m_strMensaje);
m_iPosicion1=m_ListClases1.GetCurSel();
m_iPosicion2=m_ListClases2.GetCurSel();
m_ListProtocolos.GetText(m_ListProtocolos.GetCurSel(),m_strProtocolo);
//No se aplasto cancelar
m_boolCancelar=false;
//Si no se ingresa dirección IP se mandara por defecto any = cualquiera
if(m_strIpOrigen=="0.0.0.0")
	m_strIpOrigen="any";
if(m_strIpDestino=="0.0.0.0")
	m_strIpDestino="any";
//Textos necesarios para la escritura de la regla
m_strRule+="alert ";
m_strRule+=m_strProtocolo;
m_strRule+=" ";
m_strRule+=m_strIpOrigen;
//LLenar el texto de la regla segun lo escogido en la lista
if(m_iPosicion1==0)
	m_strRule+="/8";
else if(m_iPosicion1==1)
	m_strRule+="/16";
else if(m_iPosicion1==2)
	m_strRule+="/24";
m_strRule+=" ";
m_strRule+=m_strPuertoOrigen;
if(m_bDerecha)
	m_strRule+=" -> ";
else
	m_strRule+=" <> ";
m_strRule+=m_strIpDestino;
//Llenar el texto de la regla segun lo escogido en la lista
if(m_iPosicion2==0)
	m_strRule+="/8";
else if(m_iPosicion2==1)
	m_strRule+="/16";
else if(m_iPosicion2==2)
	m_strRule+="/24";
m_strRule+=" ";
m_strRule+=m_strPuertoDestino;
m_strRule+=" ";
m_strRule+="(";
if(!m_bEsAvanzada)
{
	m_strRule+="content:\" ";
	m_strRule.Insert(m_strRule.GetLength(),temp);
	m_strRule+=m_strTexto;
	m_strRule.Insert(m_strRule.GetLength(),temp);
	m_strRule+="\";";
	m_strRule+="msg:\" ";
	m_strRule.Insert(m_strRule.GetLength(),temp);
	m_strRule+=m_strMensaje;
	m_strRule.Insert(m_strRule.GetLength(),temp);
	m_strRule+="\";)";
}

else
{
```

```
                m_TextoRegla.GetWindowText(m_strTextoReglaFinal);
                if(m_strTextoReglaFinal.GetLength()==0)
                {
                        AfxMessageBox("No se tiene texto para regla");
                        m_strTextoReglaFinal.Empty();
                        m_strRule.Empty();
                        return;
                }
                m_strRule+=m_strTextoReglaFinal;
                m_strRule.Insert(m_strRule.GetLength(),temp);
                m_strRule+=";";
                m_strRule+=")";

        }
        //Se envia a grabar archivo y se almacena el nombre del archivo almacenado
        //para ponerlo en la regla a generar
        strTemp=GrabarArchivo(m_strRule);
        MessageBox(m_strRule);
        m_strRule.Empty();
        CDialog::OnOK();
}


void CCreacionReglas::OnDireccionDer() //Función para saber si la regla es a la Derecha
{
        //Banderas de control para saber sentido, cambiadas en relación a lo seleccionado
        m_bDerecha=true;
        m_bBiDireccional=false;

}

void CCreacionReglas::OnBiDireccional() //Función para saber si la regla es bidireccional
{
        //Banderas de control para saber sentido, cambiadas en relación a lo seleccionado
        m_bDerecha=false;
        m_bBiDireccional=true;
}



void CCreacionReglas::OnCancel()
{
        m_strRule.Empty();
        m_boolCancelar=true;
        CDialog::OnCancel();//Se llama a la función virtual OnCancel();
}




CString CCreacionReglas::GrabarArchivo(CString buffer)
{
        //Se hace un filtro para solo poder ver archivos con extensión *rule
        //Y los nuevos archivos solo se grabaran con esa extensión.
        char strFilter[] = { "Reglas de Snort (*.rules)|*.rules|" };
        //Se crea una clase de dialogo para set utilizada
        CFileDialog* pDlg = new CFileDialog(FALSE, ".rules", NULL, 0, strFilter);
        //Se pone que la carpeta a abrirse este en la direccion C:\Snor\Bin
        pDlg->m_ofn.lpstrInitialDir="C:\\Snort\\bin";
        //El dialogo se creo correctamente?
```

```
	ASSERT(pDlg);
	if (pDlg->DoModal() == IDOK)
	{
		int nHandle,retVal;
		nHandle = _open(pDlg->GetPathName(),_O_BINARY | _O_CREAT |
_O_TRUNC | _O_RDWR);
		if (nHandle == -1)
		{
			MessageBox(_T("NO SE PUEDE SOBREESCRIBIR EL ARCHIVO."),
				_T("Error"),
				MB_OK | MB_ICONEXCLAMATION);
			delete pDlg;
			return;

		}
		//Se toma el nombre del archivo grabado
		m_strNombreRegla=pDlg->GetFileName();
		retVal = _write(nHandle,
			(void*)buffer.GetBuffer(buffer.GetLength()),
			buffer.GetLength());

		if (retVal != buffer.GetLength())
		{
			MessageBox(_T("Ocurrio un error"),
				_T("Error"),MB_OK | MB_ICONEXCLAMATION);
			delete pDlg;
			return;
		}
		buffer.Empty();

	_close(nHandle);
	}
	delete pDlg;
	//Se retorna el nombre del archivo
	return m_strNombreRegla;

}

void CCreacionReglas::OnAnadir()
{
	//Se crea la regla según los textos ingresados
	m_ListOpciones.GetText(m_ListOpciones.GetCurSel(),m_strTextoRegla);
	m_strTextoRegla.Insert(m_strTextoRegla.GetLength(),":");
	m_TextoRegla.GetWindowText(m_strTextoReglaFinal);
	m_strTextoReglaFinal=m_strTextoReglaFinal+m_strTextoRegla;
	m_TextoRegla.SetWindowText(m_strTextoReglaFinal);
}


void CCreacionReglas::OnAvanzadas()
{
	m_bEsAvanzada=true;
	//Muestra los controles avanzados
	m_strRule.Delete(0,m_strRule.GetLength());
	m_ListOpciones.ShowWindow(true);
	m_Anadir.ShowWindow(true);
	m_TextoRegla.ShowWindow(true);
	m_TextoOpcion.ShowWindow(true);
	m_Explicacion.ShowWindow(true);
	m_TextoPaquete.ShowWindow(false);
```

```
	m_MensajePaquete.ShowWindow(false);
	m_Texto.ShowWindow(false);
	m_Mensaje.ShowWindow(false);
	m_ExplicacionAvanzada.ShowWindow(true);
}


void CCreacionReglas::OnBasicas()
{
	m_bEsAvanzada=false;
	//Muestra los controles basicos
	m_strRule.Delete(0,m_strRule.GetLength());
	m_ListOpciones.ShowWindow(false);
	m_Anadir.ShowWindow(false);
	m_TextoRegla.ShowWindow(false);
	m_TextoOpcion.ShowWindow(false);
	m_Explicacion.ShowWindow(false);
	m_Texto.ShowWindow(true);
	m_TextoPaquete.ShowWindow(true);
	m_MensajePaquete.ShowWindow(true);
	m_Mensaje.ShowWindow(true);
	m_ExplicacionAvanzada.ShowWindow(false);
}

void CCreacionReglas::OnShowWindow(BOOL bShow, UINT nStatus)
{
	CDialog::OnShowWindow(bShow, nStatus);
}

void CCreacionReglas::OnCancelMode()
{
	CDialog::OnCancelMode();
}

void CCreacionReglas::OnEscogerItem()
{
	if(m_ListOpciones.GetCurSel())
	{


	}
	m_Explicacion.SetWindowText(CExplicar(m_ListOpciones.GetCurSel()+1));

}

CString CCreacionReglas::CExplicar(int a)
{
	//Solicitar código fuente.
}


ModoIDS.h

class CModoIDS : public CDialog
{
// Construction
public:
	CString m_strNombreLog;
	CString m_strNumeroPaquetes;
	CString m_NumeroTarjeta;
```

```
        CString m_strDireccionRegla;
        CString GrabarArchivo(CString buffer);
        CString m_strDireccion;
        CString m_strNombre;
        CCreacionReglas m_DlgCrearReglas;
        bool m_bBandera4;
        bool m_bBandera3;
        bool m_bBandera2;
        bool m_bBandera1;
        bool m_bReglasArchivos;
        bool m_bPrimera;
        CModoIDS(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
        //{{AFX_DATA(CModoIDS)
        enum { IDD = IDD_MODOIDS };
        CNumeroEdit    m_NumeroPaquetes;
        CEdit    m_Log;
        BOOL   m_bMostrarPayload;
        BOOL   m_bMostrarInfoAplicacion;
        BOOL   m_bMostrarInfoCapaDos;
        BOOL   m_bInalambrico;
        //}}AFX_DATA


// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CModoIDS)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CModoIDS)
        afx_msg void OnImprimirPayload();
        afx_msg void OnMostrarInfCapaAplicacion();
        afx_msg void OnMostrarInformacionCapaDos();
        afx_msg void OnMostrarInformacionInalambrica();
        virtual void OnOK();
        afx_msg void OnCargarReglas();
        afx_msg void OnCrearReglas();
        virtual BOOL OnInitDialog();
        virtual void OnCancel();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

