SISTEMA DE DETECCIÓN DE INTRUSOS UTILIZANDO REDES NEURONALES PARA LA RED DE DATOS DE LA UNIVERSIDAD DE MANIZALES

NELSON ANDRÉS GARCÍA ZULUAGA
ROBINSON SALGADO GIRALDO

UNIVERSIDAD DE MANIZALES
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
MANIZALES
2007

SISTEMA DE DETECCIÓN DE INTRUSOS UTILIZANDO REDES NEURONALES PARA LA RED DE DATOS DE LA UNIVERSIDAD DE MANIZALES

NELSON ANDRÉS GARCÍA ZULUAGA
ROBINSÓN SALGADO GIRALDO

Trabajo de grado para optar al título de Ingeniero de Sistemas y Telecomunicaciones

Presidente
Luís Carlos Correa Ortiz
Ingeniero Electrónico

UNIVERSIDAD DE MANIZALES
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
MANIZALES
2007

Nota de Aceptación:

_____
_____
_____
_____
_____
_____
_____
_____
_____

_____
Firma del Jurado

_____
Firma del Jurado

_____
Firma del Jurado

CONTENIDO

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMÉN

Por IDS* se entienden una serie de herramientas que al instalarse permiten monitorear y detectar posibles ataques informáticos al interior de una red de datos. El proyecto IDS Inteligente desarrolla en conjunto dos áreas de la informática: el desarrollo de software para redes de computadores y la inclusión de una técnica "inteligente"; una red neuronal perceptrón multicapa, las cuales, mediante una integración adquieren tramas en el tráfico de la red en la Universidad de Manizales para ser analizadas y obtener conclusiones acerca de intrusos y posibles ataques.

Su base tecnológica está soportada en las librerías de análisis JPCAP** con el objetivo de lograr la integración de una aplicación gráfica con la base fundamental de prueba de paquetes.

Como producto final se obtiene un software basado en entornos visuales gráficos, construido en Java bajo IDE NetBeans e integrado con una red neuronal para hacer el análisis de los datos generados. No obstante, el proyecto es un inicio de lo que puede ser un producto exitoso, dejando señalado un camino para que a futuro se complemente su base conceptual y se realicen futuros desarrollos.

* IDS: Sistema de Detección de Intrusos, por sus siglas en inglés, Intrusión Detection System.
** JPCAP: Definición original en Ingles: a network packet capture library for applications written in Java

ABSTRACT

IDS consist in a kit of tools that allow scan and detect possible attacks inside computer networks. The Intelligent IDS project develops as a whole two areas of the computer science: the technological development of software for networks and a multicap perceptron neuronal network that analyze plots in the traffic of the net in Universidad de Manizales to be analyzed and to obtain conclusions about intruders and possible attacks.

Its technological base is supported on the libraries of analysis JPCAP with the aim to achieve the integration of a graphical application with the fundamental base of packet testing.

As final product obtains software based on visual graphical environments, made in Java under the NetBeans IDE and repaid with an artificial neural network for analize the generated information. Nevertheless, the project is a beginning of what can be a successful product, makes a way notable in order that to future his conceptual base complements itself, and continue with further developments.

INTRODUCCION

Un IDS o Sistema de Detección de Intrusiones es una herramienta de seguridad informática que intenta detectar o monitorizar los eventos ocurridos en un determinado sistema informático o red informática en busca de intentos de comprometer la seguridad de dicho sistema. El proyecto IDS Inteligente pretende unir dos ramas de la informática: el estudio y análisis de tramas para un uso en particular y la aplicación de técnicas inteligentes para tales efectos.

El uso de herramientas existentes es el factor clave para lograr un buen desempeño en el producto; lo que se efectuó fue un inventario y procesamiento de plataformas, herramientas y librerías que puedan ser usadas en el proyecto, es entonces, como se logra adaptar el conjunto de librerías JPCAP para tratamiento de paquetes.

En conjunto es necesario integrar el análisis de tramas con un modelo inteligente para lograr una mayor efectividad y lograr el objetivo básico del IDS, para esto, se adapta e implementa una herramienta que permita exportar y analizar las tramas “sospechosas” basando sus técnicas en la Inteligencia Artificial.

Como producto final, se tiene un software basado en tecnologías de última generación que captura tramas en una red de datos tradicional, las analiza y efectúa un procedimiento inteligente para identificar posibles nuevos ataques, todo basado en el análisis de puertos y una red neuronal perceptrón multicapa.

1. DESCRIPCIÓN DEL PROBLEMA

La vulnerabilidad en los sistemas computacionales tanto personales como empresariales se ha convertido en una deficiencia de seguridad relacionada directamente con los tipos de software producidos por las grandes compañías. El incremento de intrusiones y malos usos en los sistemas informáticos y redes internas de una gran cantidad de empresas, ha provocado un aumento en la preocupación por la seguridad informática. Desde hace algún tiempo se vienen aplicando medidas basadas en software y hardware para combatir dicho problema. La Universidad de Manizales cuenta con una serie de sistemas de información vitales para ella, desde la información financiera de la institución hasta la información académica de los estudiantes.

Debido a esta situación, se genera la necesidad de implementar constantemente nuevas estrategias de seguridad a sus sistemas informáticos. Es posible, si se determina la naturaleza de un determinado paquete que circula por la red, implementar herramientas para detectar su peligrosidad y prevenir ataques y vulnerabilidades a una red de datos. A este tipo de sistemas se les conoce como IDS, descritos en la introducción

Por esto se propone el desarrollo de un IDS mediante el uso de redes neuronales, en particular en una red perceptrón multicapa, y contrastar su desempeño con un sistema tradicional basado en reglas de comportamiento de acuerdo con diferentes patrones de los paquetes.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Diseñar, simular e implementar un sistema de detección de paquetes que representen ataques a los sistemas de información de la Universidad de Manizales, que involucre una red neuronal perceptrón multicapa.

2.2 OBJETIVOS ESPECÍFICOS

- ✓ Adquirir y caracterizar una muestra significativa de paquetes normales y que representen ataques en la red de datos de la U de Manizales.

- ✓ Entrenar una red neuronal perceptrón multicapa a partir de los datos recolectados.

- ✓ Implementar un sistema de detección de paquetes que incorpore la red neuronal entrenada y realizar pruebas que validen su comportamiento adecuado.

3. JUSTIFICACIÓN

Debido a su constitución y a sus fundamentos, las redes neuronales artificiales presentan un gran número de características semejantes a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan información irrelevante, etc. Esto hace que ofrezcan numerosas ventajas y que este tipo de tecnología pueda ser aplicado en una red de información de tanta importancia como es la de la Universidad de Manizales.

Un sistema seguro debe contar con los siguientes aspectos: La confidencialidad hace referencia a que los objetos de un sistema han de ser accedidos únicamente por elementos autorizados a ello, y que esos elementos autorizados no van a convertir esa información en disponible para otras entidades; la integridad significa que los objetos sólo pueden ser modificados por elementos autorizados, y de una manera controlada. Por último, la disponibilidad indica que los objetos del sistema tienen que permanecer accesibles a elementos autorizados; es el contrario de la negación de servicio.

Estas razones justifican la implementación de un IDS, y en particular la investigación y desarrollo en seguridad haciendo uso de esta estrategia.

4. MARCO TEÓRICO

A continuación se muestran los elementos conceptuales e históricos que soportan el presente desarrollo, tales como los Sistemas de Detección de Intrusos y las Redes Neuronales Artificiales.

Históricamente las empresas han buscado una solución a la sistematización de sus procesos, y normalmente instalan software que les permita lograr seguridad, agilidad y control sobre sus procesos; en el caso de las universidades se tiene una situación particular: sus procesos son muy diversos y complejos, basados en lo disímil de su objeto social, hecho que las convierte en un foco problemático en el mundo de la informática.

Basados en la naturaleza de los procesos a sistematizar, la Universidad tiene las siguientes opciones para buscar solución a toda la problemática:

- ✓ Contratación: mediante la modalidad de outsourcing, la compañía puede contratar los servicios de una persona natural o empresas para que se encargue de dar solución a la problemática existente.

- ✓ Compra de software: consiste en la compra de un producto ya hecho, en el mercado es difícil de encontrar un software que cumpla con las características actuales respecto a los procedimientos llevados a cabo en el momento de captar tramas en la red para ser analizas en forma inteligente para la detección de intrusos.

- ✓ Diseño e Implementación de Software Propio: cuando la Universidad cuenta con una serie de programas académicos en informática, puede construir por si misma el software necesario para dar solución a sus problemáticas, sea éste el caso de la Universidad de Manizales.

4.1 IDS

Los IDS son sistemas de seguridad informática. La Universidad de Manizales es líder en la región en estas temáticas, prestigio ganado mediante la realización de eventos de carácter regional, nacional e internacional. Sus docentes fueron consultados al respecto, con el fin de realizar una primera aproximación a las herramientas existentes. Se llego a la conclusión que dichos sistemas se encuentran por separado, es decir, los IDS están muy desarrollados, existiendo

gran variedad de productos, tanto en software propietario como software libre; y por otro lado se encuentran muchas aplicaciones en inteligencia artificial, ya sea en aplicaciones o en modelos matemáticos. El IDS desarrollado está fundamentado en ambas temáticas.

Hoy en día existen herramientas de administración de redes, pero ninguna cumple los tres aspectos que cumple JDumper[***] (Capturar tramas, analizarlas por reglas y por RNA). Generalmente implementan uno solo de los aspectos, y en todo caso dos para los monitores, porque necesitan datos de los paquetes de red e incluyen para ello un pequeño capturador de paquetes de red o sniffer.

Ya que el aumento en la inseguridad de las redes de datos va creciendo cada día más es necesario para cualquier entidad publica o privada contar con un IDS, ya que la gravedad de estos ataques es cada día mas grave y mas aun cuando se cuenta con un IDS que use RNA[****] Hay que tener en cuenta que a veces los diferentes IDS instalados en las empresas pueden generar una saturación a la red que comienzan a generar falsos positivos que son especies de ataques detectados por el IDS que en verdad son inofensivos, con este IDS trataremos de reducir esto al máximo.

Un IDS cuenta con un motor de análisis el cual analiza los numerosos eventos que ocurren en la red y los evalúa en su contexto. Puede operar tanto fuera de línea (offline), capturando los paquetes y analizándolos posteriormente, como en línea (online) este seria crítico para lanzar una respuesta rápida y efectiva a los ataques. Los IDS aportan a nuestra seguridad una capacidad de prevención y de alerta anticipada ante cualquier actividad sospechosa. No están diseñados para detener un ataque, aunque sí pueden generar ciertos tipos de respuesta ante éstos.

Se debe tener muy en cuenta si la instalación del IDS se realiza de manera pasiva o de manera activa, la diferencia entre estas es que la pasiva almacena la alerta en un blog y con la posibilidad de enviarlo a una base de datos con el fin de dar parte al administrador de la red de las anomalías que se presentaron y la activa genera la activación del antivirus para cortar el trafico generado por la red atacante.

[***] Nombre del proyecto adoptado (con previo análisis)
[****] Redes Neuronales Artificiales

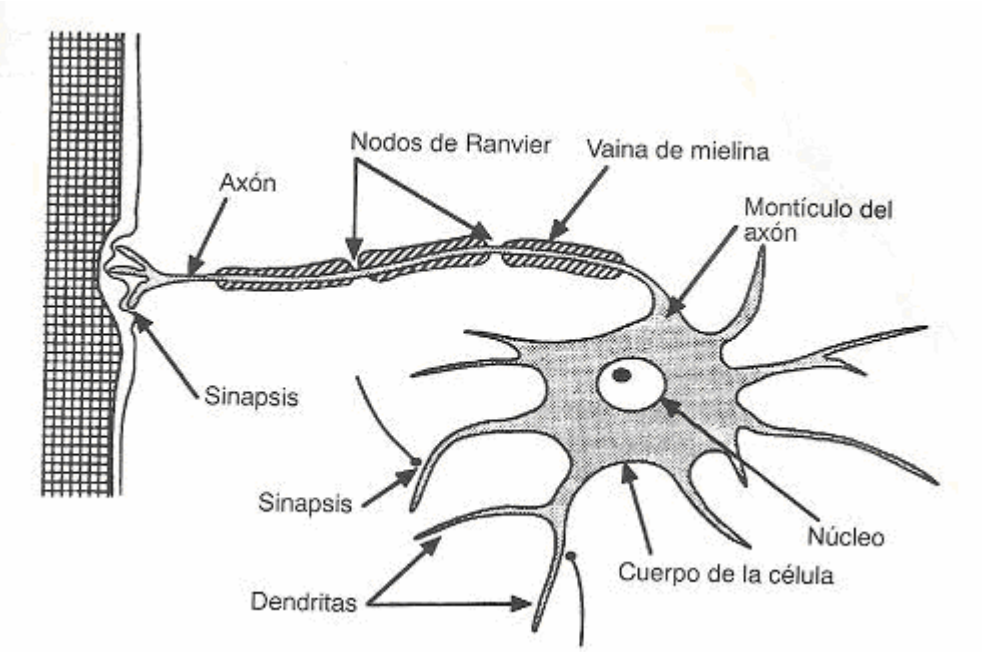
4.2 RED NEURONAL

Las Redes Neuronales surgieron del movimiento conexionista, que nació junto con la IA simbólica o tradicional. Esto fue hacia los años 50, con algunos de los primeros ordenadores de la época y las posibilidades que ofrecían. La IA simbólica se basa en que todo conocimiento se puede representar mediante combinaciones de símbolos, derivadas de otras combinaciones que representan verdades incuestionables o axiomas. Este paradigma se expuso por primera vez en el año de 1943, en un artículo escrito por McCulloch y Pitts, en el cual realizan una asociación simbólica del modelo neuronal biológico vigente en ese entonces. Las redes neuronales se usan principalmente en problemas que involucran el reconocimiento de patrones y la clasificación, entendiendo patrón como instrumento de medida o medida materializada el cual se encarga de conservar unos valores para luego servir como referencia.

4.2.1 Modelo Biológico: Una neurona es una célula viva, y como tal, contiene los mismos elementos que forman parte de todas las células biológicas, además, de poseer elementos característicos que la diferencian. En general una neurona consta de un cuerpo celular más o menos esférico de 5 a 10 micras de diámetro, del que sale una rama principal el axón, y varias ramas más cortas denominadas dendritas. A su vez el axón puede producir ramas en torno a su punto de arranque, y con frecuencia se ramifica extensamente cerca de su extremo. La neurona está formada por el cuerpo de la célula, compuesta por el núcleo el cual posee funciones de control.

Las dendritas esta compuesta por una red alrededor de la célula, el axón se extiende generalmente un centímetro y en algunas ocasiones no muy explicitas hasta un metro. Este también se ramifica en filamentos y subfilamentos mediante los que establece conexión con las dendritas y los cuerpos de las células de otras neuronas. A esta conexión se le conoce como sinapsis. Cada neurona puede establecer conexión desde una docena de neuronas hasta miles y miles de neuronas dependiendo su comportamiento, así como se puede observar en la figura 1. Las neuronas, por medio de las dendritas y el axón, reciben señales eléctricas, impulsos de neuronas vecinas, con el fin de que estas puedan excitarse, estimulando impulsos a otras neuronas, para que estas lo puedan recibir y posteriormente excitarse, y de esta forma comunicarse entre todas las neuronas.

Figura 1. Modelo Biológico.



4.2.2 Unidad de Procesamiento de una Red Neuronal: Los elementos individuales de cálculo que forman la mayoría de los modelos de sistemas neuronales artificiales no suelen denominarse *neuronas artificiales*; lo más frecuente es darles el nombre de nodos, unidades o elementos de procesamiento. Cualquier modelo de red neuronal consta de dispositivos elementales de procesamiento: las neuronas. A partir de ellas, se pueden generar representaciones específicas, de tal forma que un estado conjunto de ellas pueda significar una letra, un número o cualquier otro objeto. Su procesamiento es el siguiente:

- Las neuronas reciben estímulos externos, relacionadas con el aparato sensorial, que tomarán la información de la entrada, denominadas unidades de entrada.

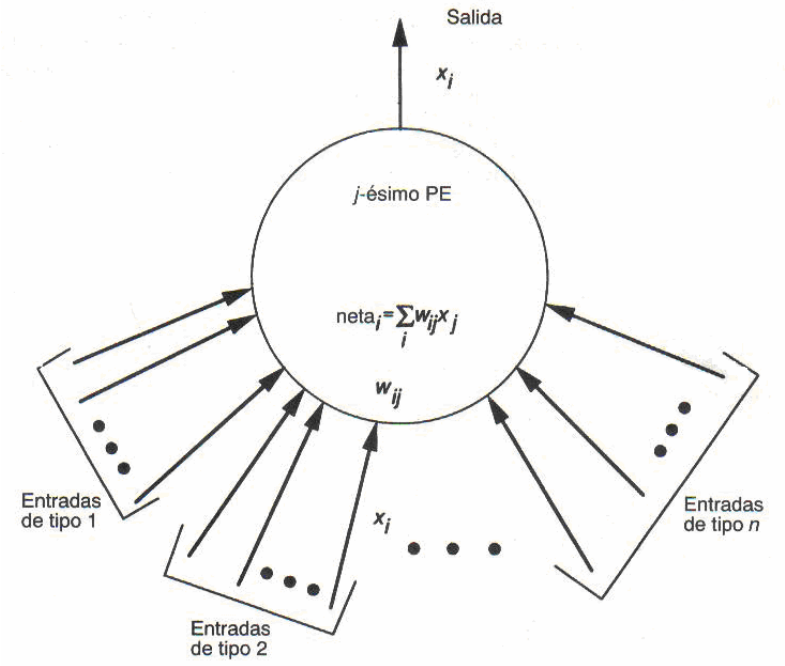
- Dicha información se transmite a ciertos elementos internos que se ocupan de su proceso. Es la sinapsis y neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de la información. Puesto que no tienen relación directa con la información de entrada y de salida, estos elementos se denominan unidades ocultas.

- Una vez ha finalizado el periodo de procesado, la información llega a las unidades de salida, cuya misión es dar la respuesta del sistema.

La tecnología basada en redes neuronales artificiales trata de imitar el funcionamiento de elementos biológicos que en este caso los constituyen las neuronas del cerebro.

El cerebro humano en su biología está formado por miles de millones de neuronas que se conectan entre sí, transmitiendo información entre ellas, luego de que esta procesa, se genera una respuesta en función del estímulo recibido. Si se tienen N unidades (neuronas), ordenadas arbitrariamente, su trabajo es simple y único, consiste en recibir las entradas de las células vecinas y calcular un valor de salida, el cual es enviado a todas las células restantes (neuronas).

Figura 2. Neurona Artificial



De esta manera se puede determinar los elementos principales que constituyen una red neuronal artificial de la siguiente manera:

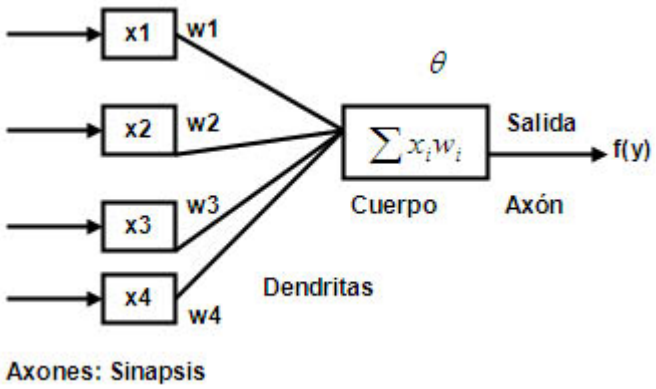
- Las conexiones que unen a las neuronas de una RNA tienen asociado un **peso**, que es el que hace que la red adquiera un conocimiento y una cantidad determinada.

- Una neurona o unidad de procesamiento se ve afectada por las salidas de las neuronas con las cuales ella esta conectada. El efecto total de las salidas de estas neuronas reflejado en la unidad de procesamiento que se esté analizando se conoce como **neta**.

- Entre las unidades o neuronas que forman una red neuronal artificial existe un conjunto de conexiones que las unen. Cada unidad transmite señales a aquellas que están conectadas a su salida. Asociada con cada unidad hay una **función de activación**, que transforma la entrada neta que la unidad presenta como resultado de estar conectada con otras unidades que le suministran información, en el valor de la salida.

De esta manera la red neuronal consiste únicamente en recibir las entradas de las neuronas vecinas y calcular un valor de salida, que es enviado a todas las neuronas restantes como se observa en la figura 2.

Figura 3. Funcionamiento Neurona Artificial



Las unidades de entrada reciben las señales desde el exterior, es decir de otras neuronas vecinas. Estas entradas pueden provenir de sensores o de otros sectores del sistema las cuales servirán para la activación de la red. Las unidades de salida envían la señal fuera del sistema neuronal, las cuales servirán de entrada a otra neurona, a través de la sinapsis. Estas salidas pueden controlar directamente potencias u otros sistemas. En el cuerpo de la neurona, es donde se produce la función de activación que va permitir que esta envíe la información a otras neuronas y puedan comunicarse. Esta activación se produce al sumar todas sus entradas por sus pesos y su respectivo umbral de activación.

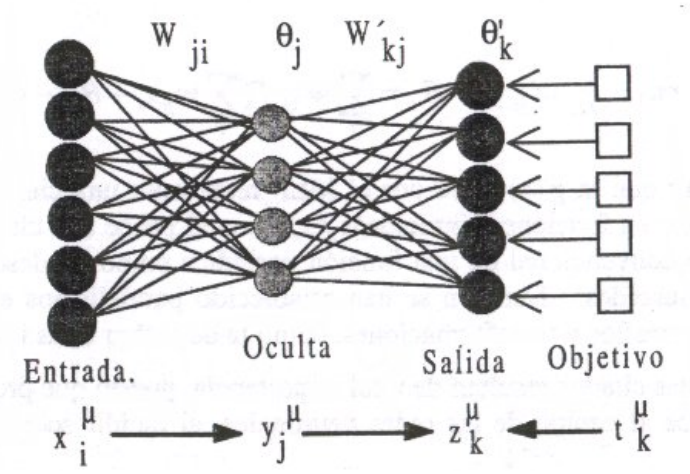
4.2.3 Aprendizaje de una Red Neuronal: La modificación del comportamiento provocado por la interacción con el entorno y como resultado de experiencias adecuadas al establecimiento de nuevos modelos de respuestas ante estímulos externos. En el caso de las redes neuronales artificiales, se puede considerar que el conocimiento se encuentra representado en los pesos de las conexiones entre las neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad se puede decir que se aprende modificando los pesos de la red. De igual manera existen algoritmos de aprendizaje que ayudan al desempeño de la red neuronal, los algoritmos son los siguientes:

- Supervisado: Este algoritmo produce una función que establece una concordancia entre deseada del sistema entre las entradas y las salidas.
- No supervisado: Es donde el modelamiento se realiza sobre unos ejemplos formados sólo por entradas al sistema.
- Por refuerzo: Este algoritmo aprende alimentándose del entorno o del mundo que lo rodea.
- Transducción: Como el supervisado pero no construye una función.
- Multi-tarea: Usan conocimiento previamente aprendido por el sistema de cara a enfrentarse a problemas parecidos a los ya vistos.

4.2.4 El perceptrón multicapa[1] (MLP): Este es uno de los tipos de redes más comunes. Se basa en otra red mas simple llamada perceptrón simple solo que el número de capas ocultas puede ser mayor o igual que una. Es una red unidireccional (feedforward), cuya arquitectura puede observarse en la Figura 4.

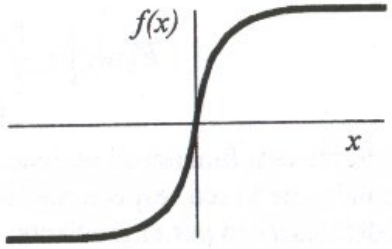
[1] PALACIOS, Francisco. Herramientas en GNU/Linux para estudiantes universitarios. Tipos de Redes Neuronales. [En Línea]. 2003. Disponible en: http://softwarelibre.unsa.edu.ar/docs/descarga/2003/curso/htmls/redes_neuronales/index.html

Figura 4. Representación de un Perceptrón Multicapa (MLP)



Las neuronas de la capa oculta usan como regla de propagación la suma ponderada de las entradas con los pesos sinápticos wij y sobre esa suma ponderada se aplica una función de transferencia de tipo sigmoide, denomina así por su forma de ***ese,*** la cual es acotada en respuesta y útil para ampliar el rango de respuesta de las capas ocultas a valores continuos entre 0 y 1; sobre la función *hardlimit,* la cual solo tiene dos posibles valores 0 o 1. (Figura 5).

Figura 5. Forma funcional de una sigmoide



El aprendizaje supervisado que se suele usar en este tipo de redes recibe el nombre de retropropagacion del error (backpropagation), debido a que, inicialmente se calcula la salida de la red y se compara con la salida deseada. Posteriormente, este error se utiliza para cambiar los pesos de las entradas de las neuronas desde la última capa hasta la primera, de ahí su nombre. Como función de rendimiento global, se usa el error cuadrático medio. Es decir, que dado un par

($x_{.f.}$, d_k) correspondiente a la entrada *k* de los datos de entrenamiento se obtiene la salida deseada asociada se calcula el error mediante la siguiente ecuación:

$$E(w_{ij}, \theta_j, w'_{kj}, \theta'_k) = \frac{1}{2} \sum_p \sum_k \left[d_k^p - f\left(\sum_j w'_{kj} y_j^p - \theta'_k \right) \right]^2 \quad [1]$$

Donde se observa que es la suma de los errores parciales debido a cada patrón (índice *p*), resultantes de la diferencia entre la salida deseada d_p y la salida que da la red f(.) ante el vector de entrada x_k. Si estas salidas son muy diferentes de las salidas deseadas, el error cuadrático medio será grande. Por último, *f* es la función de activación de las neuronas de la capa de salida *E* y la salida que proporcionan las neuronas de la ultima capa oculta.

Sobre esta función de coste global se aplica algún procedimiento de minimización. En el caso del MLP se hace mediante un descenso por gradiente. Las expresiones que resultan aplicando la regla de la cadena son las siguientes:

$$\begin{aligned}
\delta w'_{kj} &= -\epsilon \frac{\partial E}{\partial w'_{kj}} \\
\delta w'_{ji} &= -\epsilon \frac{\partial E}{\partial w_{ji}} \\
\delta w'_{kj} &= \epsilon \sum_p \Delta'^p{}_k y_j^p \qquad con \qquad \Delta'^p{}_k = [d_k^p - f(v'^p{}_k)] \frac{\partial f(v'^p{}_k)}{\partial v'^p{}_k} \\
\delta w_{ij} &= \epsilon \sum_p \Delta_j^p x_i^p \qquad con \qquad \Delta_j^p = \left(\sum_k \Delta'^p{}_k w'_{kj} \right) \frac{\partial f(v_j^p)}{\partial v_j^p}
\end{aligned} \quad [2]$$

Siendo y(k) las salidas de la capa oculta.

El aprendizaje por backpropagation puede describirse mediante el siguiente procedimiento:

- ✓ Inicializar los pesos y los umbrales iniciales de cada neurona. Hay varias posibilidades de inicialización siendo las más comunes las que introducen valores aleatorios pequeños.
- ✓ Para cada patrón del conjunto de los datos de entrenamiento.
- ✓ Obtener la respuesta de la red ante ese patrón. Esta parte se consigue propagando la entrada hacia adelante, ya que este tipo de red es feedforward. Las salidas de una capa sirven como entrada a las neuronas de la capa siguiente, procesándolas de acuerdo a la regla de propagación y la función de activación correspondientes.
- ✓ Calcular los errores asociados según la ecuación [1].
- ✓ Calcular los incrementos parciales (sumandos de las sumatorias). Estos incrementos dependen de los errores calculados en [1].
- ✓ Calcular el incremento total para todos los patrones, de los pesos y los umbrales según las expresiones en la ecuación [2].
- ✓ Actualizar pesos y umbrales.
- ✓ Calcular el error actual y volver al paso 2 si no es satisfactorio.

Este entrenamiento se realizó utilizando Neural Networks Toolbox, incluido en el paquete MATLAB®, el cual posee los algoritmos implementados que ejecutan los pasos descritos anteriormente, a partir de los datos capturados por el IDS desarrollado.

4.2. LIBRERÍA JPCAP

JPCAP es una librería de Java para capturar y enviar paquetes en una red; usando JPCAP se pueden desarrollar aplicaciones para capturar paquetes desde una interfaz de red, visualizándolos y analizándolos en Java. También es posible desarrollar aplicaciones para enviar arbitrariamente paquetes a través de una interfaz de red.[2]

JPCAP ha sido evaluado en Microsoft Windows (98/2000/XP/Vista), Linux (Fedora, Mandriva, Ubuntu), Mac OS X (Darwin), FreeBSD y Solaris. Esta librería suministra facilidades para:

- ✓ Capturar paquetes desconocidos desde el medio de transmisión.
- ✓ Guardar paquetes capturados a un archivo fuera de línea, y leer paquetes capturados desde un archivo previamente generado. Identificar

[2] JPCAP. A Java library for capturing and sending network packets. [En Línea]. Fecha de Consulta: 03-08-2007. Home. Disponible en: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/

automáticamente tipos de paquetes y generar objetos correspondientes de Java (para Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, y paquetes ICMPv4).
- ✓ Acordar filtros de paquetes para usuarios con reglas específicas antes de despacharlos de la aplicación.
- ✓ Enviar paquetes desconocidos a la red.

4.3. ATAQUES

Dentro de los ataques hechos al sistema se destacan, virus troyanos y escaneo de puertos.

Virus troyanos: Un troyano puede definirse como:

> "Un troyano o caballo de Troya es un programa que se diferencia de los virus en que no se reproduce infectando otros ficheros. Tampoco se propaga haciendo copias de sí mismo como hacen los gusanos. Su nombre deriva del parecido en su forma de actuar con los astutos griegos de la mitología. Llegan al ordenador como un programa aparentemente inofensivo, pero al ejecutarlo instala en el ordenador un segundo programa: el troyano"[3].

Los efectos de los troyanos pueden ser muy peligrosos. Permiten realizar intrusiones o ataques contra el ordenador afectado, realizando acciones tales como capturar todos los textos introducidos mediante el teclado o registrar las contraseñas introducidas por el usuario.[4] Los troyanos utilizados para el ataque fueron los siguientes:

- ✓ SubSARI
- ✓ Deep Throat, Foreplay
- ✓ Arctic
- ✓ DRAT
- ✓ ADM worm, Lion
- ✓ DMSetup
- ✓ BackGate
- ✓ CDK, Firehotcker
- ✓ RemoConChubo
- ✓ Hidden
- ✓ ProMail trojan
- ✓ Invisible Identd Deamon, Kazimas
- ✓ Happy99

---

[3] SEGURIDAD EN LA RED.Virus, gusanos, troyanos y backdoors. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en: http://www.seguridadenlared.org/es/index10esp.html

[4] Ibid.