ModoIDS.cpp

```
#include "stdafx.h"
#include "AuditoriaRed.h"
#include "ModoIDS.h"
#include "CreacionReglas.h"
#include "fcntl.h"
#include "io.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CModoIDS dialog


CModoIDS::CModoIDS(CWnd* pParent /*=NULL*/)
	: CDialog(CModoIDS::IDD, pParent)
{
	//{{AFX_DATA_INIT(CModoIDS)
	m_bMostrarPayload = FALSE;		//Inicializacion de las variables a utilizar
	m_bReglasArchivos=true;
	m_bMostrarInfoAplicacion = FALSE;
	m_bMostrarInfoCapaDos = FALSE;
	m_bInalambrico = FALSE;
	//}}AFX_DATA_INIT
}


void CModoIDS::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CModoIDS)
	DDX_Control(pDX, IDC_EDIT2, m_NumeroPaquetes);
	DDX_Control(pDX, IDC_EDIT1, m_Log);
	DDX_Check(pDX, IDC_CHECK1, m_bMostrarPayload);
	DDX_Check(pDX, IDC_CHECK2, m_bMostrarInfoAplicacion);
	DDX_Check(pDX, IDC_CHECK3, m_bMostrarInfoCapaDos);
	DDX_Check(pDX, IDC_CHECK4, m_bInalambrico);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CModoIDS, CDialog)
	//{{AFX_MSG_MAP(CModoIDS)
	ON_BN_CLICKED(IDC_CHECK1, OnImprimirPayload)
	ON_BN_CLICKED(IDC_CHECK2, OnMostrarInfCapaAplicacion)
	ON_BN_CLICKED(IDC_CHECK3, OnMostrarInformacionCapaDos)
	ON_BN_CLICKED(IDC_CHECK4, OnMostrarInformacionInalambrica)
	ON_BN_CLICKED(IDC_BUTTON1, OnCargarReglas)
	ON_BN_CLICKED(IDC_BUTTON2, OnCrearReglas)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CModoIDS message handlers

void CModoIDS::OnImprimirPayload()   //Función para saber si se selecciono o no
{	//La opción de imprimir el Payload
	if(m_bBandera1)		//Estado normal de inicio
	{
		m_bMostrarPayload=true;		//Si se selecciona el check box se pone en
Verdadero
		m_bBandera1=false;		//Se cambia el valor de la bandera
	}
	else
```

```
		m_bMostrarPayload=false;//No se selecciono el check box
}

void CModoIDS::OnMostrarInfCapaAplicacion() //Función para saber si se selecciono para
{
	//Mostrar capa Aplicacion

	if(m_bBandera2)		//Estado normal de inicio
	{
		m_bMostrarInfoAplicacion=true;//Se Seleccionó el check box correspondiente
		m_bBandera2=false;//Se cambia el valor de bandera
	}
	else
		m_bMostrarInfoAplicacion=false;		//No se selecciono el check box
}

void CModoIDS::OnMostrarInformacionCapaDos()//Función para saber si se selecciono para
{
	//Mostrar Informacion de capa Dos
	if(m_bBandera3)		//Estado normal de inicio
	{
		m_bMostrarInfoCapaDos=true;	//Se selecciono el check box correspondiente
		m_bBandera3=false;		//Se cambia el valor de bandera
	}
	else
		m_bMostrarInfoCapaDos=false;//No se selecciono el check box
}

void CModoIDS::OnMostrarInformacionInalambrica()
//Función para mostrar Captar Tramas 802.11
{
	if(m_bBandera4)		//Estado normal de inicio
	{
		m_bInalambrico=true;	//Se selecciono el check box correspondiente
		m_bBandera4=false;	//Se cambia el valor de la bandera
	}
	else
		m_bInalambrico=false;	//No se selecciono el check box

}


void CModoIDS::OnOK()
{
	//Variables necesarias para la generación de archivos
	int Numero;
	CString buffer;
	CString strTemp1;
	CString strTemp2;
	TCHAR temp[10]="\"";
	//Se toma los valores de los cuadros de texto en las variables temporales
	m_NumeroPaquetes.GetWindowText(strTemp1);
	m_Log.GetWindowText(strTemp2);
	//Se comienza escribir el String de texto que va a contener el archivo
	buffer = "cd\\snort\\bin";
	buffer += _T("\r\n\0");
	buffer +="snort -i ";
	buffer +=m_NumeroTarjeta;
	buffer +=" -v";
	//De acuerdo a la opción se escribe o no el archivo de la regla correspondiente
```

```
		if(m_bReglasArchivos)
		{
			buffer += " -c ";
			buffer += m_strNombre;
		}
		else
		{
			buffer+=" -c ";
			buffer+=m_DlgCrearReglas.m_strNombreRegla;
		}
		//Si se quiere mostrar el Payload
		if(m_bMostrarPayload)
			buffer += " -C";
		//Si se quiere mostrar Información de capa aplicación
		else if(m_bMostrarInfoAplicacion)
			buffer += " -d";
		//Si se quiere mostar Información de capa Dos
		else if(m_bMostrarInfoCapaDos)
			buffer += " -e";
		//Si se quiere Captar tramas 802.11 b y g
		else if(m_bInalambrico)
			buffer += " -w";
		buffer+= " -E";
		//Validación, para que no se metan palabras en blanco que son no validas
		if(strTemp2.IsEmpty())
		{
			//Mensaje a mostrarse
			AfxMessageBox("Debe ingresar el nombre del Log a generarse");
			return;
		}

		else if(strTemp1.IsEmpty())
		{
			//Mensaje a mostrarse
			AfxMessageBox("Debe Ingresar el número de paquetes a analizar");
			return;
		}

		//Se sigue editando el texto del archivo a generarse
		buffer +=" -n ";
		buffer +=strTemp1;
		buffer +=" -L ";
		buffer +=strTemp2;
		buffer += _T("\r\n\0");
		buffer +="pause";
		//Dialogo necesario para grabar el archivo de texto, solo con extension *.bat
		char strFilter[] = { "Archivo de Ejecucion por Lotes para Snort (*.bat)|*.bat|" };
		CFileDialog* pDlg = new CFileDialog(FALSE, ".rule", NULL, 0, strFilter);
		pDlg->m_ofn.lpstrInitialDir="C:\\Snort\\Archivos Ejecucion";
		//CDialog::OnOK();
		ASSERT(pDlg);
		if (pDlg->DoModal() == IDOK)
{
int nHandle,retVal;
nHandle = _open(pDlg->GetPathName(),_O_BINARY | _O_CREAT | _O_TRUNC |
_O_RDWR);
			m_strDireccion=pDlg->GetPathName();
			if (nHandle == -1)
			{
				MessageBox(_T("NO SE PUEDE SOBREESCRIBIR EL ARCHIVO."),
```

```
                                _T("Error"),
                                MB_OK | MB_ICONEXCLAMATION);
                        delete pDlg;
                        return;

                }

        retVal = _write(nHandle,
                        (void*)buffer.GetBuffer(buffer.GetLength()),
                        buffer.GetLength());

                if (retVal != buffer.GetLength())
                {
                        MessageBox(_T("Ocurrio un error"),
                                _T("Error"),MB_OK | MB_ICONEXCLAMATION);
                        delete pDlg;
                        return;
                }
                buffer.Empty();

        _close(nHandle);
        }
        delete pDlg;
        //Se acopla al texto generado con caracteres faltantes
        m_strDireccion.Insert(0,temp);
        m_strDireccion.Insert(m_strDireccion.GetLength(),temp);
        //Ejecutamos el archivo generado
        system(m_strDireccion);
        CDialog::OnOK();
}

void CModoIDS::OnCargarReglas()
{
        //Se cargara un archivo ya escrito
        m_bReglasArchivos=true;
        //Filtro para archivo
        char strFilter[] = { "Reglas de Snort (*.rules)|*.rules|" };
        CFileDialog* pDlg = new CFileDialog(TRUE, ".rule", NULL, 0, strFilter);
        //Inicialización de carpeta
        pDlg->m_ofn.lpstrInitialDir="c:\\Snort\\bin";
        //Se llama al dialogo de grabación de archivo de una manera DoModal()
        pDlg->DoModal();
        //Se toma el nombre del archivo generado para incluirlo en el archivo de ejecucion
        m_strNombre=pDlg->GetFileName();
        if(m_strNombre!="")
        {
                m_Log.EnableWindow(true);
                m_NumeroPaquetes.EnableWindow(true);
        }

}

void CModoIDS::OnCrearReglas()
{       //Se creará una nueva regla
        //Variable temporal necesaria para retornar el nombre del archivo generado
        CString strTemp;
        m_Log.EnableWindow(true);
        m_NumeroPaquetes.EnableWindow(true);
        //Se creará una nueva regla
        m_bReglasArchivos=false;
```

```
	//Se llama al dialodo de creacion de reglas de una manera DoModal()
	m_DlgCrearReglas.DoModal();
	if(m_DlgCrearReglas.m_boolCancelar==true)
	{
		m_Log.EnableWindow(false);
		m_NumeroPaquetes.EnableWindow(false);
	}
	else
	{
		m_Log.EnableWindow(true);
		m_NumeroPaquetes.EnableWindow(true);

	}
}

BOOL CModoIDS::OnInitDialog()
{
	m_bPrimera=true;
	//Funcion de inicializacion de dialogo de ModoIDS
	CDialog::OnInitDialog();
	m_bPrimera=false;
	m_Log.EnableWindow(false);
	m_NumeroPaquetes.EnableWindow(false);
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

CString CModoIDS::GrabarArchivo(CString buffer)
{
//Función que graba un archivo y retorna el nombre del archivo generado
//Se repiten los parametros antes mencionados
char strFilter[] = { "Reglas de Snort (*.rules)|*.rules|" };
CFileDialog* pDlg = new CFileDialog(FALSE, ".rules", NULL, 0, strFilter);
pDlg->m_ofn.lpstrInitialDir="C:\\Snort\\bin";
if (pDlg->DoModal() == IDOK)
{
int nHandle,retVal;
nHandle = _open(pDlg->GetPathName(),_O_BINARY | _O_CREAT | _O_TRUNC |
_O_RDWR);
	if (nHandle == -1)
	{
		MessageBox(_T("NO SE PUEDE SOBREESCRIBIR EL ARCHIVO."),
			_T("Error"),
			MB_OK | MB_ICONEXCLAMATION);
		delete pDlg;
		return true;

	}
	m_strDireccionRegla=pDlg->GetFileName();
	retVal = _write(nHandle,
			(void*)buffer.GetBuffer(buffer.GetLength()),
		buffer.GetLength());

	if (retVal != buffer.GetLength())
{
		MessageBox(_T("Ocurrio un error"),
			_T("Error"),MB_OK | MB_ICONEXCLAMATION);
		delete pDlg;
			return true;
		}
```

```
                buffer.Empty();

        _close(nHandle);
        }
        delete pDlg;
        //Retorna el nombre del Archivo generado
        return m_strDireccionRegla;

}

void CModoIDS::OnCancel()
{

        m_bMostrarPayload = FALSE;  //Inicialización de las variables a utilizar
        m_bReglasArchivos=true;
        m_bMostrarInfoAplicacion = FALSE;
        m_bMostrarInfoCapaDos = FALSE;
        m_bInalambrico = FALSE;
        m_Log.EnableWindow(false);
        m_NumeroPaquetes.EnableWindow(false);
        m_Log.SetWindowText("");
        m_NumeroPaquetes.SetWindowText("");
        CDialog::OnCancel();
}
void CModoIDS::OnOK()
{
        //Variables necesarias para la generación de archivos
        int Numero;
        CString buffer;
        CString strTemp1;
        CString strTemp2;
        TCHAR temp[10]="\"";
        //Se toma los valores de los cuadros de texto en las variables temporales
        m_NumeroPaquetes.GetWindowText(strTemp1);
        m_Log.GetWindowText(strTemp2);
        //Se comienza escribir el String de texto que va a contener el archivo
        buffer = "cd\\snort\\bin";
        buffer += _T("\r\n\0");
        buffer +="snort -i ";
        buffer +=m_NumeroTarjeta;
        buffer +=" -v";
        //De acuerdo a la opción se escribe o no el archivo de la regla correspondiente
        if(m_bReglasArchivos)
        {
                buffer += " -c ";
                buffer += m_strNombre;
        }
        else
        {
                buffer+=" -c ";
                buffer+=m_DlgCrearReglas.m_strNombreRegla;
        }
        //Si se quiere mostrar el Payload
        if(m_bMostrarPayload)
                buffer += " -C";
        //Si se quiere mostrar Información de capa aplicación
        else if(m_bMostrarInfoAplicacion)
                buffer += " -d";
        //Si se quiere mostar Información de capa Dos
        else if(m_bMostrarInfoCapaDos)
```

```
		buffer += " -e";
	//Si se quiere Captar tramas 802.11 b y g
	else if(m_bInalambrico)
		buffer += " -w";
	buffer+= " -E";
	//Validación, para que no se metan palabras en blanco que son no validas
	if(strTemp2.IsEmpty())
	{
		//Mensaje a mostrarse
		AfxMessageBox("Debe ingresar el nombre del Log a generarse");
		return;
	}

	else if(strTemp1.IsEmpty())
	{
		//Mensaje a mostrarse
		AfxMessageBox("Debe Ingresar el número de paquetes a analizar");
		return;
	}

	//Se sigue editando el texto del archivo a generarse
	buffer +=" -n ";
	buffer +=strTemp1;
	buffer +=" -L ";
	buffer +=strTemp2;
	buffer += _T("\r\n\0");
	buffer +="pause";
	//Dialogo necesario para grabar el archivo de texto, solo con extension *.bat
	char strFilter[] = { "Archivo de Ejecucion por Lotes para Snort (*.bat)|*.bat|" };
	CFileDialog* pDlg = new CFileDialog(FALSE, ".rule", NULL, 0, strFilter);
	pDlg->m_ofn.lpstrInitialDir="C:\\Snort\\Archivos Ejecucion";
	CDialog::OnOK();
	ASSERT(pDlg);
	if (pDlg->DoModal() == IDOK)
	{
int nHandle,retVal;
nHandle = _open(pDlg->GetPathName(),_O_BINARY | _O_CREAT | _O_TRUNC |
_O_RDWR);
		m_strDireccion=pDlg->GetPathName();
		if (nHandle == -1)
		{
			MessageBox(_T("NO SE PUEDE SOBREESCRIBIR EL ARCHIVO."),
				_T("Error"),
				MB_OK | MB_ICONEXCLAMATION);
			delete pDlg;
			return;

		}

	retVal = _write(nHandle,
			(void*)buffer.GetBuffer(buffer.GetLength()),
			buffer.GetLength());

		if (retVal != buffer.GetLength())
		{
			MessageBox(_T("Ocurrio un error"),
				_T("Error"),MB_OK | MB_ICONEXCLAMATION);
			delete pDlg;
			return;
		}
```

```
                buffer.Empty();

        _close(nHandle);
        }
        delete pDlg;
        //Se acopla al texto generado con caracteres faltantes
        m_strDireccion.Insert(0,temp);
        m_strDireccion.Insert(m_strDireccion.GetLength(),temp);
        //Ejecutamos el archivo generado
        system(m_strDireccion);
}

void CModoIDS::OnCargarReglas()
{
        //Se cargara un archivo ya escrito
        m_bReglasArchivos=true;
        //Filtro para archivo
        char strFilter[] = { "Reglas de Snort (*.rules)|*.rules|" };
        CFileDialog* pDlg = new CFileDialog(TRUE, ".rule", NULL, 0, strFilter);
        //Inicialización de carpeta
        pDlg->m_ofn.lpstrInitialDir="c:\\Snort\\bin";
        //Se llama al dialogo de grabación de archivo de una manera DoModal()
        pDlg->DoModal();
        //Se toma el nombre del archivo generado para incluirlo en el archivo de ejecucion
        m_strNombre=pDlg->GetFileName();
        MessageBox(m_strNombre);
}

void CModoIDS::OnCrearReglas()
{       //Se creará una nueva regla
        //Variable temporal necesaria para retornar el nombre del archivo generado
        CString strTemp;
        //Se creará una nueva regla
        m_bReglasArchivos=false;
        //Se llama al dialodo de creacion de reglas de una manera DoModal()
        m_DlgCrearReglas.DoModal();
}

BOOL CModoIDS::OnInitDialog()
{
        m_bPrimera=true;
        //Funcion de inicializacion de dialogo de ModoIDS
        CDialog::OnInitDialog();
        m_bPrimera=false;
        return TRUE;  // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}

CString CModoIDS::GrabarArchivo(CString buffer)
{
        //Función que graba un archivo y retorna el nombre del archivo generado
        //Se repiten los parametros antes mencionados
        char strFilter[] = { "Reglas de Snort (*.rules)|*.rules|" };
        CFileDialog* pDlg = new CFileDialog(FALSE, ".rules", NULL, 0, strFilter);
        pDlg->m_ofn.lpstrInitialDir="C:\\Snort\\bin";
        if (pDlg->DoModal() == IDOK)
        {
int nHandle,retVal;
nHandle = _open(pDlg->GetPathName(),_O_BINARY | _O_CREAT | _O_TRUNC |
_O_RDWR);
```

```
                    if (nHandle == -1)
                    {
                            MessageBox(_T("NO SE PUEDE SOBREESCRIBIR EL ARCHIVO."),
                                    _T("Error"),
                                    MB_OK | MB_ICONEXCLAMATION);
                            delete pDlg;
                            return true;

                    }
                    m_strDireccionRegla=pDlg->GetFileName();
                    retVal = _write(nHandle,
                            (void*)buffer.GetBuffer(buffer.GetLength()),
                            buffer.GetLength());

                    if (retVal != buffer.GetLength())
                    {
                            MessageBox(_T("Ocurrio un error"),
                                    _T("Error"),MB_OK | MB_ICONEXCLAMATION);
                            delete pDlg;
                            return true;
                    }
                    buffer.Empty();

            _close(nHandle);
            }
            delete pDlg;
            //Retorna el nombre del Archivo generado
            return m_strDireccionRegla;

}

void CModoIDS::OnCancel()
{

            m_bMostrarPayload = FALSE;              //Inicialización de las variables a utilizar
            m_bReglasArchivos=true;
            m_bMostrarInfoAplicacion = FALSE;
            m_bMostrarInfoCapaDos = FALSE;
            m_bInalambrico = FALSE;
            CDialog::OnCancel();
}
```

ModoSniffer.h

```
class CModoSniffer : public CDialog
{
// Construction
public:
	CString m_NumeroInterfaz;
	CString m_strDireccion;
	CModoSniffer(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CModoSniffer)
	enum { IDD = IDD_MODOSNIFFER };
	CEdit    m_NombreLog;
	CNumeroEdit    m_NumeroPaquetes;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CModoSniffer)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CModoSniffer)
	virtual void OnOK();
	virtual void OnCancel();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
```