- ✓ Attack Bot, God Message, JammerKillah
- ✓ Net Controller
- ✓ Farnaz
- ✓ Chode
- ✓ Msinit, Qaz
- ✓ Back Orifice
- ✓ Chode
- ✓ Chode, God Message worm, Msinit
- ✓ Arctic

Escaneo de puertos: El escaneo de puertos puede definirse como:

> "El escaneo de puertos es una de las más populares técnicas utilizadas para descubrir y mapear servicios que están escuchando en un puerto determinado. Usando este método un atacante puede crear una lista de las potenciales debilidades y vulnerabilidades en un puerto para dirigirse a la explotación del mismo y comprometer el host remoto"[5].

4.4. METODOLOGÍA DE DESARROLLO

El desarrollo del proyecto se guió por la metodología de Proceso Unificado (UP), mientras el lenguaje para el proceso de Ingeniería del Software es el Lenguaje Unificado de Modelado, UML. El desarrollo se implementó usando Java, bajo el uso y los lineamientos del IDE NetBeans. A continuación se realiza una breve descripción de estos elementos:

4.4.1. Proceso Unificado: El Proceso Unificado se lleva a cabo en las siguientes fases:

1. Modelo de Requisitos
2. Modelo de Análisis
3. Modelo de Diseño
4. Pruebas
5. Implementación
6. Gestión de la configuración.

[5] IT PRO MEXICO. Métodos de Escaneo de Puertos. [En Línea]. Fecha de Consulta: 03-08-2007.Inicio. Disponible en: http://www.itpromexico.com.mx/paginas/articulos/scanportmet.htm

E incluye los siguientes artefactos relevantes:

- ✓ Modelo de casos de uso: Incluye los casos de uso, el diagrama de los mismos y los diagramas de diseño del sistema.
- ✓ Modelo de Dominio.
- ✓ Modelo de clases
- ✓ Diagrama de estados del sistema
- ✓ Diagramas de arquitectura
- ✓ Diagramas de Actividad y Colaboración

4.4.2. Lenguaje Unificado de Modelado (UML): Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el OMG* Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables[6].

4.4.3. NetBeans. NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un Entorno integrado de desarrollo (IDE) desarrollado usando la Plataforma NetBeans. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software[7].

4.4.4. Java: Java es un lenguaje de programación orientado a objetos desarrollado por James Gosling y sus compañeros de Sun Microsystems al inicio de la década de 1990. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado (usando normalmente un compilador JIT), por una máquina virtual Java.

---

* Object Management Group.

[6] OBJECT MANAGEMENT GROUP. Introduction to OMG's Unified Modeling Language™ (UML®). Fecha de Consulta: 03-08-2007. Inicio > Introduction to UML. Disponible en: http://www.omg.org/gettingstarted/what_is_uml.htm

[7] NETBEANS.ORG. Bienvenido A Netbeans. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en: http://www.netbeans.org/index_es.html.

4.4.5. Rational Case: Rational es actualmente conocida como una familia de software de IBM para el levantamiento de requerimientos, diseño, construcción, pruebas y administración de proyectos en el proceso desarrollo de software. Las herramientas de Rational dan grandes beneficios sin limitar la libertad creativa de los equipos de desarrollo. Permite seleccionar el nivel de abstracción correcto para cada tarea. Utiliza el ambiente de desarrollo y el IDE[8] preferido y adecuado para el equipo. Adapta el ambiente del proyecto a las necesidades específicas. No importa cuál sea la plataforma de desarrollo o las funciones dentro del equipo (rol), hay alguna solución de Rational que puede ayudar.

Las herramientas de Rational fortalecen el desarrollo con Java con la principal plataforma de desarrollo para la comunidad Java. Nuestras herramientas galardonadas, te permitirán trabajar intuitivamente, con mayor control, y con el conjunto más amplio de recursos Java. A partir de la adquisición de Rational por IBM, la integración de las herramientas ha sido sin precedente, ofreciendo una gran facilidad de uso. Otras alianzas estratégicas con Sun Microsystems, HP, y otros asegura una compatibilidad total (end-to-end) en tu ambiente de desarrollo Java seleccionado.

4.4.6. Back Orifice y Megaping

Back Orifice es una herramienta que fue creada por una organización de Hackers en 1998, en el cual funciona como cliente – servidor. El programa contiene una serie de subprogramas que son instalados en el cliente el cual va a ser atacado, estos subprogramas también llamados troyanos son enviados a través de la red e instalados en el equipo que se ejecuten. Ya instalados estos troyanos (virus), se procede en el servidor, a través de plugins que contiene varias características, a realizar el debido daño. [9]

Por otra parte *Megaping* es un software comercial que sirve para la visión general de la configuración de un sistema y la administración física de una red en tiempo real, en este caso se utilizó una versión *shareware* como escaneador de puertos relevantes e irrelevantes de la red de datos.[10]

---

[8] Ambiente Integrado de Desarrollo

[9] BLOG DE SOFTWARE GENBETA. Back Orifice: especial software control remoto. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en: http://www.genbeta.com/2006/06/13-back-orifice-especial-software-control-remoto

[10] PIOJOSOFT - EL SITIO DE DESCARGAS. Megapin 4.3. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en http://www.piojosoft.com/index.php?empezar=0&ide=200

5. METODOLOGÍA

Para llegar a determinar el funcionamiento y el análisis del proyecto, se tuvieron encuentra algunas aplicaciones de desarrollo libre como Fedora que es una distribución de Linux, desarrollada y patrocinado por Red Hat. Con esta distribución se instalaron programas de captura de tramas como Nmap, que permitió determinar paquetes vulnerables en la red y así clasificar los paquetes a utilizar en el proyecto.

De igual manera se utilizaron bajo esta distribución, Windows XP y Windows 2003 Server aplicaciones denominadas sniffer (específicamente *Snort* y *Ethereal*) para la captura de paquetes y detección de intrusos ya que monitorizan todo un dominio de una red y son flexibles para el almacenamiento de sus datos.

De acuerdo con lo anterior se determinó que las herramientas utilizadas no fueron útiles para el presente proyecto, porque era necesario capturar datos propios en la aplicación final.

Como procedimiento final se llevaron acabo las siguientes fases.

5.1. FASE 1: RECOLECCIÓN DE INFORMACIÓN.

Se determinó que los paquetes más relevantes para la adquisición son aquellos relacionados con los protocolos UDP y TCP. El protocolo UDP proporciona una comunicación muy sencilla entre los ordenadores y en la captura de paquetes se obtuvieron tramas con mensajes y datos perdidos y dañados. A través del protocolo TCP se determinó que en la captura de la trama los datos siempre llegaron a la aplicación de destino ya que por medio de este la información llega de forma de correcta.

Para el entrenamiento de la red neuronal se utilizó la aplicación MATLAB®, la cual trae una serie de herramientas denominadas *Toolbox*, específicamente el Neural Networks Toolbox para entrenar una red neuronal perceptrón multicapa.

5.2. FASE 2: ADQUISICIÓN DE LOS PAQUETES.

Al ser necesario la captura de paquetes desde una aplicación propia, para su posterior clasificación, se utilizó la librería JPCAP, la cual permite capturar los paquetes recibidos por las tarjetas de red de un computador. Posteriormente se

Generaron los ataques previamente mencionados mediante las aplicaciones *Back Orifice y Megaping* para así obtener los datos para el posterior entrenamiento y validación de la red neuronal multicapa. Además para determinar cuales paquetes son nocivos, se establecieron una serie de reglas. Esto permite una comparación entre lo detectado mediante dichas reglas y lo obtenido usando la red neuronal.

5.2. FASE 3: ENTRENAMIENTO DE LA RED NEURONAL.

MATLAB® permite entrenar redes neuronales mostrándole pares de entrada y salida para que calcule los pesos de la red neuronal de tal forma que cuando se le presente una entrada, arroje la salida correspondiente, y así cuando se le presente una entrada que no conocía generalice y obtenga una salida acorde con el entrenamiento. Para este entrenamiento se tuvieron en cuenta las características y variables propias de la arquitectura de la red neuronal y que son arrojadas en la configuración de la red las cuales son:

- ✓ Entradas: Par de datos que alimentan la red neuronal. En este caso las entradas son el puerto y el protocolo, los cuales se transformaron en variables numéricas. Se determina si el paquete es peligroso con reglas previamente estudiadas, las cuales involucran como patrones, el puerto y el protocolo.
- ✓ Capa: Conjunto de neuronas que comparten las mismas entradas. En este caso, después de múltiples pruebas, se obtuvo una red con dos capas, veinte neuronas en la primera capa y una única neurona de salida en la segunda capa.
- ✓ Pesos: Valores que ponderan las entradas. En este caso se tienen dos conjuntos de pesos, lo que ponderan las entradas principales a la Capa 1 –W1-, y los que ponderan la salida de dicha capa, que son a su vez entradas para la Capa 2 compuesta de una única neurona –W2-.
- ✓ Bias: Pesos adicionales que suponen una entrada 1. Se utilizan para dar mayor grado de libertad al entrenamiento. Se tienen dos conjuntos de bias, los correspondientes a la Capa 1, 20 valores en total y un único valor correspondiente a la neurona de la Capa 2.

El entrenamiento se realizó utilizando la interfaz gráfica del *Neural Networks Toolbox*, denominada *nntool*, la cual permite cargar los datos obtenidos en la captura desde un archivo plano. Posteriormente se definen que datos de los importados, corresponden a las entradas de la red neuronal, y cuales a la salida conocida, es decir, si es o no un paquete peligroso (uno, si en un paquete normal, y cero si es peligroso). Como parámetros de entrenamiento se tuvieron diez mil épocas y un error máximo de 0.01, lo que significa que las posibles salidas (cero o uno) podrán convertirse en valores por encima o por debajo 0.01 de ese valor.

Una vez realizado el entrenamiento, MATLAB® arroja los valores de los pesos en ambas capas y los bias, los cuales se validaron para diferentes valores de entrada tal como se muestra a continuación:

EntradaP= [139 6];
*salida1=tansig (EntradaP*W1'+B1');*

En esta línea se multiplica la entrada (puerto y protocolo) por los pesos de la capa uno, y se le suma el bias de la capa uno. A ese resultado se le saca la tangente sigmoide, que es la función de activación utilizada en este caso, y una de las más comunes en redes *feedforward* y en perceptrones multicapa. La función tangente sigmoide se implemento en la aplicación utilizando la función tangente hiperbólica.

*salida2=logsig (salida1*W2'+B2);*

En esta línea se multiplica la salida de la capa uno por los pesos de la capa dos, y se le suma el bias de la capa dos, para posteriormente sacarle a ese único valor la logaritmo sigmoide, y así obtener la salida definitiva de la red, que será uno o cero de acuerdo con la información inicial introducida.

5.3. FASE 4: ANÁLISIS Y DISEÑO DEL SISTEMA

El análisis y diseño se implemento bajo la metodología UP (Proceso Unificado), el lenguaje para el proceso de Ingeniería del Software es el lenguaje Unificado de Modelado UML y su programación se desarrollo en lenguaje Java, bajo el uso y los lineamientos de NetBeans. Los anexos A y B contienen los diversos diagramas que ilustran este proceso.

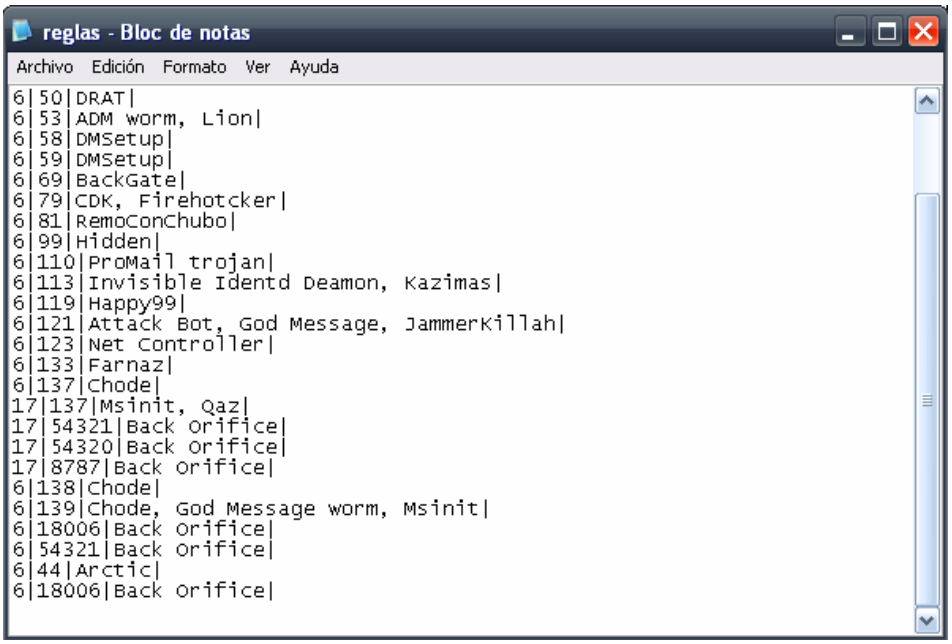
5.4. FASE 5: IMPLEMENTACIÓN Y PRUEBAS

La implementación se realizó usando NetBeans, y para realizar las pruebas se capturaron 856 tramas clasificadas en protocolos UDP y TCP. La clasificación de dichos paquetes se realizó tanto mediante unas reglas establecidas como se muestra en la figura 6, así como mediante una red neuronal. Se realizó un cálculo de la efectividad de la red neuronal comparada con las reglas predefinidas.

De las 856 tramas clasificadas se entreno la red neuronal en MATLAB® con los primeros 400 campos, ya que era necesario saber si los resultados obtenidos por la red neuronal eran validos al evaluarse con tramas que no se usaron en el entrenamiento. Por esto se dejaron 456 tramas de prueba y se observo que los resultados eran satisfactorios.

En la figura 6 que se muestra a continuación se puede observar que las reglas están implementadas dentro de un archivo plano, donde existen 3 campos: protocolo, puerto y descripción del ataque respectivamente. La primera regla que se encuentra en el archivo esta constituida de la siguiente manera: 6 | 50 | Drat |, donde el 6 es el protocolo (UDP), 50 el puerto por donde se vulnero el sistema y Drat el tipo de ataque que se quiso activar en el equipo.

Figura 6. Reglas



6. RESULTADOS

Como principal resultado de la investigación se obtuvo un sistema donde se pueden capturar todos los datos de una trama, pero teniendo en cuenta que son los más significativos ya que como mencionados anteriormente solo se utilizaron ciertas características para el IDS. A continuación se nombraran los resultados obtenidos del proyecto de Investigación:

- La aplicación cumplió con el objetivo principal: detectar intrusos por medio de redes neuronales y determinar por medio de reglas la veracidad de su funcionamiento así como se puede observar en la figura 7, figura 8 respectivamente:
- La adaptación de la librería JPCAP fue exitosa para dar solución al problema.
- Las herramientas utilizadas con el fin de vulnerar el sistema, fueron de utilidad, ya que por medio de estas se pudieron determinar el tipo de ataques.
- Se pudieron capturar tramas con las características necesarias para la detección de intrusos.

Figura 7. Datos de ataque con reglas.