ModoSniffer.cpp

```
CModoSniffer::CModoSniffer(CWnd* pParent /*=NULL*/)
	: CDialog(CModoSniffer::IDD, pParent)
{
	//{{AFX_DATA_INIT(CModoSniffer)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}


void CModoSniffer::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CModoSniffer)
	DDX_Control(pDX, IDC_EDIT2, m_NombreLog);
	DDX_Control(pDX, IDC_EDIT1, m_NumeroPaquetes);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CModoSniffer, CDialog)
	//{{AFX_MSG_MAP(CModoSniffer)
	//}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()

//////////////////////////////////////////////////////////////////////////
// CModoSniffer message handlers

void CModoSniffer::OnOK()
{

	//Variables String temporales para captar los datos de los cuadros de texto
	CString buffer,strTemp1,strTemp2,strTemp3;
	int Numero;
	//Se toma los datos de los cuadros de texto.
	m_NumeroPaquetes.GetWindowText(strTemp1);
	m_NombreLog.GetWindowText(strTemp2);
	//Analizar si esta lleno el dato de Número de paquetes
	//y si no es así mandar un mensaje de error
	if(strTemp1.IsEmpty())
	{
		AfxMessageBox("Debe Ingresar el número de paquetes a analizar");
		return;
	}
	//Analizar si esta lleno el dato de nombre del Log
	//y si no es así mandar un mensaje de error
	else if(strTemp2.IsEmpty())
	{
		AfxMessageBox("Debe ingresar el nombre del Log");
		return;
	}
	//Filtro para poder desplegar solo archivos con extension *.bat y  grabarlos
	//solo con esa extension
	char strFilter[] = { "Archivo de Ejecucion por Lotes para Snort (*.bat)|*.bat|" };
	//Dialogo para grabar el archivo con las especificaciones anteriores
	CFileDialog* pDlg = new CFileDialog(FALSE, ".rule", NULL, 0, strFilter);
	//El directorio donde se va a abrir va a estar ubivado en C:\Snort\Archivos de Ejecución
	pDlg->m_ofn.lpstrInitialDir="C:\\Snort\\Archivos Ejecucion";
	ASSERT(pDlg);
	if (pDlg->DoModal() == IDOK)
	{
int nHandle,retVal;
nHandle = _open(pDlg->GetPathName(),_O_BINARY | _O_CREAT | _O_TRUNC |
_O_RDWR);
		m_strDireccion=pDlg->GetPathName();
		if (nHandle == -1)
		{
			MessageBox(_T("NO SE PUEDE SOBREESCRIBIR EL ARCHIVO."),
				_T("Error"),
				MB_OK | MB_ICONEXCLAMATION);
			delete pDlg;
			return;

		}

		//Escritura del archivo de texto para ejecución por lotes de Snort
		buffer = "cd\\snort\\bin\\";
		buffer += _T("\r\n\0");
		buffer +="snort -i ";
		buffer +=m_NumeroInterfaz;
		buffer +=" -v ";
		buffer +=" -n ";
		buffer +=strTemp1;
```

```
			buffer +=" -L ";
			buffer +=strTemp2;
			buffer += _T("\r\n\0");
			buffer +="pause";
			retVal = _write(nHandle,
					(void*)buffer.GetBuffer(buffer.GetLength()),
					buffer.GetLength());

				if (retVal != buffer.GetLength())
				{
					MessageBox(_T("Ocurrio un error"),
						_T("Error"),MB_OK | MB_ICONEXCLAMATION);
					delete pDlg;
					return;
				}
				buffer.Empty();

			_close(nHandle);
		}
		delete pDlg;
		TCHAR temp[10]="\"";
		m_strDireccion.Insert(0,temp);
		m_strDireccion.Insert(m_strDireccion.GetLength(),temp);
		system(m_strDireccion);
		CDialog::OnOK();
}

void CModoSniffer::OnCancel()
{

		CDialog::OnCancel();
}ModuloEscaner.h

//Definición del nodo para su uso en pruebas y escritura de archivos
typedef struct
{
		int numIntentos;
		TCHAR DireccionIP[16];
		TCHAR Puerto[5];
		BOOL bEstado; //Cuando es 1 abierto , 0 es cerrado
}NODO;

class CModuloEscaner : public CDialog
{
// Construction
public:
void EscribirDatos(CString buffer);
bool m_bPrimera;
bool m_bBandera;
BOOL PonerItem(int nItem,int nSubItem,LPCTSTR strItem,int nImageIndex=-1);
void PonerTitulos(LPTSTR hdr);
BOOL HacerColumna(LPCTSTR strItem,int nItem,int nSubItem=-1,int nMask=
LVCF_FMT | LVCF_WIDTH | LVCF_TEXT | LVCF_SUBITEM,int nFmt=LVCFMT_LEFT);
bool m_bAbierto;
bool SocketRed(CString m_strIP, int m_Puerto);
CString m_strSoloPuerto;
CString m_strBajoPuerto;
CString m_strAltoPuerto;
CString m_strNumeroIntento;
CString m_strIP;
```

```
CStringList* m_Columnas;
CString m_strTexto;
int m_intPuerto;
int m_intAltoPuerto;
int m_intBajoPuerto;
int m_intNumeroIntentos;
int m_intContadorIntentos;
CPtrList* m_pListaResultados;
CPtrList m_ResultadosFinales;
CPtrList m_prueba;
void OnCaptarDatos();
bool m_bPuerto;
bool m_bLocal;
CModuloEscaner(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
        //{{AFX_DATA(CModuloEscaner)
        enum { IDD = IDD_ESCANER };
        CListCtrl           m_ListCtrlResultados;
        CIPAddressCtrl m_IpAddress;
        CNumeroEdit    m_nIntento;
        CNumeroEdit    m_nPuertoAlto;
        CNumeroEdit    m_nPuertoBajo;
        CNumeroEdit    m_nPuerto;
        BOOL   m_bGenerarReporte;
        //}}AFX_DATA


// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CModuloEscaner)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CModuloEscaner)
        afx_msg void OnEquipoRemoto();
        afx_msg void OnEquipoLocal();
        virtual BOOL OnInitDialog();
        afx_msg void OnUnPuerto();
        afx_msg void OnRangoPuertos();
        afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
        afx_msg void OnLimpiar();
        virtual void OnOK();
        afx_msg void OnReportar();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()

};


ModuloEscaner.h

typedef struct
{
        int numIntentos;
```

```
	TCHAR DireccionIP[16];
	TCHAR Puerto[5];
	TCHAR Protocolo[10];
	BOOL bEstado; //Cuando es 1 abierto , 0 es cerrado
}NODO;

class CModuloEscaner : public CDialog
{
// Construction
public:
	int m_intContador;
	CString CProtocolo(int);
	void EscribirDatos(CString buffer);
	bool m_bPrimera;
	bool m_bBandera;
	BOOL PonerItem(int nItem,int nSubItem,LPCTSTR strItem,int nImageIndex=-1);
	void PonerTitulos(LPTSTR hdr);
	BOOL HacerColumna(LPCTSTR strItem,int nItem,int nSubItem=-1,int nMask=
		LVCF_FMT | LVCF_WIDTH | LVCF_TEXT | LVCF_SUBITEM,int
nFmt=LVCFMT_LEFT);
	bool m_bAbierto;
	bool SocketRed(CString m_strIP, int m_Puerto);
	CString m_strSoloPuerto;
	CString m_strBajoPuerto;
	CString m_strAltoPuerto;
	CString m_strNumeroIntento;
	CString m_strIP;
	CStringList* m_Columnas;
	CString m_strTexto;
	int m_intPuerto;
	int m_intAltoPuerto;
	int m_intBajoPuerto;
	int m_intNumeroIntentos;
	int m_intContadorIntentos;
	CPtrList* m_pListaResultados;
	CPtrList m_ResultadosFinales;
	CPtrList m_prueba;
	void OnCaptarDatos();
	bool m_bPuerto;
	bool m_bLocal;
	CModuloEscaner(CWnd* pParent = NULL);   // standard constructor


// Dialog Data
	//{{AFX_DATA(CModuloEscaner)
	enum { IDD = IDD_ESCANER };
	CButton		m_btnLimpiar;
	CButton		m_btnPararProceso;
	CButton		m_btnOk;
	CStatic m_TextoInformativo;
	CProgressCtrl	m_cProgress;
	CListCtrl		m_ListCtrlResultados;
	CIPAddressCtrl	m_IpAddress;
	CNumeroEdit	m_nIntento;
	CNumeroEdit	m_nPuertoAlto;
	CNumeroEdit	m_nPuertoBajo;
	CNumeroEdit	m_nPuerto;
	BOOL	m_bGenerarReporte;
	//}}AFX_DATA
```

```
// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CModuloEscaner)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:


	// Generated message map functions
	//{{AFX_MSG(CModuloEscaner)
	afx_msg void OnEquipoRemoto();
	afx_msg void OnEquipoLocal();
	virtual BOOL OnInitDialog();
	afx_msg void OnUnPuerto();
	afx_msg void OnRangoPuertos();
	afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
	afx_msg void OnLimpiar();
	virtual void OnOK();
	afx_msg void OnReportar();
	virtual void OnPararProceso();
	afx_msg void OnSalir();
	virtual void OnCancel();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

};
```