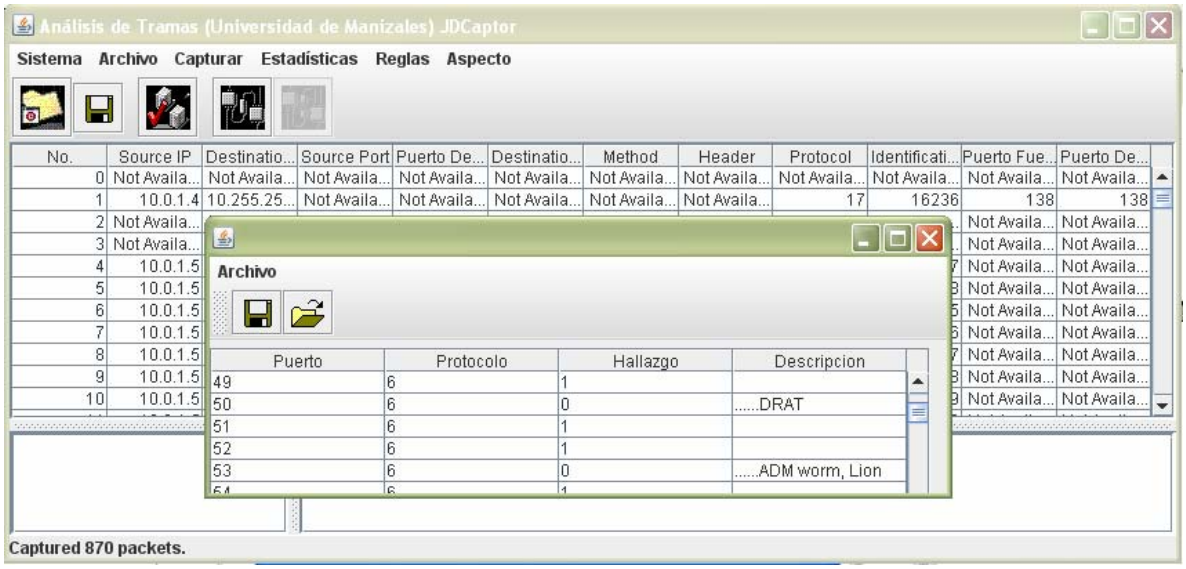
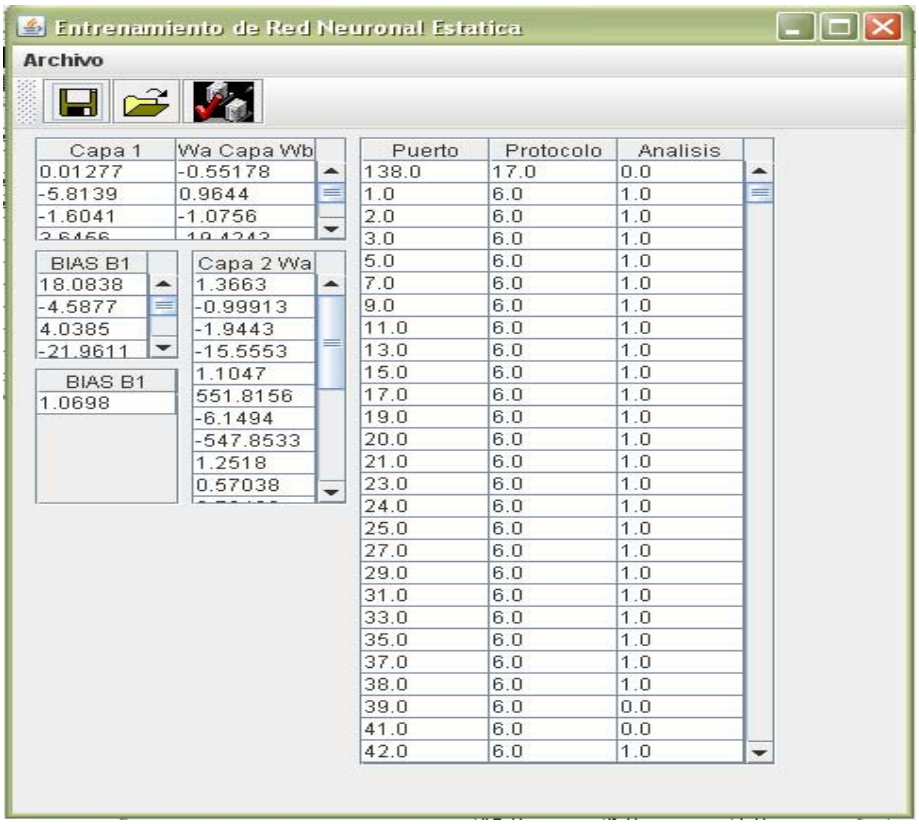


Figura 8. Datos de ataques, (entrenamiento red neuronal).



La información capturada se puede analizar mediante dos técnicas; A través de reglas o mediante una RNA previamente entrenada utilizando MATLAB®. Una vez realizado el análisis, es posible ver las conclusiones del mismo, tal como se muestra en la Figura 6. El análisis puede resumirse en: Datos identificados con un 1 si es inofensivo y con un 0 si es ataque (con su respectiva descripción).

Además de todo lo anterior en el IDS desarrollado se puede observar las estadísticas de los paquetes recibidos con gran variedad de gráficas, además de cambiar su aspecto y muchas características más que junto con las mencionadas anteriormente se podrán ver en el manual de usuario.

- En este proceso de entrenamiento de la RNA realizado en MATLAB® se tuvieron diferentes parámetros de entrenamiento, entre ellos diez neuronas y mil épocas, en la cual no se alcanza el error mínimo, tal como se observa en la

Figura 9; hasta que se encontró una arquitectura y un porcentaje de error tal que el entrenamiento convergiera. En este caso se tiene un error del 1%, una estructura de dos capas, con veinte neuronas en la primera capa y una neurona de salida en la segunda capa. El entrenamiento se realizó durante dos mil setecientas épocas, tal como se observa en la figura 10.

Figura 9. Entrenamiento con diez neuronas.

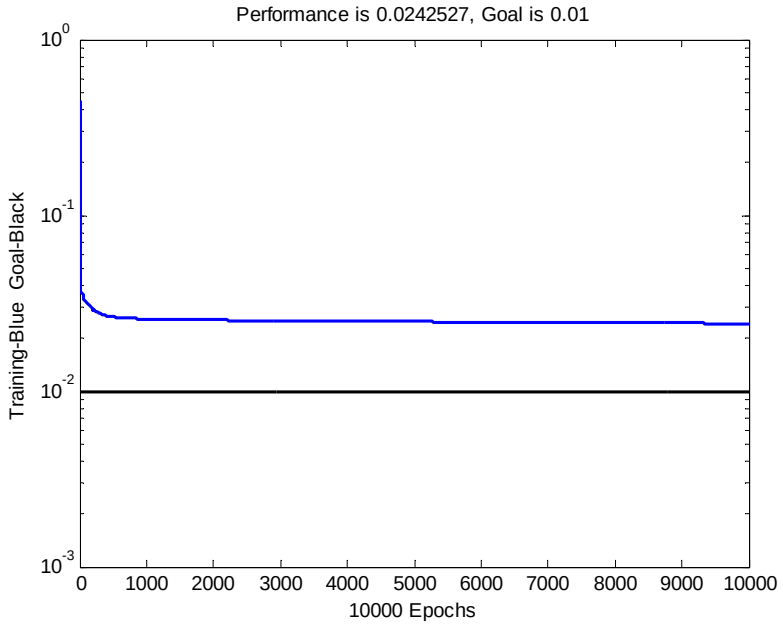
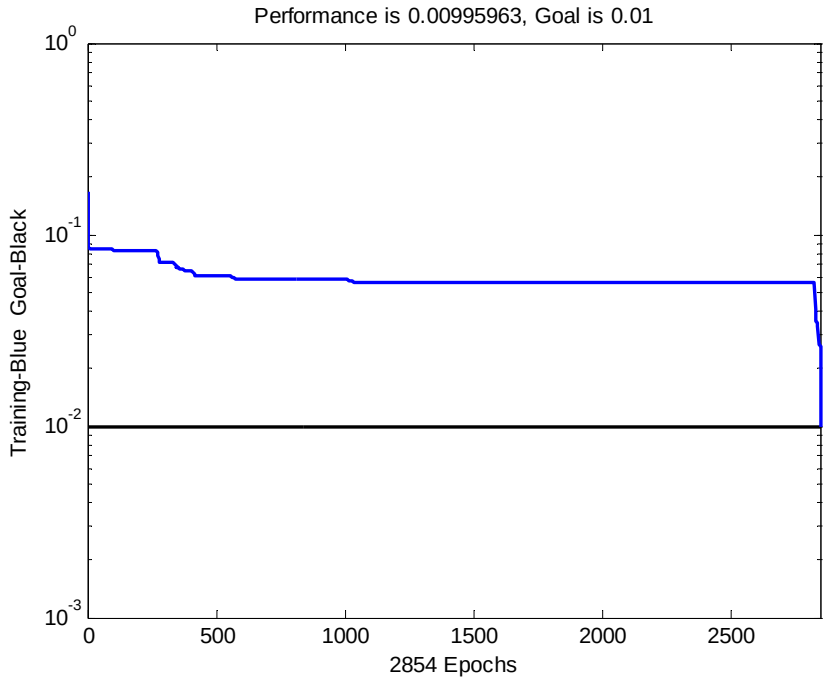


Figura 10. Entrenamiento MATLAB (10 neuronas)



- Se realizó una comparación entre la detección utilizando reglas y la realizada mediante la red neuronal de las tramas capturadas, exportando esta información desde la aplicación a una hoja electrónica utilizando Microsoft Office Excel. El resultado final fue el siguiente:

Se detectaron de 856 tramas los siguientes paquetes con su respectivo porcentaje

Detección por Reglas:

De 266 paquetes capturados determinados como ofensivos se detectaron 266 paquetes malos, obteniendo un porcentaje y un trabajo de funcionamiento del 100 %, del cual 590 paquetes buenos fueron capturados y establecidos como buenos. Para el IDS que se implemento estos resultados sirven para validar la detección alojada por el entrenamiento de la red neuronal.

# de paquetes	Tipo de paquetes	Porcentaje
266 – 266	Malos	100%
590 – 590	Buenos	100 %
		100 %

Detección mediante la Red Neuronal

# de paquetes	Tipo de paquetes	Porcentaje
266 – 230	Malos	86.46 %
590 – 570	Buenos	96.61 %
		91.535 %

Capturado 266 paquetes el sistema detecto 230 en los cuales la efectividad en la detección en el peor caso es de 91.535%, este porcentaje indica la viabilidad de la técnica utilizada (Redes Neuronales), teniendo en cuenta que el aprendizaje se da con la experiencia (entrenamiento) y que la detección podría ser mas puntual.

7. CONCLUSIONES

- ✓ El uso de las reglas fue útil al momento de establecer los datos de entrenamiento de la red neuronal debido a que las reglas son definidas por el usuario y son un 100% verídicas. De esta manera la red neuronal puede ser comparada y reconstruida para un mejor funcionamiento en su detección de paquetes de datos.

- ✓ Se utilizó la librería JPCAP, ya que el API de Java cuenta con esta herramienta que provee de varios elementos para la captura de tramas de bajo y alto nivel de una red de datos. Su programación y su codificación se hace a la necesidad del usuario y al operar sobre Java garantiza ser multiplataforma que ayudó a cumplir con los objetivos descritos inicialmente.

- ✓ Las herramientas utilizadas para vulnerar la red de datos se mostraron muy importantes durante la investigación.

- ✓ La alternativa estudiada como opción para detectar intrusos en una red de datos utilizando redes neuronales, es suficiente para conocer en el estado del medio en que nos desempeñamos, y así tomar decisiones sobre la base de la información brindada por los medios obtenidos, en este caso el IDS ya mencionado.

8. RECOMENDACIONES

En términos generales, el presente proyecto se constituye en una primera etapa de un proyecto de mayor envergadura y alcance, en el cual a la Universidad de Manizales a través de las investigaciones llevadas a cabo al interior de la Facultad de Ingeniería, desarrolle sus propias herramientas de seguridad informática y se convierta en un líder a nivel regional, nacional y latinoamericano en elaboración de software.

Específicamente, se recomienda utilizar en un siguiente estadio de este proyecto la realización de más vulnerabilidades los cuales dependan de más patrones que no solo sean puerto y protocolo, generalizando las debilidades que un sistema de información pueda tener, ya que la aplicación da la posibilidad como herramienta de seguir con este estudio.

Por otra parte a nivel de Hardware los requerimientos mínimos y que de hecho hoy en día se consiguen en el mercado son:

- ✓ Procesador Pentium IV en adelante.
- ✓ AMD de 2200 MHZ en adelante
- ✓ Memoria RAM de 128 en adelante
- ✓ Tarjeta inalámbrica o tarjeta de red para la captura de datos
- ✓ Cable de red RJ45 directo o Cruzado dependiendo la configuración de la red de datos.

BIBLIOGRAFÍA

BLOG DE SOFTWARE GENBETA. Back Orifice: especial software control remoto. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en: http://www.genbeta.com/2006/06/13-back-orifice-especial-software-control-remoto

IT PRO MEXICO. Métodos de Escaneo de Puertos. [En Línea]. Fecha de Consulta: 03-08-2007.Inicio. Disponible en: http://www.itpromexico.com.mx/paginas/articulos/scanportmet.htm

JPCAP. A Java library for capturing and sending network packets. [En Línea]. Fecha de Consulta: 03-08-2007. Home. Disponible en: http://netresearch.ics.uci.edu/kfujii/jpcap/doc/

MEMBERS.TRIPOD.COM. Redes Neuronales, Introducción y conceptos Básicos, Primera y segunda parte, [En Línea]: Fecha de consulta agosto de 2007. Disponible en: http://members.tripod.com/jesus_alfonso_lopez/RnaIntro.html

NETBEANS.ORG. Bienvenido A Netbeans. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en: http://www.netbeans.org/index_es.html.

OBJECT MANAGEMENT GROUP. Introduction to OMG's Unified Modeling Language™ (UML®). Fecha de Consulta: 03-08-2007. Inicio > Introduction to UML. Disponible en: http://www.omg.org/gettingstarted/what_is_uml.htm

PALACIOS, Francisco. Herramientas en GNU/Linux para estudiantes universitarios. Tipos de Redes Neuronales. [En Línea]. 2003. Disponible en: http://softwarelibre.unsa.edu.ar/docs/descarga/2003/curso/htmls/redes_neuronales/index.html

PIOJOSOFT - EL SITIO DE DESCARGAS. Megapin 4.3. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en http://www.piojosoft.com/index.php?empezar=0&ide=200

SEGURIDAD EN LA RED.Virus, gusanos, troyanos y backdoors. [En Línea]. Fecha de Consulta: 03-08-2007. Inicio. Disponible en: http://www.seguridadenlared.org/es/index10esp.html

UNIVERSIDAD DE ANTIOQUIA. Historia de las Redes Neuronales. [En Línea]. Fecha de Consulta: 30-01-2007. Disponible en: http://ingenieria.udea.edu.co/investigacion/mecatronica/mectronics/redes.htm

WIDROW LIRA, Cristian. REDES NEURONALES ARTIFICIALES: POTENCIAL DESARROLLO MILITAR. [En Línea]. Disponible en: http://www.revistamarina.cl/revistas/1998/5/widow.pdf

ANEXOS

ANEXO A. Modelo de Requisitos

Los paquetes enmarcan para el proyecto el contexto de la aplicación en general, tanto en las necesidades generadas, como en su estructura. Uno de los factores importantes es la Seguridad, en éste se encuentran todas los requerimientos para el acceso al medio; todo con el fin de obtener la información del tráfico, los Intrusos son una abstracción de todo lo referente al acceso y detección de intrusos en una red determinada y el procesamiento de la Información es un factor que incide tanto en la captura como en análisis requerido sobre la información.