ModuloEscaner.cpp

```
#include "stdafx.h"
#include "AuditoriaRed.h"
#include "ModuloEscaner.h"
#include "TheSocket.h"
#include "io.h"
#include "fcntl.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CModuloEscaner dialog


CModuloEscaner::CModuloEscaner(CWnd* pParent /*=NULL*/)
	: CDialog(CModuloEscaner::IDD, pParent)
{
	//Inicialización de variables.
	//{{AFX_DATA_INIT(CModuloEscaner)
	m_bGenerarReporte = FALSE;
	m_bLocal=false;
	m_bPuerto=true;
	m_bBandera=true;
	m_bPrimera=true;
	//}}AFX_DATA_INIT
```

```
}


void CModuloEscaner::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CModuloEscaner)
	DDX_Control(pDX, IDC_LIMPIAR, m_btnLimpiar);
	DDX_Control(pDX, IDC_CANCELAR, m_btnParar Proceso);
	DDX_Control(pDX, IDOK, m_btnOk);
	DDX_Control(pDX, IDC_TEXTO, m_TextoInformativo);
	DDX_Control(pDX, IDC_PROGRESS1, m_cProgress);
	DDX_Control(pDX, IDC_LIST2, m_ListCtrlResultados);
	DDX_Control(pDX, IDC_IPADDRESS1, m_IpAddress);
	DDX_Control(pDX, IDC_EDIT4, m_nIntento);
	DDX_Control(pDX, IDC_EDIT3, m_nPuertoAlto);
	DDX_Control(pDX, IDC_EDIT2, m_nPuertoBajo);
	DDX_Control(pDX, IDC_EDIT1, m_nPuerto);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CModuloEscaner, CDialog)
	//{{AFX_MSG_MAP(CModuloEscaner)
	ON_BN_CLICKED(IDC_RADIO1, OnEquipoRemoto)
	ON_BN_CLICKED(IDC_RADIO2, OnEquipoLocal)
	ON_BN_CLICKED(IDC_RADIO3, OnUnPuerto)
	ON_BN_CLICKED(IDC_RADIO4, OnRangoPuertos)
	ON_WM_SHOWWINDOW()
	ON_BN_CLICKED(IDC_LIMPIAR, OnLimpiar)
	ON_BN_CLICKED(IDC_REPORTAR, OnReportar)
	ON_BN_CLICKED(IDC_BUTTON2, OnPararProceso)
	ON_BN_CLICKED(IDC_SALIR, OnSalir)
	ON_WM_CREATE()
	ON_WM_INITMENU()
	ON_BN_CLICKED(IDC_CANCELAR, OnPararProceso)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CModuloEscaner message handlers
//Función para trabajar en modo Remoto
void CModuloEscaner::OnEquipoRemoto()
{
	m_bLocal=false;//Es Remoto
	m_IpAddress.EnableWindow(true);//Se habilita la escritura
	m_IpAddress.SetWindowText("");
}

//Función para trabajar en modo Local
void CModuloEscaner::OnEquipoLocal()
{
	m_bLocal=true;//Es Local
	m_IpAddress.EnableWindow(false);//Se habilita la escritura
	m_IpAddress.SetWindowText("");
}

//Función para inicialización del Diálogo.
BOOL CModuloEscaner::OnInitDialog()
{
	CDialog::OnInitDialog();
```

```
	//Marcar el RadioButton1
	CheckRadioButton(IDC_RADIO1,IDC_RADIO2,IDC_RADIO1);
	//Marcar el RadioButton3
	CheckRadioButton(IDC_RADIO3,IDC_RADIO4,IDC_RADIO3);
	//Habilitación de cuadros de texto
	m_nPuertoAlto.EnableWindow(false);
	m_nPuertoBajo.EnableWindow(false);
	m_TextoInformativo.SetWindowText("LISTO PARA REALIZAR PRUEBAS");
	//Creación de objetos para visualizar los resultados
	m_pListaResultados= new CPtrList;
	m_Columnas= new CStringList;
	m_nIntento.SetWindowText("1");
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}
//Función para manejar cuando es solo un puerto
void CModuloEscaner::OnUnPuerto()
{
	//Manejo de variables para funcionar solo 1 puerto
	m_bPuerto=true;
	m_nPuertoAlto.EnableWindow(false);
	m_nPuertoBajo.EnableWindow(false);
	m_nPuerto.EnableWindow(true);
	m_nPuertoAlto.SetWindowText("");
	m_nPuertoBajo.SetWindowText("");
}
//Función para manejar un rango de puertos
void CModuloEscaner::OnRangoPuertos()
{
	//Manejo de variables para funcionar en rango de puertos
	m_bPuerto=false;
	m_nPuertoAlto.EnableWindow(true);
	m_nPuertoBajo.EnableWindow(true);
	m_nPuerto.EnableWindow(false);
	m_nPuerto.SetWindowText("");
}
//Función que capta los datos
void CModuloEscaner::OnCaptarDatos()
{
	// Declaración de variables para obtención de datos necesarios
	CString IP;
	BYTE b1,b2,b3,b4;
	TCHAR temp[10]="\0";
	m_cProgress.SetPos(0);
	//Si va trabajar de manera local, ingreso la dirección de Loopback
	//127.0.0.1
	if(m_bLocal)
	{
		b1=127;
		b2=b3=0;
		b4=1;
	}
	else
	{
	//Caso contrario tomo la dirección IP byte a byte y la escribo en
	//un String para Usarlo, lo tomo del cuadro de dialogo.
	m_IpAddress.GetAddress(b1,b2,b3,b4);
	if(b1==b2&&b2==b3&&b3==b4&&b4==0&&b4==0)
	{
		AfxMessageBox("Ingrese Dirección IP");
```

```
		return;
}
else if(b4==255||b1==127)
		AfxMessageBox("Dirección IP no Lógica");
}
//Transformo de tipo Byte a Char para utilizarlo después como string
IP=_itoa(b1,temp,10);
IP+=_T('.');
IP+=_itoa(b2,temp,10);
IP+=_T('.');
IP+=_itoa(b3,temp,10);
IP+=_T('.');
IP+=_itoa(b4,temp,10);

//Igualo el valor de la dirección IP
m_strIP=IP;
//Toma de los datos de cuadro de texto a variables
m_nPuerto.GetWindowText(m_strSoloPuerto);
m_nPuertoAlto.GetWindowText(m_strAltoPuerto);
m_nPuertoBajo.GetWindowText(m_strBajoPuerto);
m_nIntento.GetWindowText(m_strNumeroIntento);
//Pasar los datos a Strings para su manejo
m_intPuerto=atoi(m_strSoloPuerto);
m_intAltoPuerto=atoi(m_strAltoPuerto);
m_intBajoPuerto=atoi(m_strBajoPuerto);
m_intNumeroIntentos=atoi(m_strNumeroIntento);
if(m_intAltoPuerto >65535)
{
		AfxMessageBox("El valor de los puertos pueden ir de 1 a 65535");
			return;
}
m_cProgress.SetRange32(0,m_intAltoPuerto - m_intBajoPuerto+1);
m_cProgress.SetStep(1);
//Poner los resultados en la lista
POSITION p = m_pListaResultados->GetHeadPosition();
while (p)
{
		POSITION temp = p;
		NODO* pNodo = (NODO*)m_pListaResultados->GetNext(p);
		m_pListaResultados->RemoveAt(temp);
		if (pNodo)
			delete pNodo;
}
//Si solo es un puerto
if(m_bPuerto)
{
		if(m_strSoloPuerto=="")
		{
			AfxMessageBox("Ingrese Puerto");
			return;
		}
		//Inicialización del contador de intentos
		//m_intContadorIntentos=1;
		UINT nAttempt = 1;
		//El contador es igual al valor ingresado en el diálogo
		m_intContador=m_intPuerto;
		//Hacer la operación según el número de intentos
		while(nAttempt <= m_intNumeroIntentos && !m_bAbierto)
		{
			//Envio de los parámetros a la función analizar y el resultado
```

```
                    //devuelto igualarlo a la variable y controlar el estado del puerto
                    m_bAbierto=SocketRed(m_strIP,m_intPuerto);
                    //Si se encuentra abierto
                    if (m_bAbierto)
                    {
                        //Se crea un nuevo nodo (Datos para la lista)
                        NODO* pNodo = new NODO;
                        //Se creo bien el nodo?
                        ASSERT(pNodo);
                        //Se lena los valores en los datos del nodo
                        strcpy(pNodo->DireccionIP,IP.GetBuffer(IP.GetLength()));
                        strcpy(pNodo->Puerto,_itoa(m_intContador,temp,10));
                        pNodo->bEstado = 1; //El Puerto esta abierto
                        pNodo->numIntentos = nAttempt;
                        strcpy(pNodo->Protocolo,CProtocolo(m_intContador));
                        m_pListaResultados->AddTail(pNodo);//Se añade el nodo a la
lista
                    }
                    nAttempt++;//Siguiente intento
                }
                //Si no se encuentra abierto
                if (!m_bAbierto)
                {
                    //Se hace el mismo procedimiento anterior
                    //Solo cambia el estado a 0 (Cerrado)
                    NODO* pNodo = new NODO;
                    ASSERT(pNodo);
                    strcpy(pNodo->DireccionIP,IP.GetBuffer(IP.GetLength()));
                    strcpy(pNodo->Puerto,_itoa(m_intContador,temp,10));
                    pNodo->bEstado = 0; //El puerto esta cerrado
                    pNodo->numIntentos = nAttempt-1;
                    strcpy(pNodo->Protocolo,CProtocolo(m_intContador));
                    m_pListaResultados->AddTail(pNodo);

                }
                int nIndex = 0;//Inicialización de la posición de la lista gráfica
                //Escogemos la posición en la cual nos encontramos
                POSITION pos = m_pListaResultados->GetHeadPosition();
                while (pos)
                {
                    //Casting para pasar de un pNodo a m_pListaResultados
                    NODO* pNodo = (NODO*)m_pListaResultados->GetNext(pos);
                    //Llenamos la lista según la posición en la que nos encontramos
                    PonerItem(nIndex,0,pNodo->DireccionIP);
                    PonerItem(nIndex,1,pNodo->Puerto);
                    PonerItem(nIndex,5,pNodo->Protocolo);
                    if (pNodo->bEstado)
                    {
                        PonerItem(nIndex,2,_T("ABIERTO"));
                        PonerItem(nIndex,4,_T("*"));
                    }
                    else
                    {
                        PonerItem(nIndex,2,_T("CERRADO"));
                        PonerItem(nIndex,4,_T(" "));
                    }
                    PonerItem(nIndex++,3,_itoa(pNodo->numIntentos,temp,10));
                }
        m_bAbierto=false;
        }
```

```
	//Si es un rango de puertos
	else
	{
		if(m_intAltoPuerto<=m_intBajoPuerto)
		{
			AfxMessageBox("Error en rango de puertos");
			return;
		}

		//Se repite el proceso anterior mencionado sino que una vez por cada
		//valor de puerto desde el puerto Inferior ingresado hasta el puerto superior.
	for (m_intContador = m_intBajoPuerto; m_intContador <= m_intAltoPuerto;
m_intContador++)
		{

			m_bAbierto=false;
			UINT nAttempt=1;
			while(nAttempt <=m_intNumeroIntentos && !m_bAbierto)
			{
				TCHAR temp1[10]="\0";
				CString str = _T("Probando el Puerto# ");
				str += itoa(m_intContador,temp1,10);
				str += _T(", IP=");
				str += IP;
				str += _T(", Intento=");
				//str += itoa(m_intNumeroIntentos,temp1,10);
				str += itoa(nAttempt,temp1,10);
				m_TextoInformativo.SetWindowText(str);
				str.Empty();
				m_bAbierto=SocketRed(m_strIP,m_intContador);

				if (m_bAbierto)
				{
					NODO* pNodo = new NODO;
					ASSERT(pNodo);
					strcpy(pNodo-
>DireccionIP,IP.GetBuffer(IP.GetLength()));
					strcpy(pNodo->Puerto,_itoa(m_intContador,temp,10));
					strcpy(pNodo->Protocolo,CProtocolo(m_intContador));
					pNodo->bEstado = 1; //El Puerto esta abierto
					pNodo->numIntentos = nAttempt;
					m_pListaResultados->AddTail(pNodo);
				}
				nAttempt++;
			}
			if (!m_bAbierto)
			{
				NODO* pNodo = new NODO;
				ASSERT(pNodo);
				strcpy(pNodo->DireccionIP,IP.GetBuffer(IP.GetLength()));
				strcpy(pNodo->Puerto,_itoa(m_intContador,temp,10));
				strcpy(pNodo->Protocolo,CProtocolo(m_intContador));
				pNodo->bEstado = 0; //El puerto esta cerrado
				pNodo->numIntentos = nAttempt-1;
				m_pListaResultados->AddTail(pNodo);
			}

			MSG message;
			if (::PeekMessage(&message,NULL,0,0,PM_REMOVE))
			{
```

```
                    ::TranslateMessage(&message);
                    ::DispatchMessage(&message);
                }

                m_cProgress.StepIt();
            }
            int nIndex = 0;


                POSITION pos = m_pListaResultados->GetHeadPosition();
                while (pos)
                {
                    NODO* pNodo = (NODO*)m_pListaResultados->GetNext(pos);
                    PonerItem(nIndex,0,pNodo->DireccionIP);
                    PonerItem(nIndex,1,pNodo->Puerto);
                    PonerItem(nIndex,5,pNodo->Protocolo);
                    if (pNodo->bEstado)
                        {
                        PonerItem(nIndex,2,_T("ABIERTO"));
                        PonerItem(nIndex,4,_T("*"));
                        }
                    else
                    {
                            PonerItem(nIndex,2,_T("CERRADO"));
                            PonerItem(nIndex,4,_T(" "));
                    }
                        PonerItem(nIndex++,3,_itoa(pNodo-
>numIntentos,temp,10));
                }
        }



POSITION pos =      m_pListaResultados->GetHeadPosition();
while (pos)
        {
                NODO* pNodo = (NODO*)       m_pListaResultados->GetNext(pos);
                m_strTexto +=_T("\r\n\0");
                m_strTexto += pNodo->DireccionIP;
                m_strTexto += _T("\t");
                m_strTexto += pNodo->Puerto;
                m_strTexto += _T("\t");
                if (pNodo->bEstado)
                m_strTexto+= _T("ABIERTO");
                else
                m_strTexto += _T("CERRADO");
        }
}



//Función para analizar el estado del puerto, esta función retorna
//una bandera según este valor sabemos si es abierto o cerrado
bool CModuloEscaner::SocketRed(CString m_strIP, int m_intPuerto)
{
        //Creamos un nuevo socket para el proceso de comprobación
        CTheSocket* pSocket;
        pSocket = new CTheSocket;
        //Se creo bien el Socket?
        ASSERT(pSocket);
```

```
	//Si no se puede crear el Socket
	if (!pSocket->Create())
	{
		delete pSocket;
		pSocket = NULL;
		return FALSE;//No se pudo hacer conprobación
	}
		while (!pSocket->Connect(m_strIP , m_intPuerto))
	{
		//No se pudo conectar el scoket
		delete pSocket;
		pSocket = NULL;
		return FALSE;//Estado del puerto es Cerrado

	}
	//Si se creo el socket
	pSocket->Close();//Lo cerramos
	delete pSocket;//Lo borramos
	return TRUE;//Hay conexión entonces puerto Abierto

}

//Funciones para manejo gráfico de la lista para ingreso de datos
BOOL CModuloEscaner::HacerColumna(LPCTSTR strItem, int nItem, int nSubItem,
                                                                    int nMask, int nFmt)
{
	//Creación de las los campos para poner lo datos
	LV_COLUMN lvc;
	lvc.mask = nMask;
	lvc.fmt = nFmt;
	lvc.pszText = (LPTSTR) strItem;
	// Separación de cada campo en hoja de resultados
	lvc.cx = m_ListCtrlResultados.GetStringWidth(lvc.pszText)+35 ;
	if(nMask & LVCF_SUBITEM)
	{
		if(nSubItem != -1)
			lvc.iSubItem = nSubItem;
		else
			lvc.iSubItem = nItem;
	}
	return m_ListCtrlResultados.InsertColumn(nItem,&lvc);

}

//Función para poner los títulos en las columnas de la lista
void CModuloEscaner::PonerTitulos(LPTSTR hdr)
{
	if (m_Columnas)
		m_Columnas->AddTail(hdr);
}


BOOL CModuloEscaner::PonerItem(int nItem, int nSubItem, LPCTSTR strItem, int
nImageIndex)
{
	//Llenar los datos en la lista del dialogo de resultados
	LV_ITEM lvItem;
	lvItem.mask = LVIF_TEXT;
	lvItem.iItem = nItem;
	lvItem.iSubItem = nSubItem;
```

```
	lvItem.pszText = (LPTSTR) strItem;
	if(nImageIndex != -1)
	{
		lvItem.mask |= LVIF_IMAGE;
		lvItem.iImage |= LVIF_IMAGE;
	}
	if(nSubItem == 0)
		return m_ListCtrlResultados.InsertItem(&lvItem);
	return m_ListCtrlResultados.SetItem(&lvItem);

}


void CModuloEscaner::OnShowWindow(BOOL bShow, UINT nStatus)
{

	//Poner título a los nombres de cada cuadro
	CDialog::OnShowWindow(bShow, nStatus);
	PonerTitulos(_T("Direccion IP"));
	PonerTitulos(_T("Puerto No"));
	PonerTitulos(_T("Estado"));
	PonerTitulos(_T("Intentos"));
	PonerTitulos(_T("Marca"));
	PonerTitulos(_T("Protocolo"));
	//Se comienza de la posicion 0
	int nIndex = 0;
	//Cogemos la posición en la cual esta señalandi
	POSITION pos = m_Columnas->GetHeadPosition();
	while (pos)
	{
		CString hdr = (CString)m_Columnas->GetNext(pos);
		HacerColumna(hdr,nIndex++,1);
	}
}

//Función para limpiar los datos de la lista
void CModuloEscaner::OnLimpiar()
{

	m_pListaResultados->RemoveAll();
	m_ListCtrlResultados.DeleteAllItems();
	m_strTexto.Empty();
}

//Función para realizar todo el análisis
void CModuloEscaner::OnOK()
{
	//Se capta los datos para realizar las pruebas
	OnCaptarDatos();

}

void CModuloEscaner::OnReportar()
{
	//Variables para definir la fecha y hora de la prueba
	char d[10],m[10],a[10],h[10],min[10];
	COleDateTime time;
	CString fecha, horas, str_dia, str_mes, str_anio, str_hora, str_minuto, cabecera;
	cabecera= "\nDirección IP:\tPuerto:\tEstado:\n";
	int dia, mes, anio, hora, minuto;
```

```
	time=COleDateTime::GetCurrentTime();
	dia=time.GetDay();
	mes=time.GetMonth();
	anio=time.GetYear();
	hora=time.GetHour();
	minuto=time.GetMinute();
	itoa(dia,d,10);
	itoa(mes,m,10);
	itoa(anio,a,10);
	itoa(hora,h,10);
	itoa(minuto,min,10);
	//Se toman los valores String de los datos de hora y fecha
	str_dia=d;
	str_mes=m;
	str_anio=a;
	str_hora=h;
	str_minuto=min;
	if(str_minuto.GetLength()==1)
		str_minuto.Insert(0,"0");
	//Se escribe el valor de String a imprimirse en el archivo
	fecha += "FECHA:";
	fecha += _T("\t");
	fecha += str_dia;
	fecha += _T("/");
	fecha += str_mes;
	fecha += _T("/");
	fecha += str_anio;
	horas +="HORA:";
	horas += _T("\t");
	horas += str_hora;
	horas += _T(":");
	horas += str_minuto;
	if(	m_pListaResultados->IsEmpty())
	{
		AfxMessageBox("No se tienen datos todavia");
		return;
	}
	//texto que se escribirá en los archivos de reporte generados
	//De acuerdo al tipo de Puertos sea un solo puerto o un rango
	CString buffer;//Variable para escritura de archivo
	buffer = "REPORTE DE DATOS OBTENIDOS POR EL ESCANER DE PUERTOS\n";
	buffer += _T("\r\n\0");
	buffer += fecha;
	buffer += _T("\r\n\0");
	buffer += horas;
	buffer += _T("\r\n\0");
	buffer += cabecera;
	buffer +=m_strTexto;
	buffer += _T("\r\n\0");
	buffer += _T("\r\n\0");
	buffer += "Realizado por Santiago Oñate";
	buffer += _T("\r\n\0");
	buffer += _T("\r\n\0");
	buffer += "Escuela Politécnica Nacional";
	buffer += _T("\r\n\0");
	buffer += _T("\r\n\0");
	EscribirDatos(buffer);
}

void CModuloEscaner::EscribirDatos(CString buffer)
```

```
{
	//Filtro para crear un diálogo de Grabar archivos, en formato (.txt)
	char strFilter[] = { "Archivo de Reporte de Puertos (*.xls)|*.xls|" };
	CFileDialog* pDlg = new CFileDialog(FALSE, ".xls", NULL, 0, strFilter);
	pDlg->m_ofn.lpstrInitialDir="C:\\";
	CDialog::OnOK();
	ASSERT(pDlg);//Se creo bien el diálogo
	if (pDlg->DoModal() == IDOK)
	{
int nHandle,retVal;
nHandle = _open(pDlg->GetPathName(),_O_BINARY | _O_CREAT | _O_TRUNC |
_O_RDWR);
		if (nHandle == -1)
		{
			MessageBox(_T("NO SE PUEDE SOBREESCRIBIR EL ARCHIVO."),
				_T("Error"),
				MB_OK | MB_ICONEXCLAMATION);
			delete pDlg;
			return;
		}
		//Función para escribir
		retVal = _write(nHandle,
		(void*)buffer.GetBuffer(buffer.GetLength()),
		buffer.GetLength());
		if (retVal != buffer.GetLength())
		{
			MessageBox(_T("Ocurrio un error mientras se escribian los datos"),
			_T("Error"),MB_OK | MB_ICONEXCLAMATION);
			delete pDlg;//Borramos el Diálogo
			return;
		}
	buffer.Empty();
	_close(nHandle);
	delete pDlg;

	}
}

void CModuloEscaner::OnPararProceso()
{
	m_intContador = m_intAltoPuerto+1;
	m_TextoInformativo.SetWindowText("LISTO PARA REALIZAR PRUEBAS");
}

void CModuloEscaner::OnSalir()
{
	CDialog::OnCancel();
}

void CModuloEscaner::OnCancel()
{
	//Función para Salir de la aplicación
	CDialog::OnCancel();
}

CString CModuloEscaner::CProtocolo(int contador)
{
	switch(contador)
	{
		case 5:
```

```
            return "RJE";
            break;
            case 18:
            return   "MSP";
            break;
            case 21:
            return   "FTP";
            break;
            case 22:
            return   "SSH";
            break;
            case 23:
            return "TELNET";
            break;
            case 25:
            return "SMTP";
            break;
            case 29:
            return "MSG ICP";
            break;
            case 43:
            return "WHO IS";
            break;
            case 53:
            return "DNS";
            break;
            case 69:
            return "TFTP";
            break;
            case 80:
            return   "HTTP";
            break;
            case 110:
            return   "POP3";
            break;
            case 135:
            return   "RPC";
            break;
            case 139:
            return   "NETBIOS";
            break;
            case 143:
            return   "IMAP";
            break;
            case 161:
            return   "SNMP";
            break;
            case 162:
            return   "SNMPtrap";
            break;
            case 179:
            return   "BGP";
            break;
            case 443:
            return   "SSL";
            break;
            default:
            return   " ";
            break;
} }
```