Diagrama de paquetes (Requisitos)

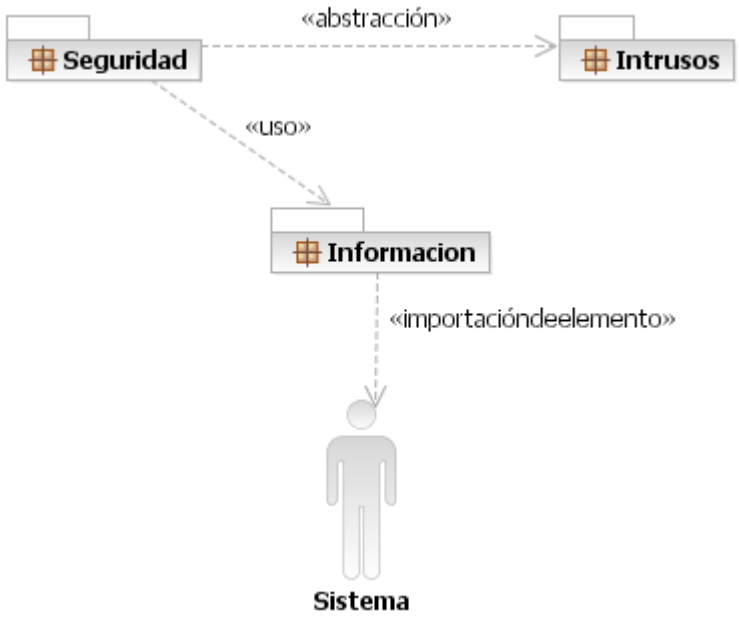


Diagrama de Casos de Uso (Paquete Seguridad)

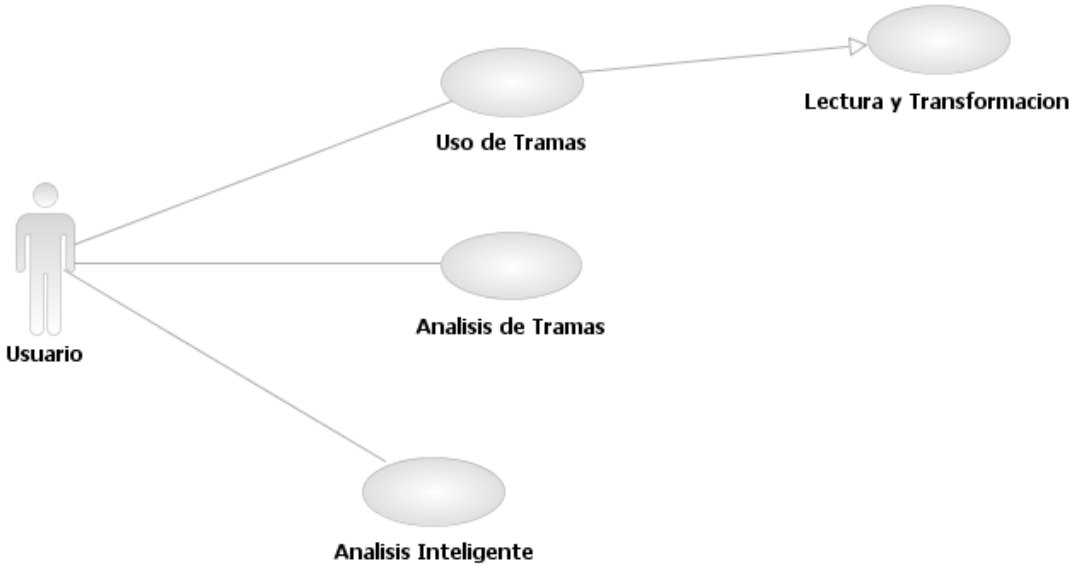


Diagrama de Casos de Uso (Paquete Intrusos)

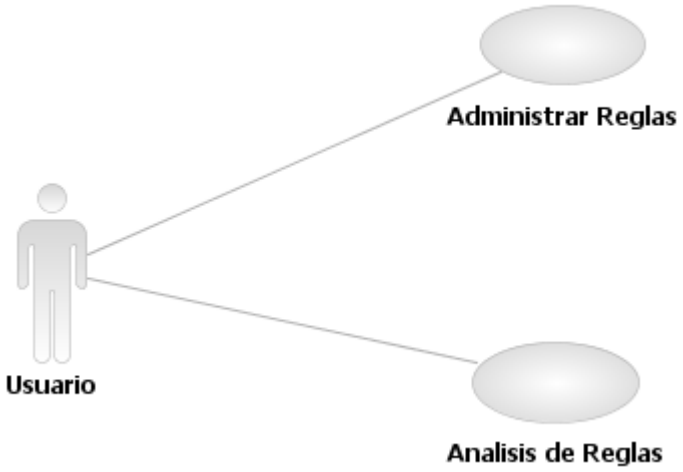


Diagrama de Casos de Uso (Paquete Información)

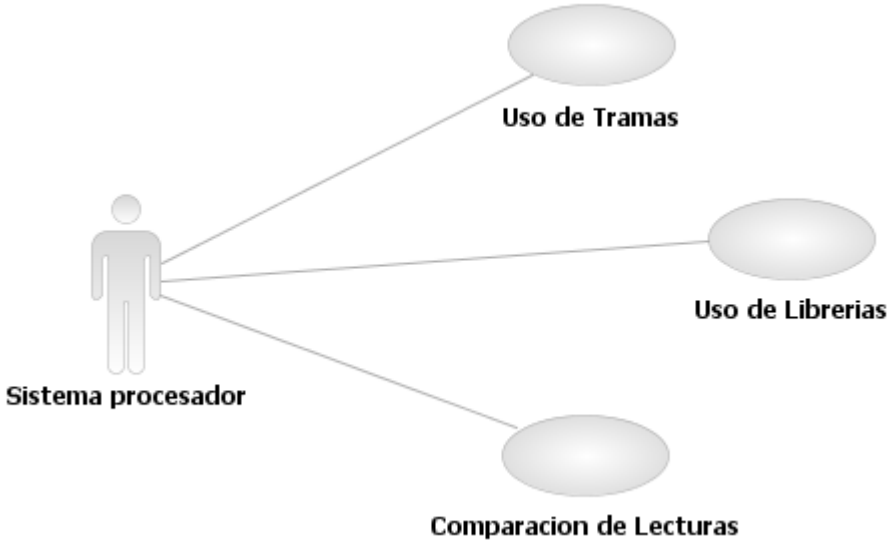


Diagrama de actividad (lectura de tramas)

Uno de los requisitos más relevantes es la lectura de tramas en la red; todo se logra mediante el acceso al medio, pasando por un análisis previo para la identificación y detección de intrusos, también es necesario obtener toda la información contextual. Es posible observar en el siguiente diagrama se ilustra los procesos y procedimientos básicos y lineales para lograr dicha función.

Diagrama de Actividad (Lectura de Tramas)

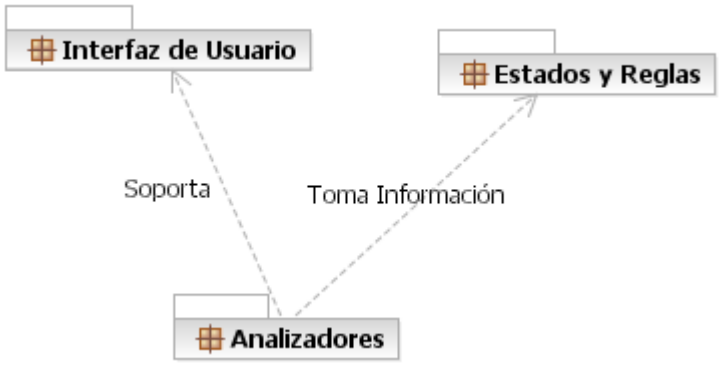
ANEXO B. Modelo de Análisis

Casos de Uso (Nivel Requisitos)

El análisis de la aplicación está basado en el modelo de requisitos, todo con el fin de crear las diferentes interfaces, controles y entidades, tal y como lo establece la ingeniería de modelos propuesta por el proceso unificado (UP) bajo la notación UML 2.0.

Se requiere la adaptación de una librería de tal forma que permita estructurar las características de la aplicación en modelos vista controlador, es por esto que se deben construir paquetes, tal y como se puede observar en el diagrama descrito.

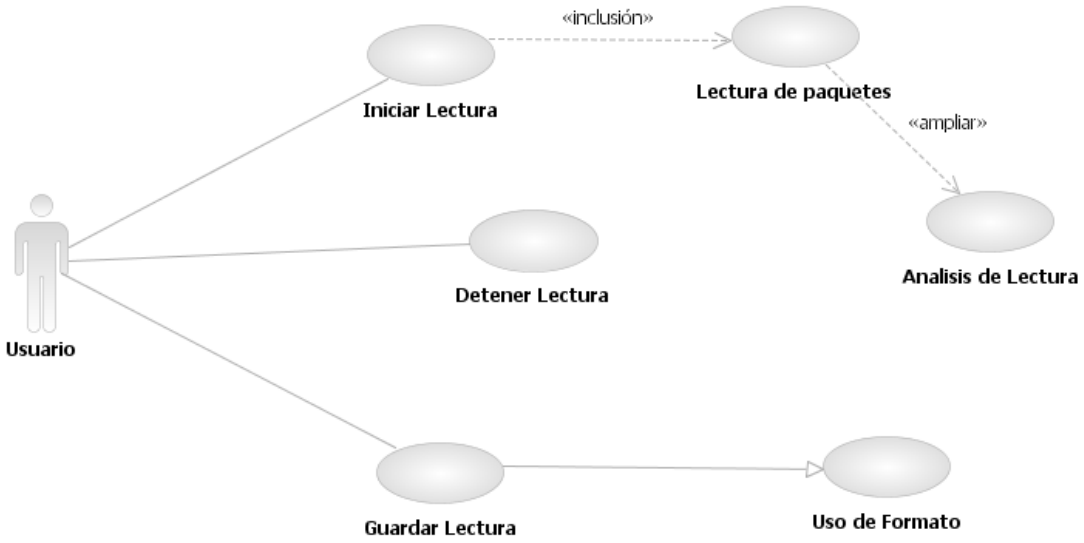
Vista de modelos (Paquete Análisis)



Se define como lineamiento principal los casos de uso que abarquen todo lo indicado, es por esto que se tiene como los siguientes aspectos:

La lectura de tramas es un factor principalmente en el momento del análisis de los requerimientos, es por esto que se estructuran los casos de uso para abarcar dicha temática, por tal motivo se indica en un caso de uso independiente, para luego ser tomada en el modelo de diseño como tal, así como se observa en el diagrama:

Casos de Uso (Analisis de Lectura)



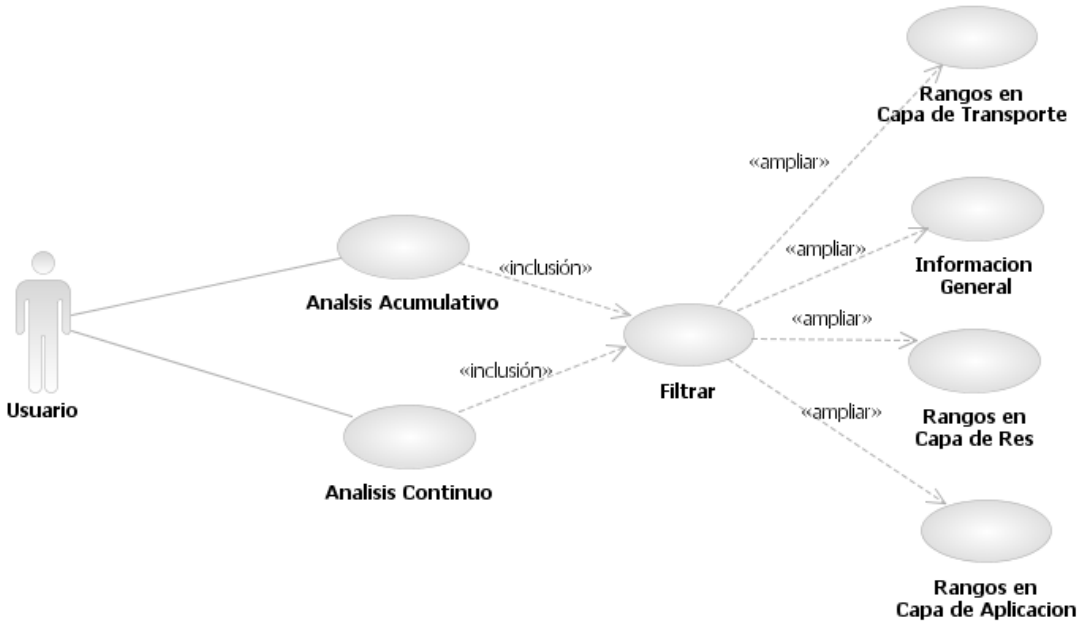
Casos de Uso para Análisis y Procesamiento

Los casos de uso en el modelo de análisis no ayuda a comprender, estructurar y definir cómo es realmente la necesidad plasmada en el modelo de requisitos; en éste caso de encuentran dos áreas con un procesamiento y tratamiento diferente, pero contextos diferentes, es entonces el caso de dos tipos de análisis, el acumulativo y el continuo.

Análisis acumulativo: Para éste análisis es apropiado tomar la lectura efectuada en el medio (como lo indican los casos de uso anteriores) para determinar cual y que tipo de información fue tomada de la red para el respectivo análisis.

Análisis continuo: Es un tipo de tratamiento que se diferencia del anterior mostrando la información completa, ya sean rangos de transporte, información general, estadísticas y entre otros.

Casos de Uso para Análisis y Procesamiento

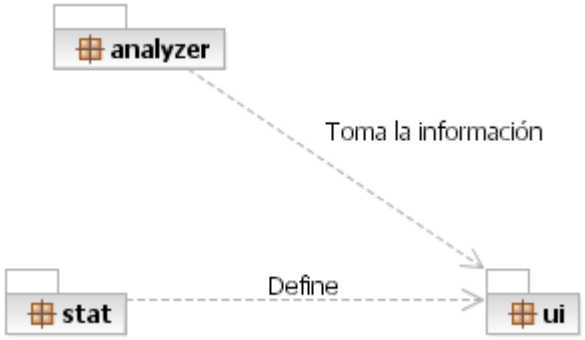


Diagramas de Clase

Diagramas de clase a nivel de núcleo

Los diagramas a nivel núcleo son las que se usan para lograr los objetivos trazados en el proyecto, para implementar con mayor efectividad de reutilización y mejorar las prestaciones de la librería a adaptar es necesario contextualizar las clases en paquetes, observe el diagrama abajo descrito.