ANEXO 5

PRINCIPALES REGLAS DIFUNDIDAS DEL SISTEMA DE DETECCIÓN DE INTRUSOS SNORT

 # ATTACK RESPONSES

$Id: attack-responses.rules,v 1.17 2002/11/04 14:00:35 cazz Exp $

These signatures are those when they happen, its usually because a machine
has been compromised. These should not false that often and almost always
mean a compromise.

alert tcp any 80 -> any any (msg:"ATTACK RESPONSES http dir listing"; content: "Volume Serial Number"; flow:from_server,established; sid:1292; rev:4;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES command completed"; content:"Command completed"; nocase; flow:from_server,established; sid:494; rev:5;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES command error"; content:"Bad command or filename"; nocase; flow:from_server,established; sid:495; rev:5;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES file copied ok"; content:"1 file(s) copied"; nocase; flow:from_server,established; sid:497; rev:5;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES Invalid URL"; content:"Invalid URL"; nocase; flow:from_server,established; sid:1200; rev:6;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES index of /cgi-bin/ response"; flow:from_server,established; content:"Index of /cgi-bin/"; nocase; sid:1666; rev:3;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES 403 Forbidden"; flow:from_server,established; content:"HTTP/1.1 403"; depth:12; sid:1201; rev:6;)
alert ip any any -> any any (msg:"ATTACK RESPONSES id check returned root"; content: "uid=0(root)"; sid:498; rev:3;)
alert tcp any 8002 -> any any (msg:"ATTACK RESPONSES oracle one hour install"; flow:from_server,established; content:"Oracle Applications One-Hour Install"; sid:1464; rev:2;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES id check returned www"; flow:from_server,established; content:"uid="; content:"(www)"; sid:1882; rev:2;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES id check returned nobody"; flow:from_server,established; content:"uid="; content:"(nobody)"; sid:1883; rev:2;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES id check returned web"; flow:from_server,established; content:"uid="; content:"(web)"; sid:1884; rev:2;)
alert tcp any 80 -> any any (msg:"ATTACK RESPONSES id check returned http"; flow:from_server,established; content:"uid="; content:"(http)"; sid:1885; rev:2;)

alert tcp any 80 -> any any (msg:"ATTACK RESPONSES id check returned apache"; flow:from_server,established; content:"uid="; content:"(apache)"; sid:1886; rev:2;)
alert tcp any 749 -> any any (msg:"ATTACK-RESPONSE successful kadmind bufferflow attempt"; flow:established,from_server; content:"*GOBBLE*"; depth:8; reference:cve,CAN-2002-1235; reference:url,www.kb.cert.org/vuls/id/875073; sid:1900; rev:1;)
alert tcp any 751 -> any any (msg:"ATTACK-RESPONSE successful kadmind bufferflow attempt"; flow:established,from_server; content:"*GOBBLE*"; depth:8; reference:cve,CAN-2002-1235; reference:url,www.kb.cert.org/vuls/id/875073; sid:1901; rev:1;)

 # BACKDOOR RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: backdoor.rules,v 1.25 2002/08/18 20:28:43 cazz Exp $
#---------------
#---------------
#

alert tcp any 27374 -> any any (msg:"BACKDOOR subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485; reference:url,www.hackfix.org/subseven/; sid:103; rev:4;)
alert tcp any 1024: -> any 2589 (msg:"BACKDOOR - Dagger_1.4.0_client_connect"; flags: A+; content: "|0b 00 00 00 07 00 00 00|Connect"; depth: 16; reference:url,www.tlsecurity.net/backdoor/Dagger.1.4.html; reference:arachnids,483; sid:104; rev:4;)
alert tcp any 2589 -> any 1024: (msg:"BACKDOOR - Dagger_1.4.0"; flags: A+; content: "|3200000006000000|Drives|2400|"; depth: 16; reference:arachnids,484; reference:url,www.tlsecurity.net/backdoor/Dagger.1.4.html; sid:105; rev:4;)
alert tcp any 80 -> any 1054 (msg:"BACKDOOR ACKcmdC trojan scan"; seq: 101058054; ack: 101058054; flags: A;reference:arachnids,445; sid:106; rev:3;)
alert tcp any 16959 -> any any (msg:"BACKDOOR subseven DEFCON8 2.1 access"; content: "PWD"; content:"acidphreak"; nocase; flags: A+; sid:107; rev:4;)
alert tcp any any -> any 7597 (msg:"BACKDOOR QAZ Worm Client Login access"; flags: A+; content:"|71 61 7a 77 73 78 2e 68 73 71|"; reference:MCAFEE,98775; sid:108; rev:3;)
alert tcp any 12345 -> any any (msg:"BACKDOOR netbus active"; flow:from_server,established; content:"NetBus"; reference:arachnids,401; sid:109; rev:3;)
alert tcp any any -> any 12345 (msg:"BACKDOOR netbus getinfo"; flags: A+; content: "GetInfo|0d|"; reference:arachnids,403; sid:110; rev:3;)
alert tcp any any -> any 12346 (msg:"BACKDOOR netbus getinfo"; flags: A+; content: "GetInfo|0d|"; reference:arachnids,403; sid:111; rev:3;)
alert tcp any 80 -> any any (msg:"BACKDOOR BackOrifice access"; flags: A+; content: "server|3a| BO|2f|"; reference:arachnids,400; sid:112; rev:3;)

alert udp any 4120 -> any any (msg:"BACKDOOR DeepThroat access"; content: "--Ahhhhhhhhhh"; reference:arachnids,405; sid:113; rev:3;)
alert tcp any 12346 -> any any (msg:"BACKDOOR netbus active"; flags: A+; content: "NetBus"; reference:arachnids,401; sid:114; rev:3;)
alert tcp any 20034 -> any any (msg:"BACKDOOR netbus active"; flags: A+; content: "NetBus"; reference:arachnids,401; sid:115; rev:3;)
alert udp any any -> any 31337 (msg:"BACKDOOR BackOrifice access"; content: "|ce63 d1d2 16e7 13cf 39a5 a586|"; reference:arachnids,399; sid:116; rev:3;)
alert tcp any 146 -> any 1024: (msg:"BACKDOOR Infector.1.x"; content: "WHATISIT"; flags: A+; reference:arachnids,315; sid:117; rev:3;)
alert tcp any 666 -> any 1024: (msg:"BACKDOOR SatansBackdoor.2.0.Beta"; content: "Remote|3A| You are connected to me."; flags:A+; reference:arachnids,316; sid:118; rev:3;)
alert tcp any 6789 -> any any (msg:"BACKDOOR Doly 2.0 access"; content: "|57 74 7a 75 70 20 55 73 65|"; flags: A+; depth: 32; reference:arachnids,312; sid:119; rev:3;)
alert tcp any 146 -> any 1000:1300 (msg:"BACKDOOR Infector 1.6 Server to Client"; content:"|57 48 41 54 49 53 49 54|"; flags:A+; sid:120; rev:3;)
alert tcp any 1000:1300 -> any 146 (msg:"BACKDOOR Infector 1.6 Client to Server Connection Request"; content:"|46 43 20|"; flags:A+; sid:121; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 System Info Client Request"; content:"13"; reference:arachnids,106; sid:122; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 FTP Status Client Request"; content:"09"; reference:arachnids,106; sid:124; rev:3;)
alert udp any 2140 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 E-Mail Info From Server"; content:"Retreaving"; reference:arachnids,106; sid:125; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 E-Mail Info Client Request"; content:"12"; reference:arachnids,106; sid:126; rev:3;)
alert udp any 2140 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 Server Status From Server"; content:"Host"; reference:arachnids,106; sid:127; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Server Status Client Request"; content:"10"; reference:arachnids,106; sid:128; rev:3;)
alert udp any 2140 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 Drive Info From Server"; content:"C - "; reference:arachnids,106; sid:129; rev:3;)
alert udp any 2140 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 System Info From Server"; content:"Comp Name"; reference:arachnids,106; sid:130; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Drive Info Client Request"; content:"130"; reference:arachnids,106; sid:131; rev:3;)
alert udp any 2140 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 Server FTP Port Change From Server"; content:"FTP Server changed to"; reference:arachnids,106; sid:132; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Cached Passwords Client Request"; content:"16"; reference:arachnids,106; sid:133; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 RAS Passwords Client Request"; content:"17"; reference:arachnids,106; sid:134; rev:3;)

alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Server Password Change Client Request"; content:"91"; reference:arachnids,106; sid:135; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Server Password Remove Client Request"; content:"92"; reference:arachnids,106; sid:136; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Rehash Client Request"; content:"911"; reference:arachnids,106; sid:137; rev:3;)
alert udp any 60000 -> any 3150 (msg:"BACKDOOR DeepThroat 3.1 Server Rehash Client Request"; content:"shutd0wnM0therF***eR"; reference:arachnids,106; sid:138; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 ICQ Alert OFF Client Request"; content:"88"; reference:arachnids,106; sid:140; rev:3;)
alert tcp any 31785 -> any any (msg:"BACKDOOR HackAttack 1.20 Connect"; flags: A+; content:"host"; sid:141; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 ICQ Alert ON Client Request"; content: "40"; reference:arachnids,106; sid:142; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Change Wallpaper Client Request"; content:"20"; reference:arachnids,106; sid:143; rev:3;)
alert tcp any !80 -> any 21554 (msg:"BACKDOOR GirlFriendaccess"; flags: A+; content:"Girl"; reference:arachnids,98; sid:145; rev:3;)
alert tcp any 30100 -> any any (msg:"BACKDOOR NetSphere access"; flags: A+; content:"NetSphere"; reference:arachnids,76; sid:146; rev:3;)
alert tcp any 6969 -> any any (msg:"BACKDOOR GateCrasher"; flags: A+; content:"GateCrasher";reference:arachnids,99; sid:147; rev:3;)
alert udp any 2140 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 Keylogger Active on Network"; content:"KeyLogger Is Enabled On port"; reference:arachnids,106; sid:148; rev:3;)
alert udp any 60000 -> any 3150 (msg:"BACKDOOR DeepThroat 3.1 Client Sending Data to Server on Network"; content:"|00 23|"; reference:arachnids,106; sid:149; rev:3;)
alert udp any 3150 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 Server Active on Network"; content:"|00 23|"; reference:arachnids,106; sid:150; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Client Sending Data to Server on Network"; reference:arachnids,106; sid:151; rev:3;)
alert tcp any 5401:5402 -> any any (msg:"BACKDOOR BackConstruction 2.1 Connection"; flags: A+; content:"c|3A|\\"; sid:152; rev:3;)
alert tcp any 23476 -> any any (msg:"BACKDOOR DonaldDick 1.53 Traffic"; flags: A+; content:"pINg"; sid:153; rev:3;)
alert udp any 3150 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 Wrong Password"; content:"Wrong Password"; reference:arachnids,106; sid:154; rev:3;)
alert tcp any 30100:30102 -> any any (msg:"BACKDOOR NetSphere 1.31.337 access"; flags: A+; content:"NetSphere"; reference:arachnids,76; sid:155; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Visible Window List Client Request"; content:"37"; reference:arachnids,106; sid:156; rev:3;)

alert tcp any any -> any 666 (msg:"BACKDOOR BackConstruction 2.1 Client FTP Open Request"; flags: A+; content:"FTPON"; sid:157; rev:3;)
alert tcp any 666 -> any any (msg:"BACKDOOR BackConstruction 2.1 Server FTP Open Reply"; flags: A+; content:"FTP Port open"; sid:158; rev:3;)
alert tcp any any -> any 5032 (msg:"BACKDOOR NetMetro File List"; flags: A+; content:"|2D 2D|"; reference:arachnids,79; sid:159; rev:3;)
#alert tcp any 5031 -> any !53:80 (msg:"BACKDOOR NetMetro Incoming Traffic"; flags: A+; reference:arachnids,79; sid:160; rev:2;)
alert udp any 3344 -> any 3345 (msg:"BACKDOOR Matrix 2.0 Client connect"; content:"activate"; reference:arachnids,83; sid:161; rev:3;)
alert udp any 3345 -> any 3344 (msg:"BACKDOOR Matrix 2.0 Server access"; content:"logged in"; reference:arachnids,83; sid:162; rev:3;)
alert tcp any 5714 -> any any (msg:"BACKDOOR WinCrash 1.0 Server Active" ; flags:SA; content:"|B4 B4|"; reference:arachnids,36; sid:163; rev:3;)
alert udp any 2140 -> any 60000 (msg:"BACKDOOR DeepThroat 3.1 Server Active on Network"; reference:arachnids,106; sid:164; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Keylogger on Server ON"; content:"KeyLogger Is Enabled On port"; reference:arachnids,106; sid:165; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Show Picture Client Request"; content:"22"; reference:arachnids,106; sid:166; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Hide/Show Clock Client Request"; content:"32"; reference:arachnids,106; sid:167; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Hide/Show Desktop Client Request"; content:"33"; reference:arachnids,106; sid:168; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Swap Mouse Buttons Client Request"; content:"34"; reference:arachnids,106; sid:169; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Enable/Disable CTRL-ALT-DEL Client Request"; content:"110"; reference:arachnids,106; sid:170; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Freeze Mouse Client Request"; content:"35"; reference:arachnids,106; sid:171; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Show Dialog Box Client Request"; content:"70"; reference:arachnids,106; sid:172; rev:4;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Show Replyable Dialog Box Client Request"; content:"71"; reference:arachnids,106; sid:173; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Hide/Show Start Button Client Request"; content:"31"; reference:arachnids,106; sid:174; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Resolution Change Client Request"; content:"125"; reference:arachnids,106; sid:175; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Hide/Show Start Button Client Request"; content:"04"; reference:arachnids,106; sid:176; rev:3;)

alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Keylogger on Server OFF"; content:"KeyLogger Shut Down"; reference:arachnids,106; sid:177; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 FTP Server Port Client Request"; content:"21"; reference:arachnids,106; sid:179; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Process List Client request"; content:"64"; reference:arachnids,106; sid:180; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Close Port Scan Client Request"; content:"121"; reference:arachnids,106; sid:181; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Registry Add Client Request"; content:"89"; reference:arachnids,106; sid:182; rev:3;)
alert icmp 255.255.255.0/24 any -> any any (msg:"BACKDOOR SIGNATURE - Q ICMP"; itype: 0; dsize: >1; reference:arachnids,202; sid:183; rev:3;)
alert tcp 255.255.255.0/24 any -> any any (msg:"BACKDOOR Q access"; flags:A+; dsize: >1; reference:arachnids,203; sid:184; rev:3;)
alert tcp any any -> any 79 (msg:"BACKDOOR CDK"; content: "ypi0ca"; nocase; flags: A+; depth: 15; reference:arachnids,263; sid:185; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Monitor on/off Client Request"; content:"07"; reference:arachnids,106; sid:186; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Delete File Client Request"; content:"41"; reference:arachnids,106; sid:187; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Kill Window Client Request"; content:"38"; reference:arachnids,106; sid:188; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Disable Window Client Request"; content:"23"; reference:arachnids,106; sid:189; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Enable Window Client Request"; content:"24"; reference:arachnids,106; sid:190; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Change Window Title Client Request"; content:"60"; reference:arachnids,106; sid:191; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Hide Window Client Request"; content:"26"; reference:arachnids,106; sid:192; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Show Window Client Request"; content:"25"; reference:arachnids,106; sid:193; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Send Text to Window Client Request"; content:"63"; reference:arachnids,106; sid:194; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Server Response"; content:"Ahhhh My Mouth Is Open"; reference:arachnids,106; sid:195; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Hide/Show Systray Client Request"; content:"30"; reference:arachnids,106; sid:196; rev:3;)

alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Create Directory Client Request"; content:"39"; reference:arachnids,106; sid:197; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 All Window List Client Request"; content:"370"; reference:arachnids,106; sid:198; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Play Sound Client Request"; content:"36"; reference:arachnids,106; sid:199; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Run Program Normal Client Request"; content:"14"; reference:arachnids,106; sid:200; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Run Program Hidden Client Request"; content:"15"; reference:arachnids,106; sid:201; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Get NET File Client Request"; content:"100"; reference:arachnids,106; sid:202; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Find File Client Request"; content:"117"; reference:arachnids,106; sid:203; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 Find File Client Request"; content:"118"; reference:arachnids,106; sid:204; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 HUP Modem Client Request"; content:"199"; reference:arachnids,106; sid:205; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 CD ROM Open Client Request"; content:"02"; reference:arachnids,106; sid:206; rev:3;)
alert udp any 60000 -> any 2140 (msg:"BACKDOOR DeepThroat 3.1 CD ROM Close Client Request"; content:"03"; reference:arachnids,106; sid:207; rev:3;)
alert tcp any 555 -> any any (msg:"BACKDOOR PhaseZero Server Active on Network"; flags: A+; content:"phAse"; sid:208; rev:3;)
alert tcp any any -> any 23 (msg:"BACKDOOR w00w00 attempt";flags: A+; content:"w00w00"; reference:arachnids,510; sid:209; rev:3;)
alert tcp any any -> any 23 (msg:"BACKDOOR attempt"; flags: A+; content:"backdoor"; nocase; sid:210; rev:2;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC r00t attempt";flags: A+; content:"r00t"; sid:211; rev:2;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC rewt attempt";flags: A+; content:"rewt"; sid:212; rev:2;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC linux rootkit attempt";flags: A+; content:"wh00t!"; sid:213; rev:2;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC linux rootkit attempt lrkr0x";flags: A+; content:"lrkr0x"; sid:214; rev:2;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC linux rootkit attempt";flags: A+; content:"d13hh["; nocase; sid:215; rev:2;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC linux rootkit satori attempt";flags: A+; content:"satori"; reference:arachnids,516; sid:216; rev:4;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC sm4ck attempt";flags: A+; content:"hax0r";sid:217; rev:2;)
alert tcp any any -> any 23 (msg:"BACKDOOR MISC solaris 2.5 attempt";flags: A+; content:"friday"; sid:218; rev:2;)

alert tcp any any -> any 23 (msg:"BACKDOOR HidePak backdoor attempt";flags: A+; content:"StoogR"; sid:219; rev:4;)
alert tcp any any -> any 23 (msg:"BACKDOOR HideSource backdoor attempt";flags: A+; content:"wank"; sid:220; rev:4;)
alert tcp any 31790 -> any 31789 (msg:"BACKDOOR hack-a-tack attempt"; content: "A"; depth: 1; reference:arachnids,314; flags:A+; sid:614; rev:2;)

 # BAD TRAFFIC RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: bad-traffic.rules,v 1.18 2002/08/18 20:28:43 cazz Exp $
#------------------
#------------------
These signatures are representitive of traffic that should never be seen on
any network. None of these signatures include datagram content checking
and are extremely quick signatures
#

alert tcp any any <> any 0 (msg:"BAD TRAFFIC tcp port 0 traffic"; reference:cve,CVE-1999-0675; reference:nessus,10074; sid:524; rev:4;)
alert udp any any <> any 0 (msg:"BAD TRAFFIC udp port 0 traffic"; reference:cve,CVE-1999-0675; reference:nessus,10074; sid:525; rev:4;)
alert tcp any any -> any any (msg:"BAD TRAFFIC data in TCP SYN packet"; flags:S; dsize:>6; reference:url,www.cert.org/incident_notes/IN-99-07.html; sid:526; rev:4;)
alert ip any any <> 127.0.0.0/8 any (msg:"BAD TRAFFIC loopback traffic"; reference:url,rr.sans.org/firewall/egress.php; sid:528; rev:3;)
alert ip any any -> any any (msg:"BAD TRAFFIC same SRC/DST"; sameip; reference:cve,CVE-1999-0016; reference:url,www.cert.org/advisories/CA-1997-28.html; sid:527; rev:3;)
alert ip any any -> any any (msg:"BAD TRAFFIC ip reserved bit set"; fragbits:R; sid:523; rev:3;)
alert ip any any -> any any (msg:"BAD TRAFFIC 0 ttl"; ttl:0; reference:url,www.isi.edu/in-notes/rfc1122.txt;
reference:url,support.microsoft.com/default.aspx?scid=kb\;EN-US\;q138268;
sid:1321; rev:5;)
alert ip any any -> any any (msg:"BAD TRAFFIC bad frag bits"; fragbits:MD; sid:1322; rev:4;)
alert ip any any -> any any (msg:"BAD TRAFFIC Unassigned/Reserved IP protocol"; ip_proto:>134; sid:1627; rev:1;)
alert ip any any -> any any (msg:"BAD TRAFFIC Non-Standard IP protocol"; ip_proto:!1; ip_proto:!2; ip_proto:!6; ip_proto:!47; ip_proto:!50; ip_proto:!51; ip_proto:!89; sid:1620; rev:2;)
alert tcp any any -> [232.0.0.0/8,233.0.0.0/8,239.0.0.0/8] any (msg:"BAD TRAFFIC syn to multicast address"; flags:S+; sid:1431; rev:4;)

 # CHAT RULES

$Id: chat.rules,v 1.13 2002/11/05 21:01:56 cazz Exp $
#------------
#------------
These signatures look for people using various types of chat programs (for
example: AIM, ICQ, and IRC) which may be against corporate policy

alert tcp any any -> any 1863 (msg:"CHAT MSN chat access"; flow:to_server,established; content:"text/plain"; depth:100; sid:540; rev:6;)
alert tcp any any -> any any (msg:"CHAT ICQ access"; flow:to_server,established; content: "User-Agent\:ICQ"; sid:541; rev:6;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC nick change"; flow:to_server,established; content: "NICK "; offset:0; sid:542; rev:8;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC DCC file transfer request"; flow:to_server,established; content:"PRIVMSG "; nocase; offset:0; content:" \:.DCC SEND"; nocase; sid:1639; rev:3;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC DCC chat request"; flow:to_server,established; content:"PRIVMSG "; nocase; offset:0; content:" \:.DCC CHAT chat"; nocase; sid:1640; rev:3;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC channel join"; flow:to_server,established; content:"JOIN \: \\#"; nocase; offset:0; sid:1729; rev:2;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC message"; flow:to_server,established; content:"PRIVMSG "; nocase; offset:0; sid:1463; rev:3;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC dns request"; flow:to_server,established; content:"USERHOST "; nocase; offset:0; sid:1789; rev:1;)
alert tcp any 6666:7000 -> any any (msg:"CHAT IRC dns response"; flow:to_client,established; content:"\:"; offset:0; content:" 302 "; content:"=+"; sid:1790; rev:2;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC EXPLOIT topic overflow"; flow:to_client,established; content:"|eb 4b 5b 53 32 e4 83 c3 0b 4b 88 23 b8 50 77|"; reference:cve,CVE-1999-0672; reference:bugtraq,573; sid:307; rev:5;)
alert tcp any any -> any 6666:7000 (msg:"CHAT IRC EXPLOIT Ettercap parse overflow attempt"; flow:to_server,established; content:"PRIVMSG nickserv IDENTIFY"; nocase; offset:0; content:!"|0a|"; within:150; reference:url,www.bugtraq.org/dev/GOBBLES-12.txt; sid:1382; rev:6;)
alert tcp any any -> any any (msg:"CHAT AIM login"; flow:to_server,established; content:"|2a 01|"; offset:0; depth:2; sid:1631; rev:4;)
alert tcp any any -> any any (msg:"CHAT AIM send message"; flow:to_server,established; content:"|2a 02|"; offset:0; depth:2; content:"|00 04 00 06|"; offset:6; depth:4; sid:1632; rev:4;)
alert tcp any any -> any any (msg:"CHAT AIM recieve message"; flow:to_client; content:"|2a 02|"; offset:0; depth:2; content:"|00 04 00 07|"; offset:6; depth:4; sid:1633; rev:3;)

 # FTP RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: ftp.rules,v 1.33 2002/11/05 21:01:59 cazz Exp $
#----------
#----------

alert tcp any any -> any 21 (msg:"FTP EXPLOIT STAT * dos attempt"; flow:to_server,established; content:"STAT "; nocase; content:"*"; reference:bugtraq,4482;sid:1777; rev:1;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT STAT ? dos attempt"; flow:to_server,established; content:"STAT "; nocase; content:"?"; reference:bugtraq,4482; sid:1778; rev:1;)
alert tcp any any -> any 21 (msg:"FTP .forward"; content: ".forward"; flow:to_server,established; reference:arachnids,319; sid:334; rev:4;)
alert tcp any any -> any 21 (msg:"FTP .rhosts"; flow:to_server,established; content:".rhosts"; reference:arachnids,328; sid:335; rev:4;)
alert tcp any any -> any 21 (msg:"FTP CWD ~root attempt"; content:"CWD "; content:" ~root"; nocase; flow:to_server,established; reference:cve,CVE-1999-0082; reference:arachnids,318; sid:336; rev:5;)
alert tcp any any -> any 21 (msg:"FTP CEL overflow attempt";flow:to_server,established; content:"CEL "; nocase; content:!"|0a|"; within:100; reference:bugtraq,679; reference:cve,CVE-1999-0789; reference:arachnids,257; sid:337; rev:5;)
alert tcp any any -> any 21 (msg:"FTP CWD overflow attempt"; flow:to_server,established; content:"CWD "; nocase; content:!"|0a|"; within:100; sid:1919; rev:1;)
alert tcp any any -> any 21 (msg:"FTP CMD overflow attempt"; flow:to_server,established; content:"CMD "; nocase; content:!"|0a|"; within:100; sid:1621; rev:8;)
alert tcp any any -> any 21 (msg:"FTP STAT overflow attempt"; flow:to_server,established; content:"STAT "; nocase; content:!"|0a|"; within:100; reference:url,labs.defcom.com/adv/2001/def-2001-31.txt; sid:1379; rev:5;)
alert tcp any any -> any 21 (msg:"FTP SITE CHOWN overflow attempt"; flow:to_server,established; content:"SITE "; nocase; content:" CHOWN "; nocase; content:!"|0a|"; within:100; reference:cve,CAN-2001-0065; sid:1562; rev:6;)
alert tcp any any -> any 21 (msg:"FTP SITE NEWER overflow attempt"; flow:to_server,established; content:"SITE "; nocase; content:" NEWER "; nocase; content:!"|0a|"; within:100; reference:cve,CVE-1999-0800; sid:1920; rev:1;)
alert tcp any any -> any 21 (msg:"FTP SITE overflow attempt"; flow:to_server,established; content:"SITE "; nocase; content:!"|0a|"; within:100; reference:cve,CAN-2001-0755; reference:cve,CAN-2001-0770; reference:cve,CVE-1999-0838; sid:1529; rev:7;)
alert tcp any any -> any 21 (msg:"FTP USER overflow attempt"; flow:to_server,established,no_stream; content:"USER "; nocase;

content:!"|0a|"; within:100; reference:bugtraq,4638; reference:cve,CAN-2000-0479; sid:1734; rev:6;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT format string"; flow:to_server,established; content: "SITE EXEC |25 30 32 30 64 7C 25 2E 66 25 2E 66 7C 0A|"; depth: 32; nocase; reference:cve,CVE-2000-0573; reference:bugtraq,1387; reference:arachnids,453; sid:338; rev:4;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT OpenBSD x86 ftpd"; flow:to_server,established; content: " |90 31 C0 99 52 52 B017 CD80 68 CC 73 68|"; reference:cve,CVE-2001-0053; reference:bugtraq,2124; reference:arachnids,446; sid:339; rev:4;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT overflow"; flow:to_server,established; content:"|5057 440A 2F69|"; sid:340; rev:3;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT overflow"; flow:to_server,established; content:"|5858 5858 582F|"; sid:341; rev:3;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT wu-ftpd 2.6.0 site exec format string overflow Solaris 2.8"; flow:to_server,established; content: "|901BC00F 82102017 91D02008|"; reference:bugtraq,1387; reference:cve,CAN-2000-0573; reference:arachnids,451; sid:342; rev:4;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT wu-ftpd 2.6.0 site exec format string check"; flow:to_server,established; content:"f%.f%.f%.f%.f%."; depth:32; reference:arachnids,286; reference:bugtraq,1387; reference:cve,CAN-2000-0573;sid:346; rev:4;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT wu-ftpd 2.6.0"; flow:to_server,established; content:"|2e2e3131|venglin@"; reference:arachnids,440; reference:bugtraq,1387; sid:348; rev:3;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT MKD overflow"; flow:to_server,established; content:"MKD AAAAAA"; reference:bugtraq,113; reference:cve,CVE-1999-0368; sid:349; rev:4;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT x86 linux overflow"; flow:to_server,established; content:"|31c0 31db b017 cd80 31c0 b017 cd80|"; reference:bugtraq,113; reference:cve,CVE-1999-0368; sid:350; rev:3;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT x86 linux overflow"; flow:to_server,established; content:"|31db 89d8 b017 cd80 eb2c|"; reference:bugtraq,113; reference:cve,CVE-1999-0368;sid:351; rev:3;)
alert tcp any any -> any 21 (msg:"FTP EXPLOIT x86 linux overflow"; flow:to_server,established; content:"|83 ec 04 5e 83 c6 70 83 c6 28 d5 e0 c0|";reference:bugtraq, 113; reference:cve, CVE-1999-0368; sid:352; rev:3;)
alert tcp any any -> any 21 (msg:"FTP adm scan"; flow:to_server,established; content:"PASS ddd@|0a|"; reference:arachnids,332; sid:353; rev:4;)
alert tcp any any -> any 21 (msg:"FTP iss scan"; flow:to_server,established; content:"pass -iss@iss"; reference:arachnids,331; sid:354; rev:4;)
alert tcp any any -> any 21 (msg:"FTP pass wh00t"; flow:to_server,established; content:"pass wh00t"; nocase; reference:arachnids,324; sid:355; rev:4;)
alert tcp any any -> any 21 (msg:"FTP passwd retreval attempt"; flow:to_server,established; content:"RETR"; nocase; content:"passwd"; reference:arachnids,213; sid:356; rev:4;)
alert tcp any any -> any 21 (msg:"FTP piss scan"; flow:to_server,established; content:"pass -cklaus";sid:357; rev:4;)
alert tcp any any -> any 21 (msg:"FTP saint scan"; flow:to_server,established; content:"pass -saint"; reference:arachnids,330; sid:358; rev:4;)