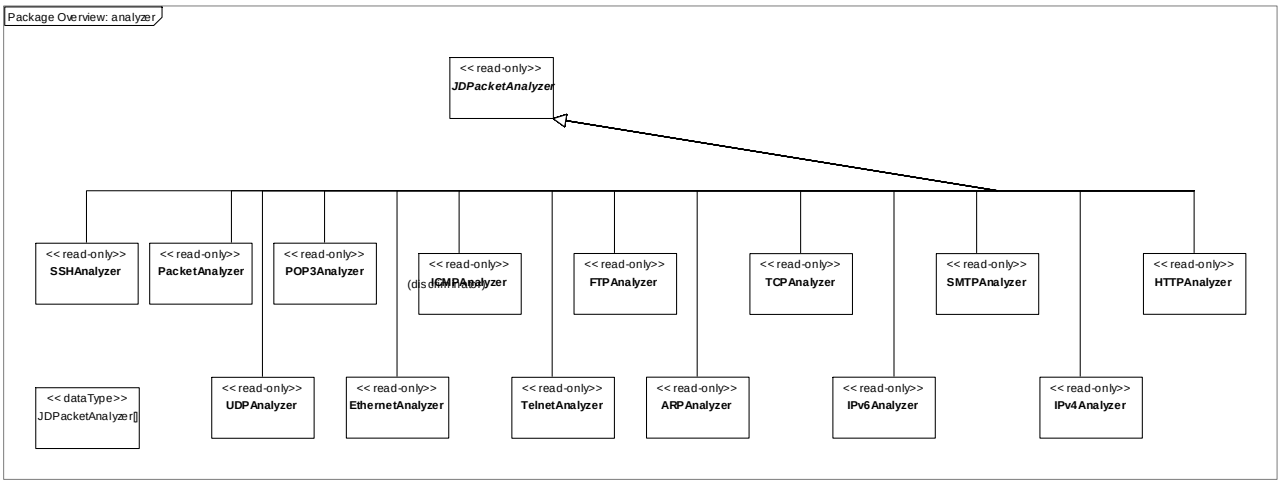
Diagramas de clase a nivel de núcleo (Paquetes)



Paquete analyzer

Es donde se especifica cada uno de los paquetes en forma tal que se evidencien las clases que estos contiene:

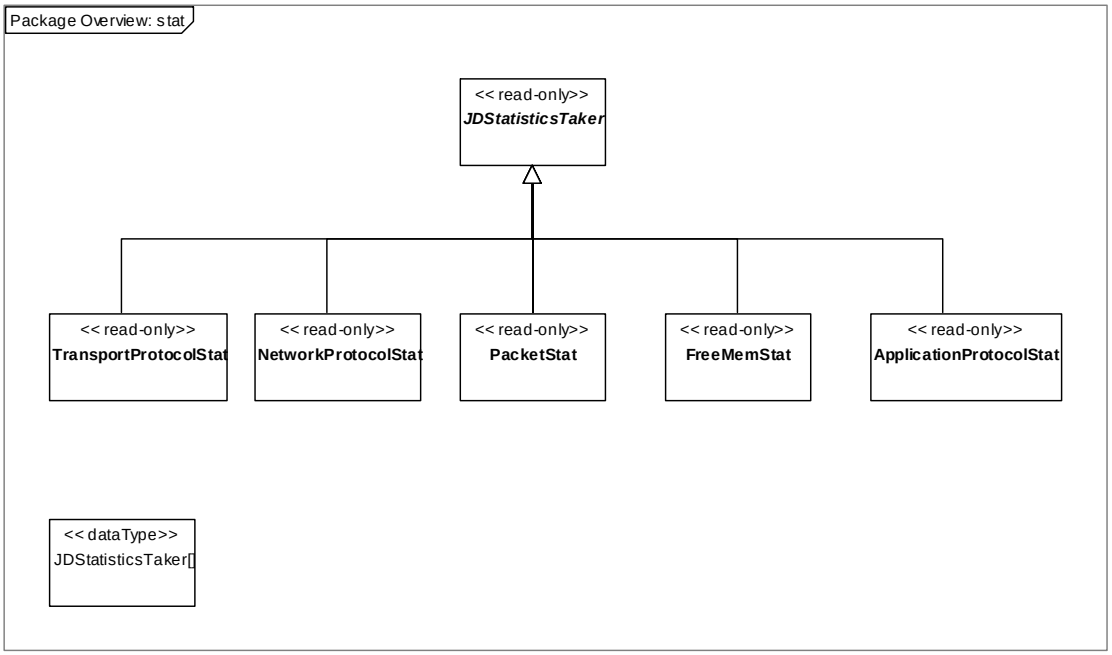
Dicha información será estipulada con mejor detalle en el modelo de diseño, donde se construye todo lo referente al uso de dichas clases; cabe anotar que se toman todas las características de una librería adecuada para lograr los objetivos.



Paquete Stat

En el paquete stat encontraremos todo lo referente a la librería JPCAP, es decir, todos los estados estipulados para las capas de transporte, aplicación, y de red.
Como se indica en el diagrama, se logra mediante una herencia estipulada por UML 2.0 como una generalización, esto se obtiene usando la herramienta POSEIDON, la cual nos permite hacer un mejor análisis de la estructura de la librería en sí, su principal objetivo es no cometer errores en el momento del análisis, debido a que esto influye directamente en el modelo de diseño.

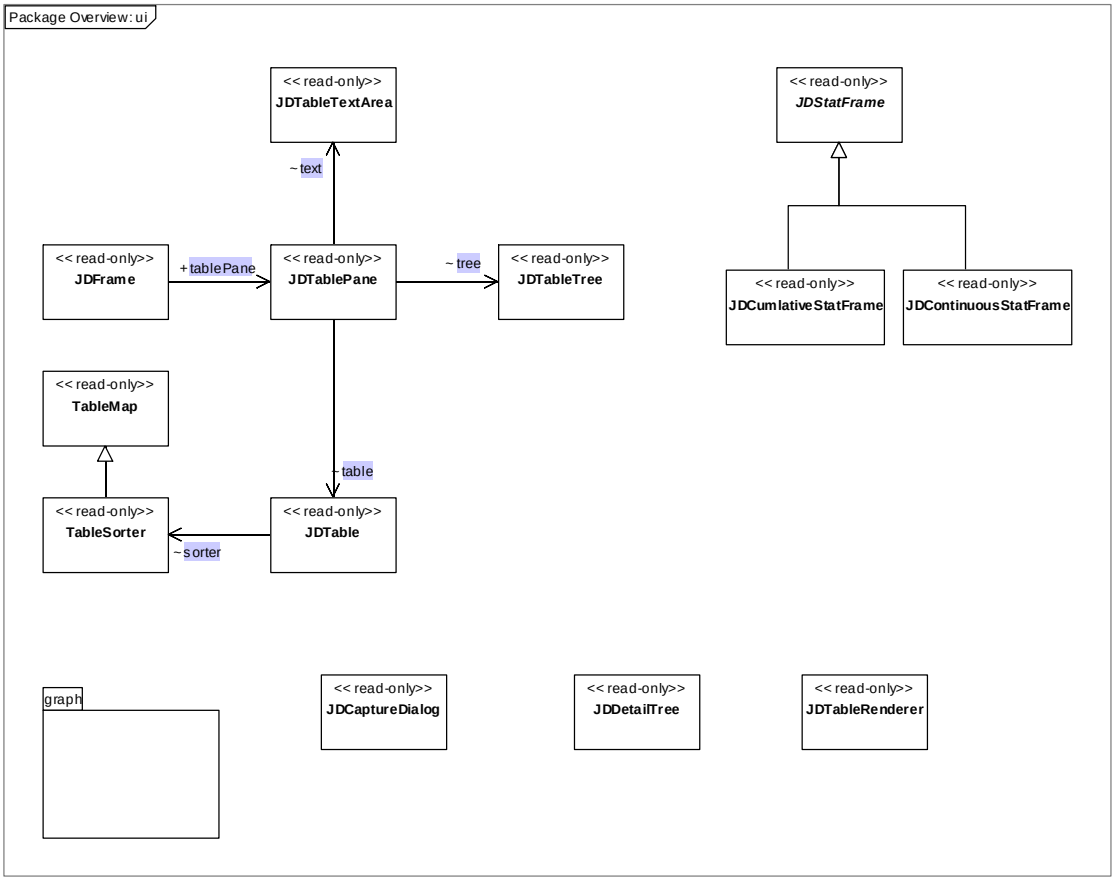
Diagrama Paquete stat



Paquete ui

Todo lo referente a interfaces debe ser concebido en forma independiente, para acercar el modelo a vista controlador, y lograr mejorar sus prestaciones en cuando a efectividad y calidad de software.

Diagrama Paquete ui



Diagramas de clase (análisis de la aplicación)

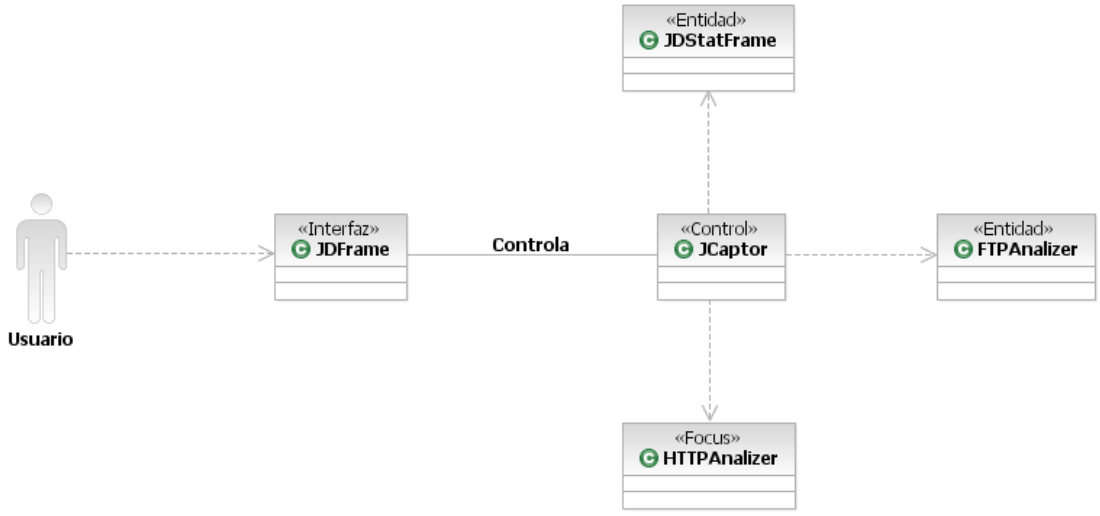
Para comenzar es necesario tener en cuenta los lineamientos que hay en el proceso unificado de desarrollo de software, el cual se adapta a las herramientas de desarrollo seleccionadas para el proyecto, entre las que se tiene a NETBEANS 6.0, IBM RATIONAL MODELER y POSEIDON.

El siguiente diagrama enmarca lo que es la ventana principal, teniendo en cuenta tres tipos de clases para éste diagrama, interfaz, entidad y control; éstos definen las clases que se usan como aporte primordial en el modelo de diseño.

Inicio de la aplicación

Para el inicio de la aplicación es apropiado tener en cuenta las clases que derivarán el arranque de la aplicación, así de la siguiente manera:

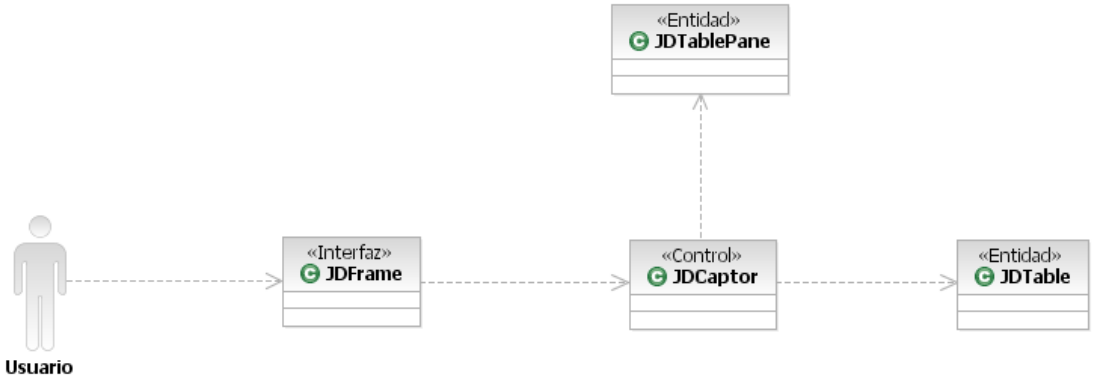
Diagrama de Clases Inicio de la aplicación



Lectura de Tramas

La lectura de las tramas se hace por medio de la interfaz principal, se puede llegar por medio del menú o por medio de otro elemento, el cual deberá ser definido en el modelo de diseño, abajo se observa un diagrama que muestra las clases que afectan dicho procedimiento.

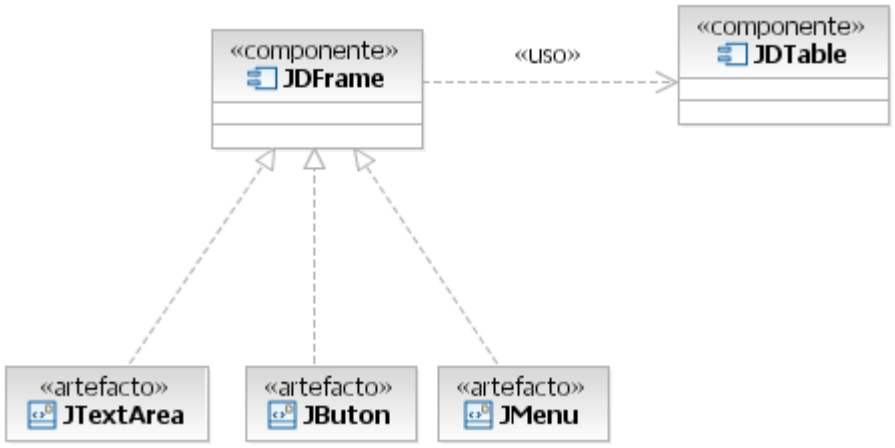
Diagrama de clases Lectura de Tramas



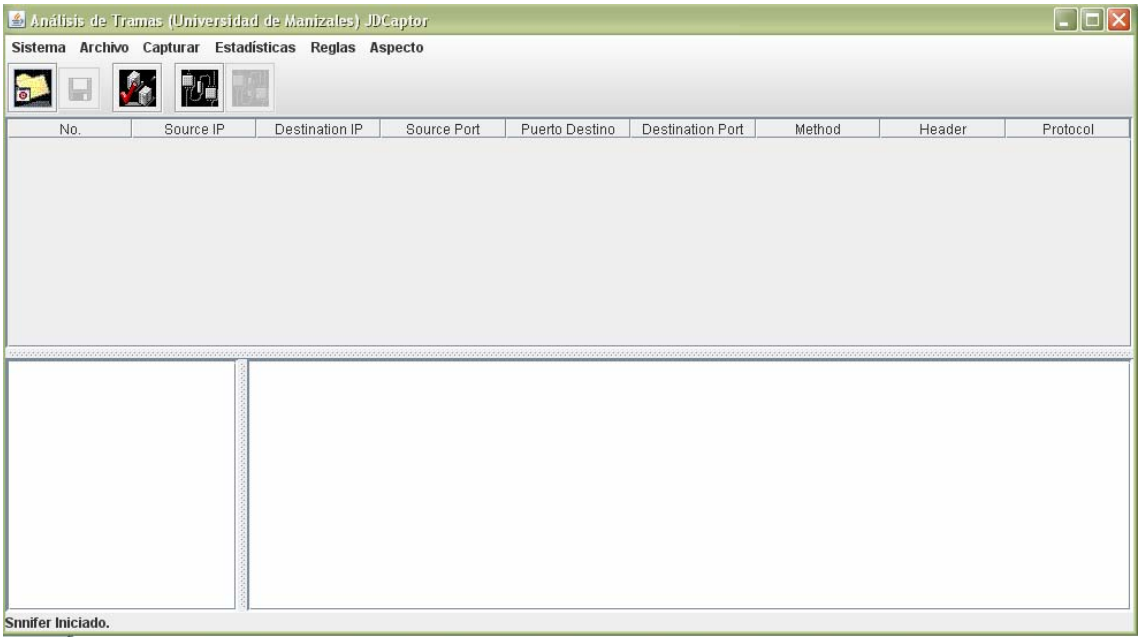
ANEXO C. Modelo de Diseño

Diseño de la aplicación

Diagrama de componentes Ventana principal

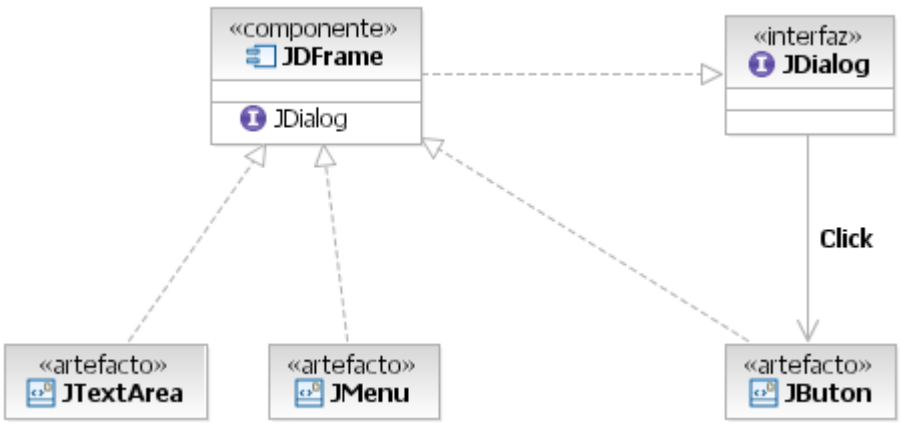


Diseño Ventana Principal

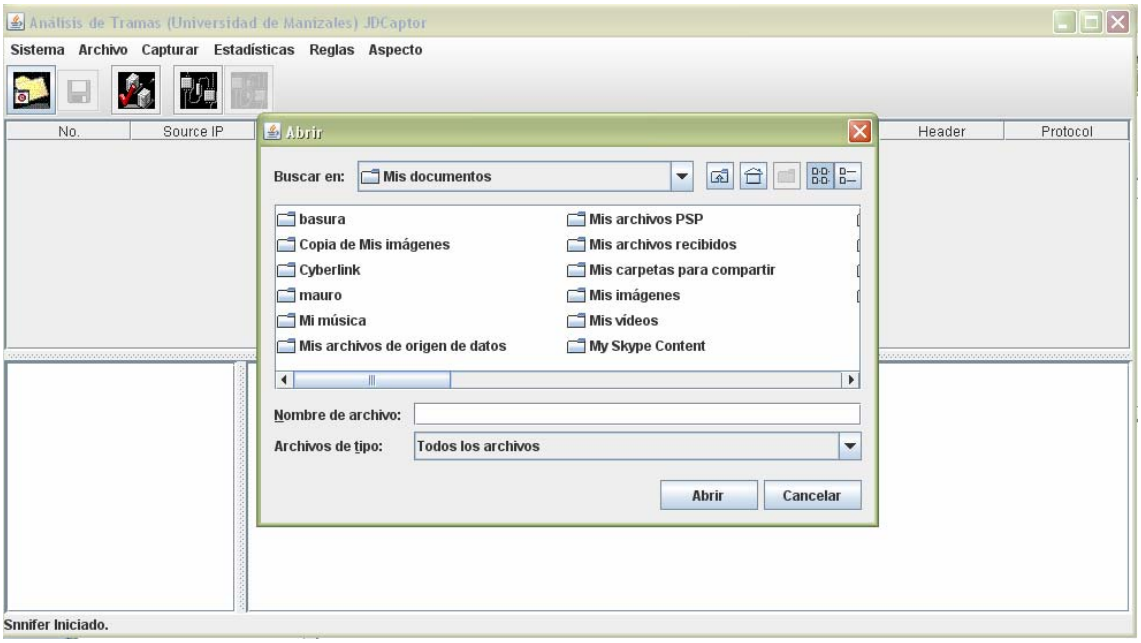


Abrir un archivo

Diagrama de componentes (Abrir un archivo)

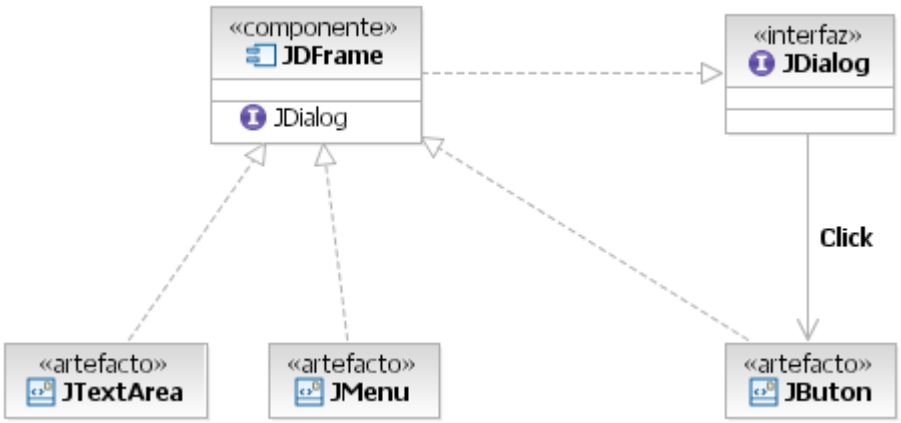


Diseño Abrir un archivo

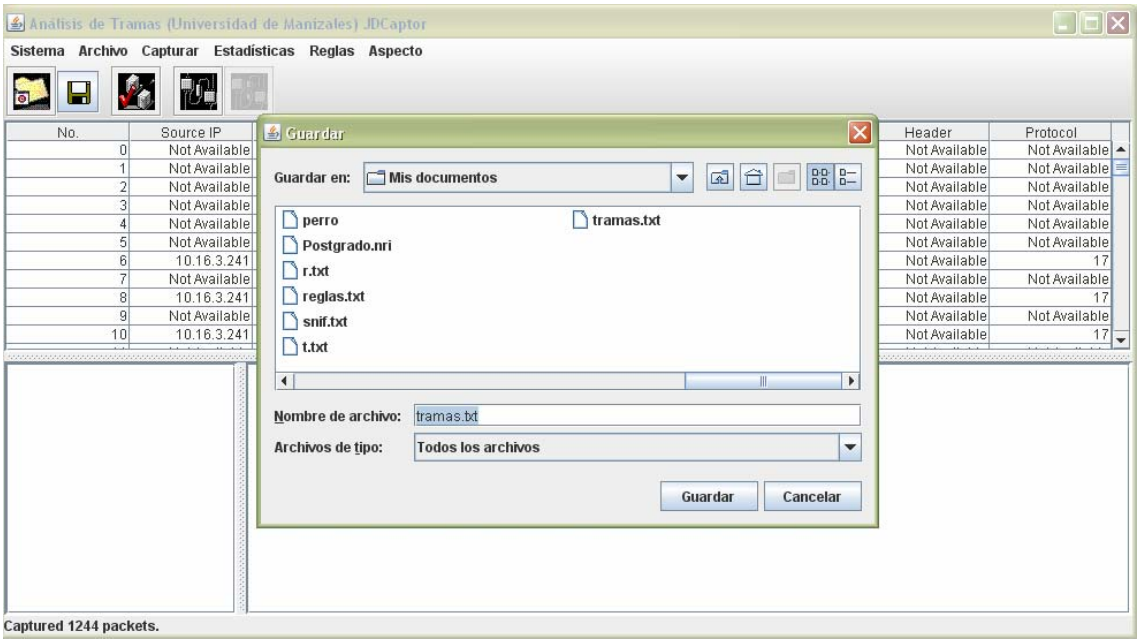


Guardar un archivo

Diagrama de componentes Guardar un archivo

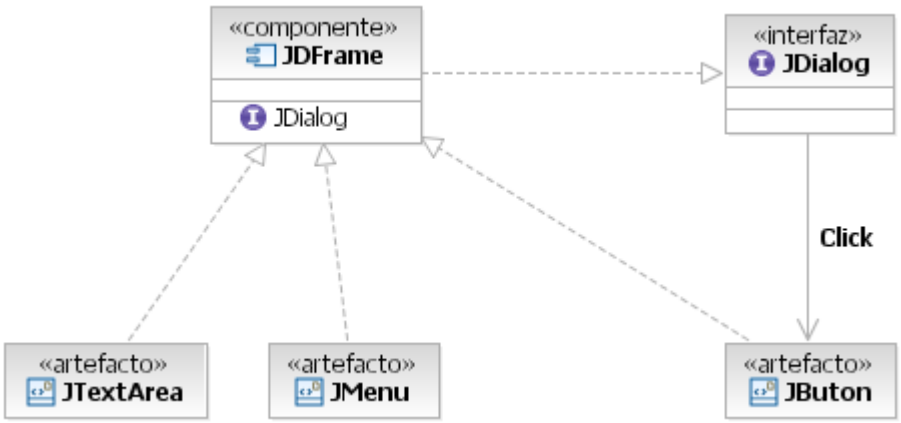


Diseño Guardar un archivo

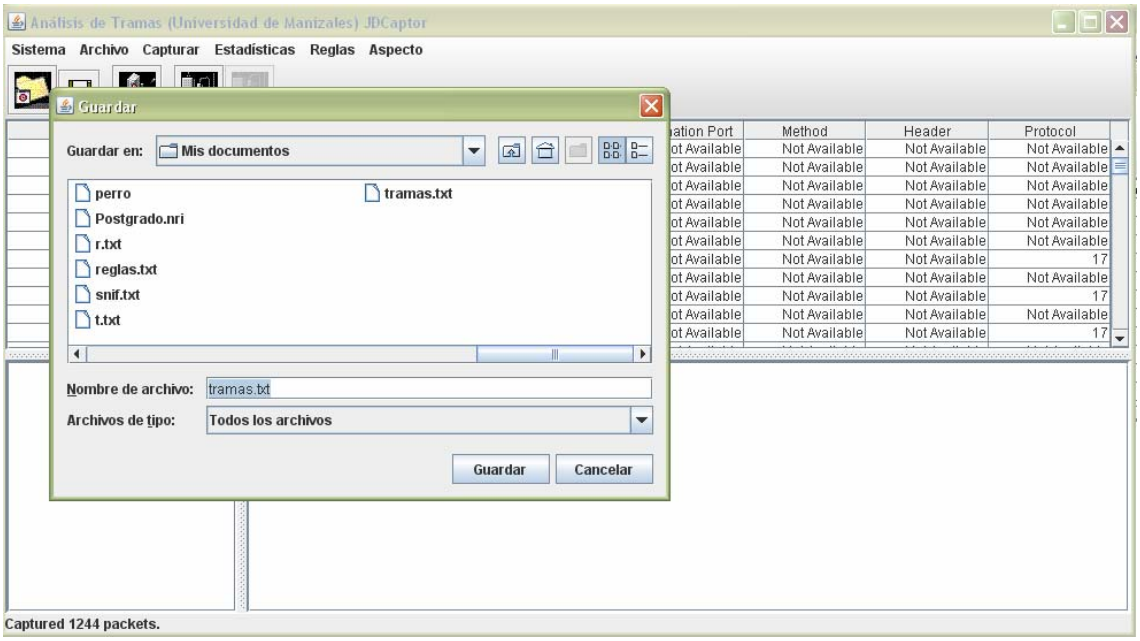


Exportar para análisis

Diagrama de componentes Exportar para análisis

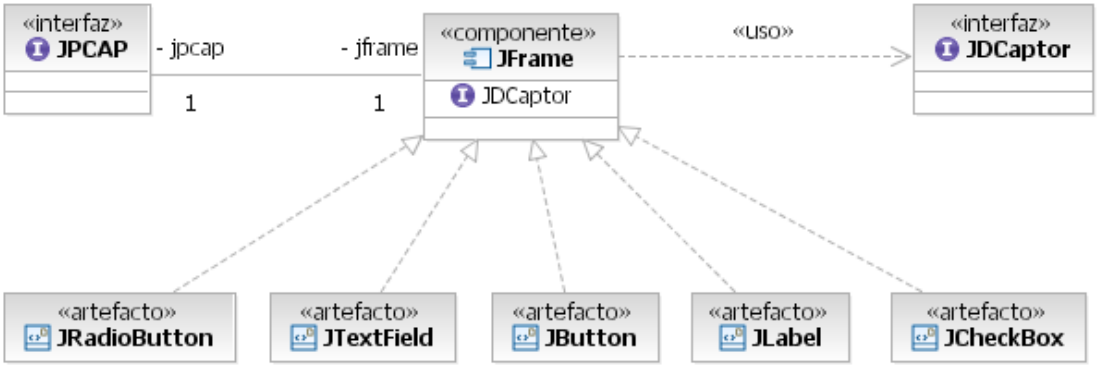


Diseño Exportar para análisis

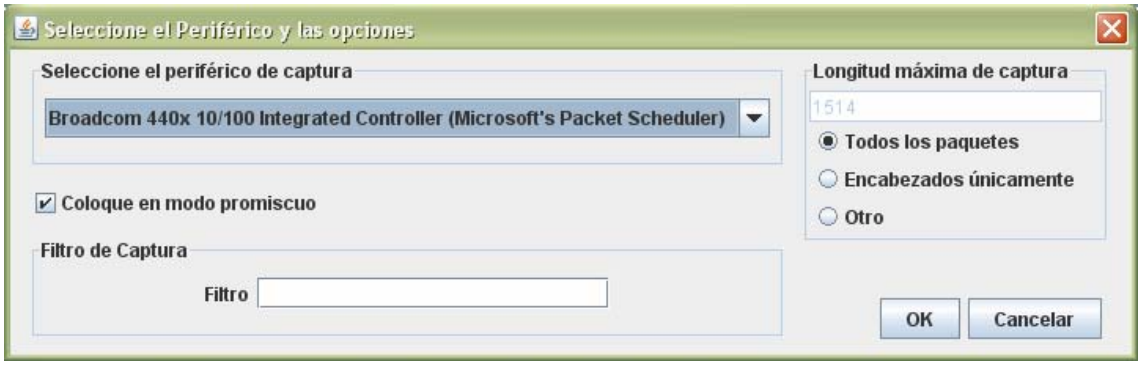


Lectura de Tramas

Diagrama de componentes Lectura de Tramas



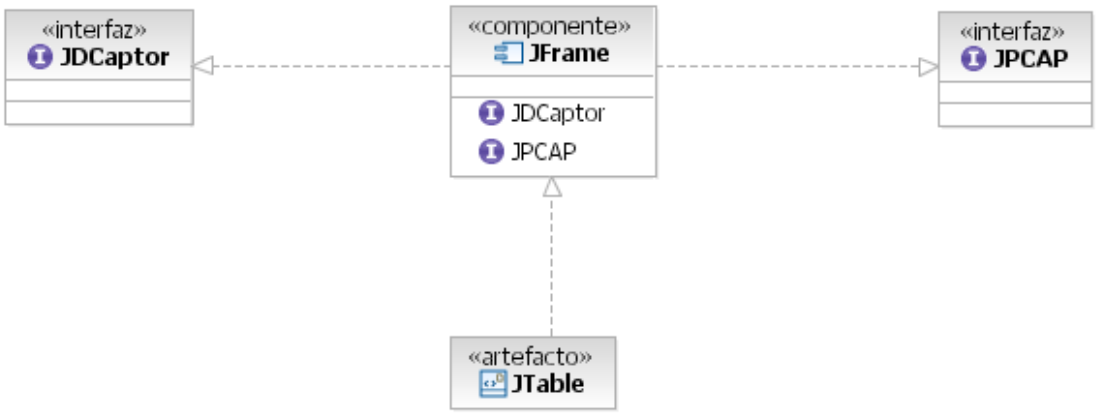
Diseño Lectura de Tramas



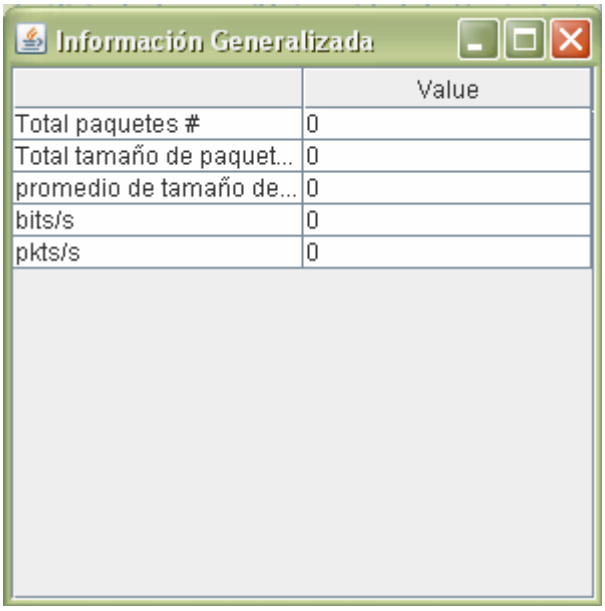
MÓDULO DE ESTADÍSTICAS

Información general

Diagrama de componentes (Información general)