alert tcp any any -> any 21 (msg:"FTP satan scan"; flow:to_server,established; content:"pass -satan"; reference:arachnids,329; sid:359; rev:4;)
alert tcp any any -> any 21 (msg:"FTP serv-u directory transversal"; flow:to_server,established; content: ".%20."; nocase; reference:bugtraq,2025; reference:cve,CVE-2001-0054; sid:360; rev:4;)
alert tcp any any -> any 21 (msg:"FTP SITE ZIPCHK attempt"; flow:to_server,established; content:"SITE "; nocase; content:" ZIPCHK "; nocase; content:!"|0a|"; within:100; reference:cve,CVE-2000-0040; sid:1921; rev:1;)
alert tcp any any -> any 21 (msg:"FTP site exec"; flow:to_server,established; content:"site "; nocase; content:" exec "; offset:4; nocase; reference:bugtraq,2241; reference:arachnids,317; sid:361; rev:6;)
alert tcp any any -> any 21 (msg:"FTP tar parameters"; flow:to_server,established; content:"RETR "; nocase; content:" --use-compress-program"; nocase; reference:bugtraq,2240; reference:arachnids,134; reference:cve,CVE-1999-0202; sid:362; rev:6;)
alert tcp any any -> any 21 (msg:"FTP CWD ..."; flow:to_server,established; content:"CWD "; content:" ..."; sid:1229; rev:4;)
alert tcp any any -> any 21 (msg:"FTP wu-ftp bad file completion attempt ["; flow:to_server,established; content:"~"; content:"["; reference:cve,CVE-2001-0550; reference:cve,CAN-2001-0886; reference:bugtraq,3581; sid:1377; rev:9;)
alert tcp any any -> any 21 (msg:"FTP wu-ftp bad file completion attempt {"; flow:to_server,established; content:"~"; content:"{"; reference:cve,CVE-2001-0550; reference:cve,CAN-2001-0886; reference:bugtraq,3581; sid:1378; rev:9;)
alert tcp any any -> any 21 (msg:"FTP ADMw0rm ftp login attempt"; flow:to_server,established; content:"USER w0rm|0D0A|"; reference:arachnids,01; sid:144; rev:6;)
alert tcp any any -> any 21 (msg:"FTP file_id.diz access"; flow:to_server,established; content:"RETR "; nocase; content:"file_id.diz"; nocase;sid:1445; rev:2;)
alert tcp any any -> any 21 (msg:"FTP \"STOR 1MB\" possible warez site"; flow:to_server,established; content:"STOR 1MB"; nocase; depth: 8;sid:543; rev:4;)
alert tcp any any -> any 21 (msg:"FTP \"RETR 1MB\" possible warez site"; flow:to_server,established; content:"RETR 1MB"; nocase; depth: 8; sid:544; rev:4;)
alert tcp any any -> any 21 (msg:"FTP \"CWD /\" possible warez site"; flow:to_server,established; content:"CWD / "; nocase; depth: 6; sid:545; rev:3;)
alert tcp any any -> any 21 (msg:"FTP \"CWD \" possible warez site"; flow:to_server,established; content:"CWD "; nocase; depth: 5; sid:546; rev:4;)
alert tcp any any -> any 21 (msg:"FTP \"MKD \" possible warez site"; flow:to_Server,established; content:"MKD "; nocase; depth: 5; sid:547; rev:4;)
alert tcp any any -> any 21 (msg:"FTP \"MKD . \" possible warez site"; flow:to_server,established; content:"MKD ."; nocase; depth: 5; sid:548; rev:4;)
alert tcp any any -> any 21 (msg:"FTP \"MKD / \" possible warez site"; flow:to_server,established; content:"MKD / "; nocase; depth: 6; sid:554; rev:5;)
alert tcp any any -> any 21 (msg:"FTP CWD ~<NEWLINE> attempt"; content:"CWD "; content:" ~|0A|"; flow:to_server,established; reference:cve,CAN-2001-0421; reference:bugtraq,2601;sid:1672; rev:2;)

alert tcp any any -> any 21 (msg:"FTP CWD ~<CR><NEWLINE> attempt"; content:"CWD "; content:" ~|0D0A|"; flow:to_server,established; reference:cve,CAN-2001-0421; reference:bugtraq,2601; sid:1728; rev:2;)
alert tcp any any -> any 21 (msg:"FTP CWD .... attempt"; content:"CWD "; content:" ...."; flow:to_server,established; reference:bugtraq,4884;sid:1779; rev:1;)
alert tcp any any -> any 21 (msg:"FTP RNFR ././ attempt"; flow:to_server,established; content:"RNFR "; nocase; content:" ././"; nocase; sid:1622; rev:5;)
alert tcp any any -> any 21 (msg:"FTP invalid MODE"; flow:to_server,established; content:"MODE "; nocase; content:!" B"; nocase; content:!" A"; nocase; content:!" S"; nocase; content:!" C"; nocase; sid:1623; rev:4;)
alert tcp any any -> any 21 (msg:"FTP large PWD command"; flow:to_server,established; content:"PWD"; nocase; dsize:10; sid:1624; rev:3;)
alert tcp any any -> any 21 (msg:"FTP large SYST command"; flow:to_server,established; content:"SYST"; nocase; dsize:10; sid:1625; rev:3;)
alert tcp any any -> any 21 (msg:"FTP format string attempt"; flow:to_server,established; content:"%p"; nocase;sid:1530; rev:4;)
alert tcp any any -> any 21 (msg:"FTP command overflow attempt"; flow:to_server,established,no_stream; dsize:>100; reference:bugtraq,4638; sid:1748; rev:4;)

 # MYSQL RULES

$Id: mysql.rules,v 1.2 2002/08/18 20:28:43 cazz Exp $
#----------
#----------
#
These signatures detect unusual and potentually malicous mysql traffic.
#
These signatures are not enabled by default as they may generate false
positive alarms on networks that do mysql development.
#

alert tcp any any -> any 3306 (msg:"MYSQL root login attempt"; flow:to_server,established; content:"|0A 00 00 01 85 04 00 00 80 72 6F 6F 74 00|"; sid:1775; rev:1;)
alert tcp any any -> any 3306 (msg:"MYSQL show databases attempt"; flow:to_server,established; content:"|0f 00 00 00 03|show databases"; sid:1776; rev:1;)

 # POP3 RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: pop3.rules,v 1.5 2002/11/05 21:02:00 cazz Exp $
#--------------
#--------------

alert tcp any any -> any 110 (msg:"POP3 USER overflow attempt"; flow:to_server,established; content:"USER "; nocase; content:!"|0a|"; within:50; reference:cve,CVE-1999-0494; reference:nessus,10311; sid:1866; rev:3;)
alert tcp any any -> any 110 (msg:"POP3 PASS overflow attempt"; flow:to_server,established; content:"PASS "; nocase; content:!"|0a|"; within:50; reference:cve,CAN-1999-1511; reference:nessus,10325; sid:1634; rev:5;)
alert tcp any any -> any 110 (msg:"POP3 APOP overflow attempt"; flow:to_server,established; content:"APOP "; nocase; content:!"|0a|"; within:256; reference:cve,CAN-2000-0841; reference:bugtraq,1652; reference:nessus,10559; sid:1635; rev:5;)
alert tcp any any -> any 110 (msg:"POP3 EXPLOIT x86 bsd overflow"; flow:to_server,established; content:"|5e0 e31c 0b03 b8d7 e0e8 9fa 89f9|"; sid:286; rev:4;)
alert tcp any any -> any 110 (msg:"POP3 EXPLOIT x86 bsd overflow"; flow:to_server,established; content:"|685d 5eff d5ff d4ff f58b f590 6631|"; sid:287; rev:4;)
alert tcp any any -> any 110 (msg:"POP3 EXPLOIT x86 linux overflow"; flow:to_server,established; content:"|d840 cd80 e8d9 ffff ff|/bin/sh"; sid:288; rev:4;)
alert tcp any any -> any 110 (msg:"POP3 EXPLOIT x86 sco overflow"; flow:to_server,established; content:"|560e 31c0 b03b 8d7e 1289 f989 f9|"; sid:289; rev:4;)
alert tcp any any -> any 110 (msg:"POP3 EXPLOIT qpopper overflow"; flow:to_server,established; content:"|E8 D9FF FFFF|/bin/sh"; reference:bugtraq,830; reference:cve,CAN-1999-0822; sid:290; rev:5;)

 # SMTP RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: smtp.rules,v 1.24 2002/11/05 21:02:01 cazz Exp $
#-----------
#-----------

alert tcp any any -> any 25 (msg:"SMTP RCPT TO overflow"; flow:to_server,established; content:"rcpt to|3a|"; nocase; content:!"|0a|"; within:800; reference:cve,CAN-2001-0260; reference:bugtraq,2283; sid:654; rev:7;)
alert tcp any any -> any 25 (msg:"SMTP chameleon overflow"; flow:to_server,established,no_stream; content: "HELP "; nocase; depth:5; content:!"|0a|"; within:500; reference:bugtraq,2387; reference:arachnids,266; reference:cve,CAN-1999-0261; sid:657; rev:7;)

alert tcp any 113 -> any 25 (msg:"SMTP sendmail 8.6.9 exploit"; flow:to_server,established; content:"|0a|D/"; reference:arachnids,140; reference:cve,CVE-1999-0204; sid:655; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP EXPLOIT x86 windows CSMMail overflow"; flow:to_server,established; content:"|eb53 eb20 5bfc 33c9 b182 8bf3 802b|"; reference:bugtraq,895; reference:cve,CVE-2000-0042; sid:656; rev:5;)
alert tcp any any -> any 25 (msg:"SMTP exchange mime DOS"; flow:to_server,established; content:"|63 68 61 72 73 65 74 20 3D 20 22 22|"; sid:658; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP expn decode"; flow:to_server,established; content:"expn decode"; nocase; reference:arachnids,32; sid:659; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP expn root"; flow:to_server,established; content:"expn root"; nocase; reference:arachnids,31; sid:660; rev:5;)
alert tcp any any -> any 25 (msg:"SMTP expn *@"; flow:to_server,established; content:"expn *@"; nocase; reference:cve,CAN-1999-1200; sid:1450; rev:3;)
alert tcp any any -> any 25 (msg:"SMTP majordomo ifs"; flow:to_server,established; content:"eply-to|3a| a~.`/bin/"; reference:cve,CVE-1999-0208; reference:arachnids,143; sid:661; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 5.5.5 exploit"; flow:to_server,established; content:"mail from|3a20227c|"; nocase; reference:arachnids,119; sid:662; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 5.5.8 overflow"; flow:to_server,established; content: "|7c 73 65 64 20 2d 65 20 27 31 2c 2f 5e 24 2f 27|"; reference:arachnids,172; reference:cve,CVE-1999-0095; sid:663; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 5.6.4 exploit"; flow:to_server,established; content:"rcpt to|3a| decode"; nocase; reference:arachnids,121; sid:664; rev:5;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 5.6.5 exploit"; flow:to_server,established; content:"MAIL FROM|3a207c|/usr/ucb/tail"; nocase; reference:arachnids,122; sid:665; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 8.4.1 exploit"; flow:to_server,established; content:"rcpt to|3a207c| sed '1,/^$/d'|7c|"; nocase;reference:arachnids,120; sid:666; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 8.6.10 exploit"; flow:to_server,established; content:"Croot|0d0a|Mprog, P=/bin/"; reference:arachnids,123; sid:667; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 8.6.10 exploit"; flow:to_server,established; content:"Croot|09090909090909|Mprog,P=/bin"; reference:arachnids,124; sid:668; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 8.6.9 exploit"; flow:to_server,established; content:"|0a|Croot|0a|Mprog";reference:arachnids,142; reference:cve,CVE-1999-0204; sid:669; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP sendmail 8.6.9 exploit"; flow:to_server,established; content:"|0a|C|3a|daemon|0a|R"; reference:cve,CVE-1999-0204; reference:arachnids,139; sid:670; rev:4;)

alert tcp any any -> any 25 (msg:"SMTP sendmail 8.6.9c exploit"; flow:to_server,established; content:"|0a|Croot|0d0a|Mprog"; reference:arachnids,141; reference:cve,CVE-1999-0204; sid:671; rev:4;)
alert tcp any any -> any 25 (msg:"SMTP vrfy decode"; flow:to_server,established; content:"vrfy decode"; nocase; reference:arachnids,373; sid:672; rev:3;)
alert tcp any any -> any 25 (msg:"SMTP vrfy root"; flow:to_server,established; content:"vrfy root"; nocase; sid:1446; rev:3;)
alert tcp any any -> any 25 (msg:"SMTP ehlo cybercop attempt"; flow:to_server,established; content:"ehlo cybercop|0a|quit|0a|"; reference:arachnids,372; sid:631; rev:5;)
alert tcp any any -> any 25 (msg:"SMTP expn cybercop attempt"; flow:to_server,established; content:"expn cybercop"; reference:arachnids,371; sid:632; rev:5;)
alert tcp any any -> any 25 (msg:"SMTP HELO overflow attempt"; flow:to_server,established; content:"HELO "; offset:0; depth:5; content:!"|0a|"; within:500; reference:cve,CVE-2000-0042; reference:nessus,10324; sid:1549; rev:9;)
alert tcp any any -> any 25 (msg:"SMTP ETRN overflow attempt"; flow:to_server,established; content:"ETRN "; offset:0; depth:5; content:!"|0A|"; within:500; reference:cve,CAN-2000-0490; sid:1550; rev:6;)

 # TELNET RULES

$Id: telnet.rules,v 1.25 2002/08/18 20:28:43 cazz Exp $
#-------------
#-------------
#
These signatures are based on various telnet exploits and unpassword
protected accounts.
#

alert tcp any any -> any 23 (msg:"TELNET solaris memory mismanagement exploit attempt"; flow:to_server,established; content:"|A0 23 A0 10 AE 23 80 10 EE 23 BF EC 82 05 E0 D6 90 25 E0|"; sid:1430; rev:5;)
alert tcp any any -> any 23 (msg:"TELNET SGI telnetd format bug"; flow:to_server,established; content:"_RLD"; content:"bin/sh"; reference:arachnids,304; sid:711; rev:5;)
alert tcp any any -> any 23 (msg:"TELNET ld_library_path"; flow:to_server,established; content:"ld_library_path"; reference:cve,CVE-1999-0073; reference:arachnids,367; sid:712; rev:5;)
alert tcp any any -> any 23 (msg:"TELNET livingston DOS"; flow:to_server,established; content:"|fff3 fff3 fff3 fff3 fff3|"; reference:arachnids,370; sid:713; rev:5;)

alert tcp any any -> any 23 (msg:"TELNET resolv_host_conf"; flow:to_server,established; content:"resolv_host_conf"; reference:arachnids,369; sid:714; rev:4;)
alert tcp any 23 -> any any (msg:"TELNET Attempted SU from wrong group"; flow:from_server,established; content:"to su root"; nocase; sid:715; rev:6;)
alert tcp any 23 -> any any (msg:"TELNET not on console"; flow:from_server,established; content:"not on system console"; nocase; reference:arachnids,365; sid:717; rev:6;)
alert tcp any 23 -> any any (msg:"TELNET login incorrect"; content:"Login incorrect"; flow:from_server,established; reference:arachnids,127; sid:718; rev:6;)
alert tcp any 23 -> any any (msg:"TELNET root login"; content:"login\: root"; flow:from_server,established; sid:719; rev:5;)
alert tcp any 23 -> any any (msg:"TELNET bsd telnet exploit response"; flow:from_server,established; content: "|0D0A|[Yes]|0D0A FFFE 08FF FD26|"; sid:1252; rev:8; reference:bugtraq,3064; reference:cve,CAN-2001-0554;)
alert tcp any any -> any 23 (msg:"TELNET bsd exploit client finishing"; flow:to_client,established; dsize:>200; content:"|FF F6 FF F6 FF FB 08 FF F6|"; offset:200; depth:50; sid:1253; reference:bugtraq,3064; reference:cve,CAN-2001-0554; rev:7;)
alert tcp any any -> any 23 (msg:"TELNET 4Dgifts SGI account attempt"; flow:to_server,established; content:"4Dgifts"; reference:cve,CAN-1999-0501;sid:709; rev:6;)
alert tcp any any -> any 23 (msg:"TELNET EZsetup account attempt"; flow:to_server,established; content:"OutOfBox"; reference:cve,CAN-1999-0501; sid:710; rev:6;)
alert tcp any 23 -> any any (msg:"TELNET access"; flow:from_server,established; content:"|FF FD 18 FF FD 1F FF FD 23 FF FD 27 FF FD 24|"; reference:arachnids,08; reference:cve,CAN-1999-0619; sid:716; rev:5;)

 # NETBIOS RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: netbios.rules,v 1.16 2002/08/18 20:28:43 cazz Exp $
#--------------
#--------------

alert tcp any any -> any 139 (msg:"NETBIOS nimda .eml"; content:"|00|E|00|M|00|L"; flow:to_server,established; reference:url,www.datafellows.com/v-descs/nimda.shtml; sid:1293; rev:6;)
alert tcp any any -> any 139 (msg:"NETBIOS nimda .nws"; content:"|00|N|00|W|00|S"; flow:to_server,established; reference:url,www.datafellows.com/v-descs/nimda.shtml; sid:1294; rev:6;)
alert tcp any any -> any 139 (msg:"NETBIOS nimda RICHED20.DLL"; content:"R|00|I|00|C|00|H|00|E|00|D|00|2|00|0"; flow:to_server,established; reference:url,www.datafellows.com/v-descs/nimda.shtml; sid:1295; rev:6;)

alert tcp any any -> any 139 (msg:"NETBIOS DOS RFPoison"; flow:to_server,established; content: "|5C 00 5C 00 2A 00 53 00 4D 00 42 00 53 00 45 00 52 00 56 00 45 00 52 00 00 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00|";reference:arachnids,454; sid:529; rev:5;)
alert tcp any any -> any 139 (msg:"NETBIOS NT NULL session"; flow:to_server,established; content: "|00 00 00 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00 38 00 31|"; reference:bugtraq,1163; reference:cve,CVE-2000-0347; reference:arachnids,204; sid:530; rev:7;)
alert tcp any any -> any 139 (msg:"NETBIOS RFParalyze Attempt"; flow:to_server,established; content:"BEAVIS"; content:"yep yep"; sid:1239; rev:5;)
alert tcp any any -> any 139 (msg:"NETBIOS SMB ADMIN$access"; flow:to_server,established; content:"\\ADMIN$|00 41 3a 00|"; reference:arachnids,340; sid:532; rev:4;)
alert tcp any any -> any 139 (msg:"NETBIOS SMB C$ access"; flow:to_server,established; content: "|5c|C$|00 41 3a 00|";reference:arachnids,339; sid:533; rev:5;)
alert tcp any any -> any 139 (msg:"NETBIOS SMB CD.."; flow:to_server,established; content:"\\..|2f 00 00 00|"; reference:arachnids,338; sid:534; rev:4;)
alert tcp any any -> any 139 (msg:"NETBIOS SMB CD..."; flow:to_server,established; content:"\\...|00 00 00|"; reference:arachnids,337; sid:535; rev:4;)
alert tcp any any -> any 139 (msg:"NETBIOS SMB D$access"; flow:to_server,established; content:"\\D$|00 41 3a 00|"; reference:arachnids,336; sid:536; rev:4;)
alert tcp any any -> any 139 (msg:"NETBIOS SMB IPC$access"; flow:to_server,established; content:"\\IPC$|00 41 3a 00|"; reference:arachnids,335; sid:537; rev:4;)
alert tcp any any -> any 139 (msg:"NETBIOS SMB IPC$access"; flow:to_server,established; content:"|5c00|I|00|P|00|C|00|$|000000|IPC|00|"; reference:arachnids,334; sid:538; rev:4;)
alert tcp any any -> any 139 (msg:"NETBIOS Samba clientaccess"; flow:to_server,established; content:"|00|Unix|00|Samba"; reference:arachnids,341; sid:539; rev:4;)

 # DOS RULES

$Id: dos.rules,v 1.25 2002/10/28 22:08:04 cazz Exp $
#----------
#----------

alert ip any any -> any any (msg:"DOS Jolt attack"; fragbits: M; dsize:408; reference:cve,CAN-1999-0345; sid:268; rev:2;)
alert tcp any any -> any any (msg:"DOS Land attack"; id:3868; seq: 3868; flags:S; reference:cve,CVE-1999-0016; sid:269; rev:2;)

alert udp any any -> any any (msg:"DOS Teardrop attack"; id:242; fragbits:M; reference:cve,CAN-1999-0015; reference:url,www.cert.org/advisories/CA-1997-28.html; reference:bugtraq,124; sid:270; rev:2;)
alert udp any 19 <> any 7 (msg:"DOS UDP echo+chargen bomb"; reference:cve,CAN-1999-0635; reference:cve,CVE-1999-0103; sid:271; rev:3;)
alert ip any any -> any any (msg:"DOS IGMP dos attack"; content:"|02 00|"; depth: 2; ip_proto: 2; fragbits: M+; reference:cve,CVE-1999-0918; sid:272; rev:2;)
alert ip any any -> any any (msg:"DOS IGMP dos attack"; content:"|00 00|"; depth:2; ip_proto:2; fragbits:M+; reference:cve,CVE-1999-0918; sid:273; rev:2;)
alert icmp any any -> any any (msg:"DOS ath"; content:"+++ath"; nocase; itype: 8; reference:cve,CAN-1999-1228; reference:arachnids,264; sid:274; rev:2;)
alert tcp any any <> any any (msg:"DOS NAPTHA"; flags:S; seq: 6060842; id: 413; reference:cve,CAN-2000-1039; reference:url,www.microsoft.com/technet/security/bulletin/MS00-091.asp; reference:url,www.cert.org/advisories/CA-2000-21.html; reference:url,razor.bindview.com/publish/advisories/adv_NAPTHA.html; reference:bugtraq,2022; sid:275; rev:4;)
alert tcp any any -> any 7070 (msg:"DOS Real Audio Server"; flow:to_server,established; content: "|fff4 fffd 06|"; reference:bugtraq,1288; reference:cve,CVE-2000-0474; reference:arachnids,411; sid:276; rev:2;)
alert tcp any any -> any 7070 (msg:"DOS Real Server template.html"; flow:to_server,established; content:"/viewsource/template.html?"; nocase; reference:cve,CVE-2000-0474; reference:bugtraq,1288; sid:277; rev:3;)
alert tcp any any -> any 8080 (msg:"DOS Real Server template.html"; flow:to_server,established; content:"/viewsource/template.html?"; nocase; reference:cve,CVE-2000-0474; reference:bugtraq,1288; sid:278; rev:3;)
alert udp any any -> any 161 (msg:"DOS Bay/Nortel Nautica Marlin"; dsize:0; reference:bugtraq,1009; reference:cve,CVE-2000-0221; sid:279; rev:2;)
alert udp any any -> any 9 (msg:"DOS Ascend Route"; content: "|4e 41 4d 45 4e 41 4d 45|"; offset: 25; depth: 50; reference:bugtraq,714; reference:cve,CVE-1999-0060; reference:arachnids,262; sid:281; rev:2;)
alert tcp any any -> any 617 (msg:"DOS arkiea backup"; flow:to_server,established; dsize:>1445; reference:bugtraq,662; reference:cve,CVE-1999-0788; reference:arachnids,261; sid:282; rev:4;)
alert tcp any any -> any 139 (msg: "DOS Winnuke attack"; flags: U+; reference: bugtraq,2010; reference:cve,CVE-1999-0153; sid:1257; rev:3;)
alert tcp any any -> any 3372 (msg:"DOS MSDTC attempt"; flow:to_server,established; dsize:>1023; reference:bugtraq,4006; sid:1408; rev:5;)
alert tcp any any -> any 6004 (msg:"DOS iParty DOS attempt"; flow:to_server,established; content:"|FF FF FF FF FF FF|"; offset:0; reference:cve,CAN-1999-1566; sid:1605; rev:3;)

 # TFTP RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: tftp.rules,v 1.8 2002/08/18 20:28:43 cazz Exp $
#----------

#----------
#
These signatures are based on TFTP traffic. These include malicious files
that are distrubted via TFTP.
#
The last two signatures refer to generic GET and PUT via TFTP, which is
generally frowned upon on most networks, but may be used in some enviornments

alert udp any any -> any 69 (msg:"TFTP GET Admin.dll"; content: "|0001|"; offset:0; depth:2; content:"admin.dll"; nocase; reference:url,www.cert.org/advisories/CA-2001-26.html; sid:1289; rev:2;)
alert udp any any -> any 69 (msg:"TFTP GET nc.exe"; content: "|0001|"; offset:0; depth:2; content:"nc.exe"; nocase; sid:1441; rev:1;)
alert udp any any -> any 69 (msg:"TFTP GET shadow"; content: "|0001|"; offset:0; depth:2; content:"shadow"; nocase; sid:1442; rev:1;)
alert udp any any -> any 69 (msg:"TFTP GET passwd"; content: "|0001|"; offset:0; depth:2; content:"passwd"; nocase; sid:1443; rev:1;)
alert udp any any -> any 69 (msg:"TFTP parent directory"; content:".."; reference:arachnids,137; reference:cve,CVE-1999-0183; sid:519; rev:1;)
alert udp any any -> any 69 (msg:"TFTP root directory"; content:"|0001|/"; offset:0; depth:3; reference:arachnids,138; reference:cve,CVE-1999-0183; sid:520; rev:2;)
alert udp any any -> any 69 (msg:"TFTP Put"; content:"|00 02|"; offset:0; depth:2; reference:cve,CVE-1999-0183; reference:arachnids,148; sid:518; rev:3;)
alert udp any any -> any 69 (msg:"TFTP Get"; content:"|00 01|"; offset:0; depth:2; sid:1444; rev:2;)

 # SCAN RULES

(C) Copyright 2001,2002, Martin Roesch, Brian Caswell, et al.
All rights reserved.
$Id: scan.rules,v 1.17 2002/10/28 22:08:09 cazz Exp $
#----------
#----------
These signatures are representitive of network scanners. These include
port scanning, ip mapping, and various application scanners.
#
NOTE: This does NOT include web scanners such as whisker. Those are
in web*
#

alert tcp any 10101 -> any any (msg:"SCAN myscan"; ttl: >220; ack: 0; flags: S;reference:arachnids,439; sid:613; rev:1;)
alert tcp any any -> any 113 (msg:"SCAN ident version request"; flow:to_server,established; content: "VERSION|0A|"; depth: 16;reference:arachnids,303; sid:616; rev:3;)

alert tcp any any -> any 22 (msg:"SCAN ssh-research-scanner"; flow:to_server,established; content:"|00 00 00 60 00 00 00 00 00 00 00 00 01 00 00 00|"; sid:617; rev:2;)
alert tcp any any -> any 80 (msg:"SCAN cybercop os probe"; flags: SF12; dsize: 0; reference:arachnids,146; sid:619; rev:1;)
alert tcp any any -> any 3128 (msg:"SCAN Squid Proxy attempt"; flags:S; sid:618; rev:2;)
alert tcp any any -> any 1080 (msg:"SCAN SOCKS Proxy attempt"; flags:S; reference:url,help.undernet.org/proxyscan/; sid:615; rev:3;)
alert tcp any any -> any 8080 (msg:"SCAN Proxy \(8080\) attempt"; flags:S; sid:620; rev:2;)
alert tcp any any -> any any (msg:"SCAN FIN"; flags: F; reference:arachnids,27; sid:621; rev:1;)
alert tcp any any -> any any (msg:"SCAN ipEye SYN scan"; flags:S; seq:1958810375; reference:arachnids,236; sid:622; rev:2;)
alert tcp any any -> any any (msg:"SCAN NULL";flags:0; seq:0; ack:0; reference:arachnids,4; sid:623; rev:1;)
alert tcp any any -> any any (msg:"SCAN SYN FIN";flags:SF; reference:arachnids,198; sid:624; rev:1;)
alert tcp any any -> any any (msg:"SCAN XMAS";flags:SRAFPU; reference:arachnids,144; sid:625; rev:1;)
alert tcp any any -> any any (msg:"SCAN nmap XMAS";flags:FPU; reference:arachnids,30; sid:1228; rev:1;)
alert tcp any any -> any any (msg:"SCAN nmap TCP";flags:A;ack:0; reference:arachnids,28; sid:628; rev:1;)
alert tcp any any -> any any (msg:"SCAN nmap fingerprint attempt";flags:SFPU; reference:arachnids,05; sid:629; rev:1;)
alert tcp any any -> any any (msg:"SCAN synscan portscan"; id: 39426; flags: SF;reference:arachnids,441; sid:630; rev:1;)
alert tcp any any -> any any (msg:"SCAN cybercop os PA12 attempt"; content:"AAAAAAAAAAAAAAAA"; depth:16; flags:PA12; reference:arachnids,149; sid:626; rev:2;)
alert tcp any any -> any any (msg:"SCAN cybercop os SFU12 probe"; content: "AAAAAAAAAAAAAAAA"; depth:16; flags: SFU12; ack: 0; reference:arachnids,150; sid:627; rev:2;)
alert udp any any -> any 10080:10081 (msg:"SCAN Amanda client version request"; content:"Amanda"; nocase; sid:634; rev:2;)
alert udp any any -> any 49 (msg:"SCAN XTACACS logout"; content: "|8007 0000 0700 0004 0000 0000 00|";reference:arachnids,408; sid:635; rev:1;)
alert udp any any -> any 7 (msg:"SCAN cybercop udp bomb"; content:"cybercop"; reference:arachnids,363; sid:636; rev:1;)
alert udp any any -> any any (msg:"SCAN Webtrends Scanner UDP Probe"; content: "|0A|help|0A|quite|0A|"; reference:arachnids,308; sid:637; rev:2;)
alert tcp any any -> any 22 (msg:"SCAN SSH Version map attempt"; flow:to_server,established; content:"Version_Mapper"; nocase; sid:1638; rev:4;)