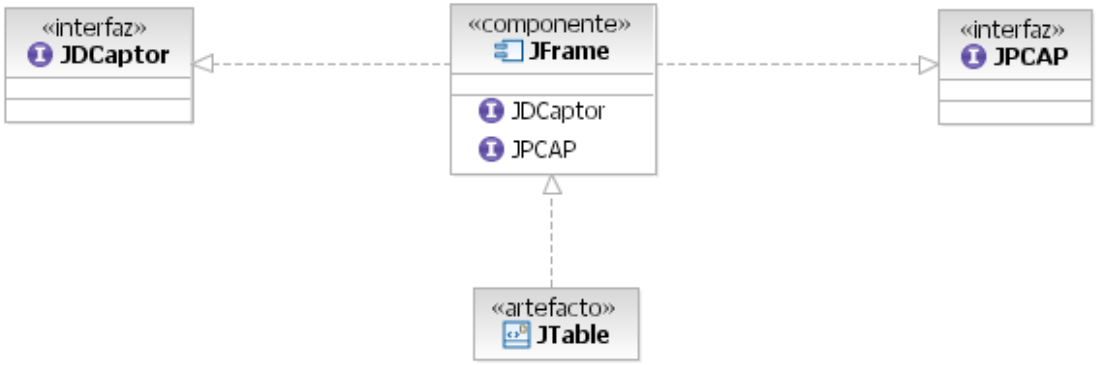


Diseño Información general

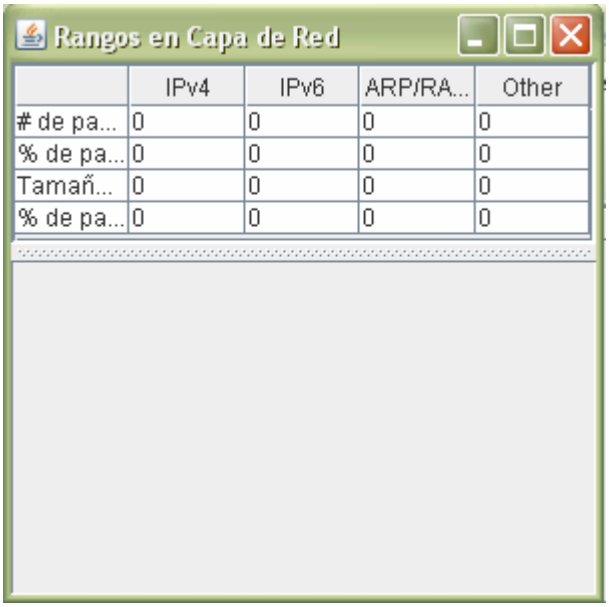


Rangos en capa de Red

Diagrama de componentes Rangos en capa de Red

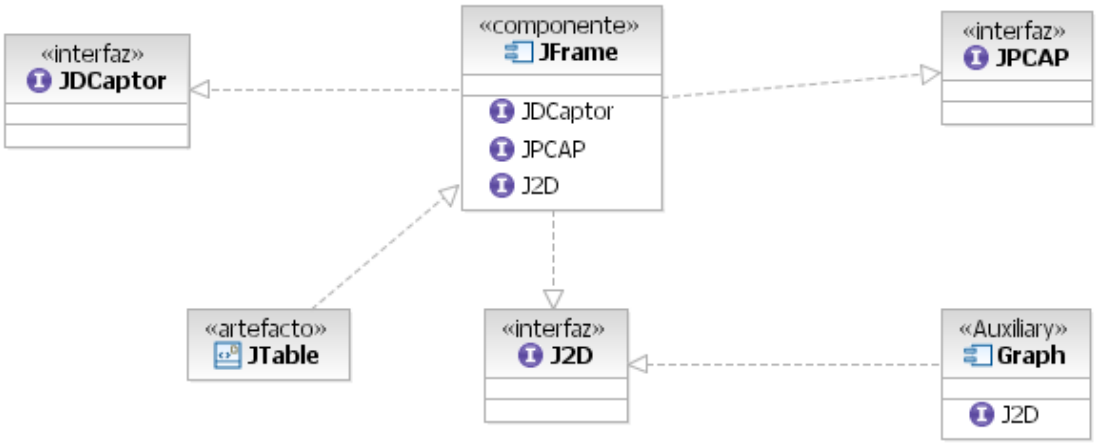


Diseño Rangos en capa de Red

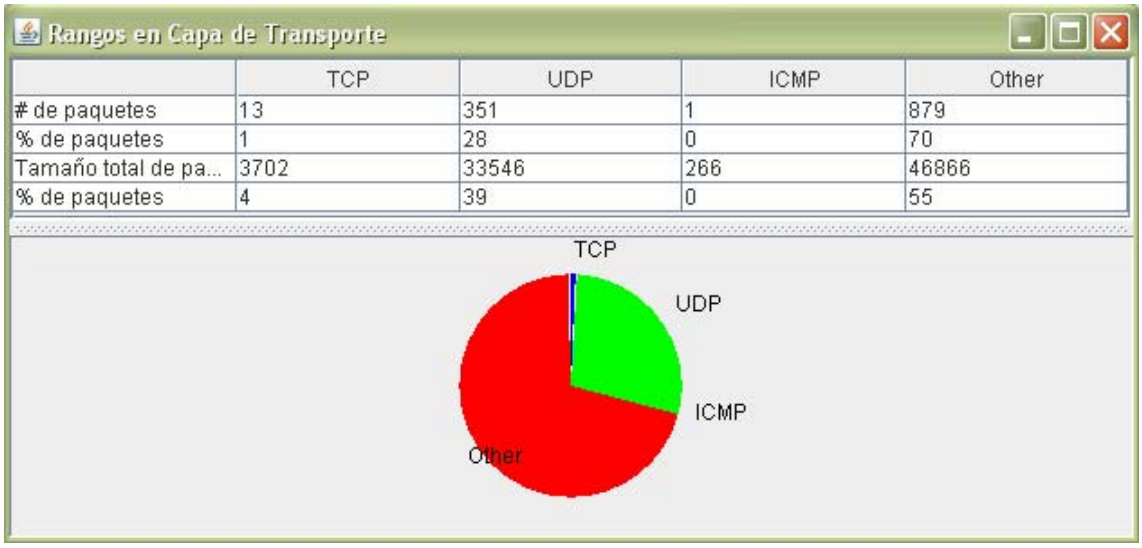


Rangos en capa de transporte

Diagrama de componentes Rangos en capa de transporte

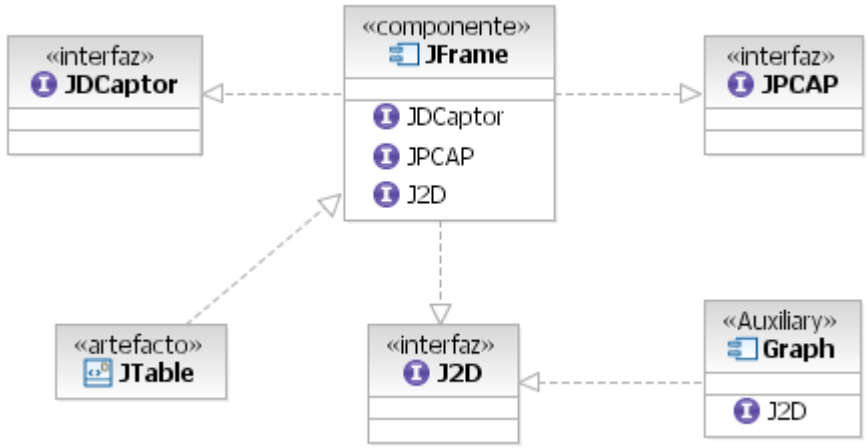


Rangos en capa de transporte

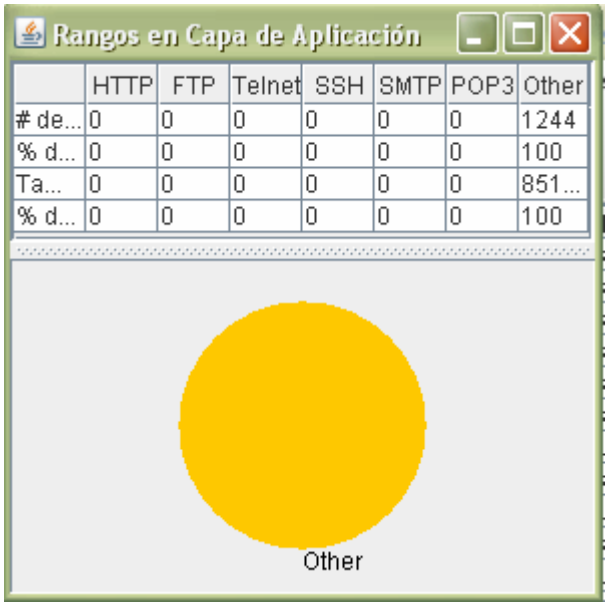


Rangos en capa de aplicación

Diagrama de componentes Rangos en capa de aplicación

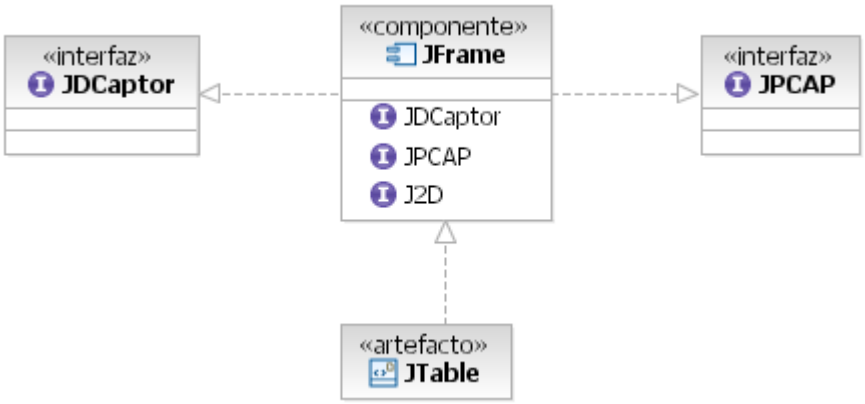


Diseño Diagrama de componentes Rangos en capa de aplicación

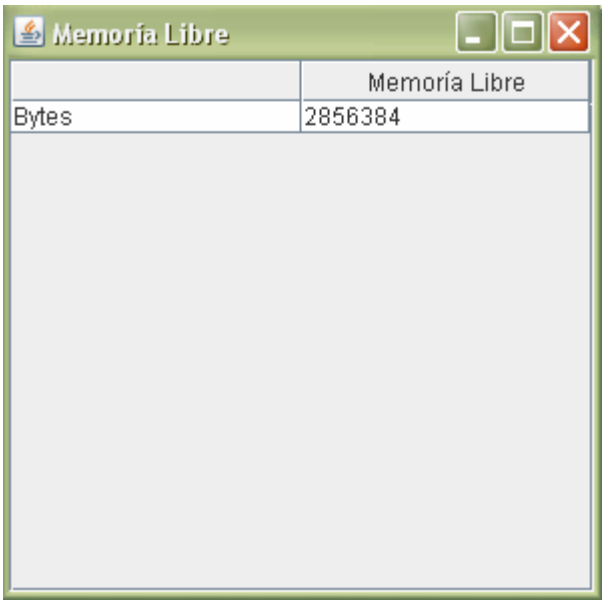


Memoria libre

Diagrama de componentes Memoria libre



Diseño Memoria libre



Análisis Estadístico Continuo

Diagrama de componentes Análisis Estadístico Continuo

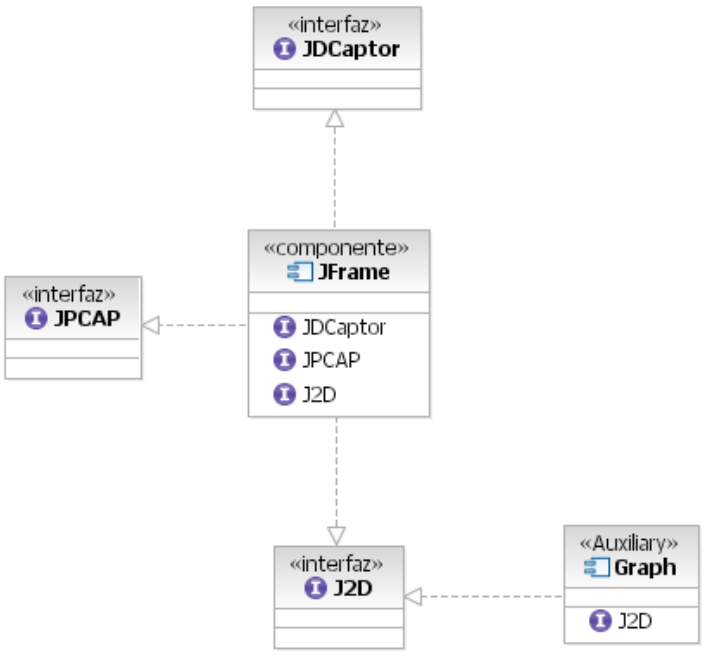
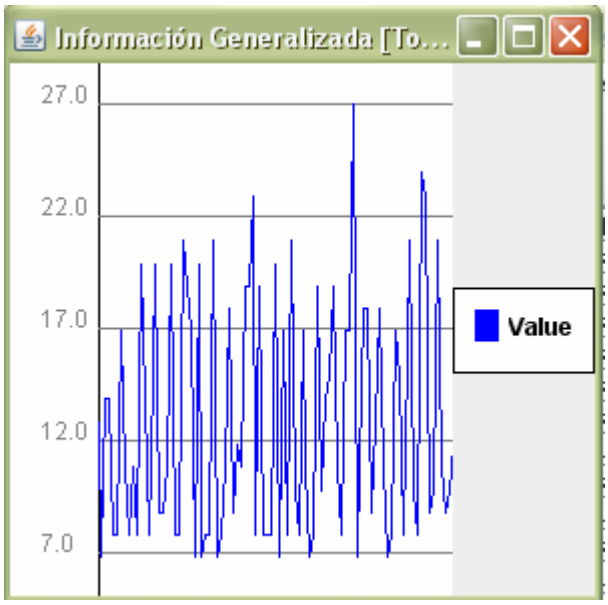
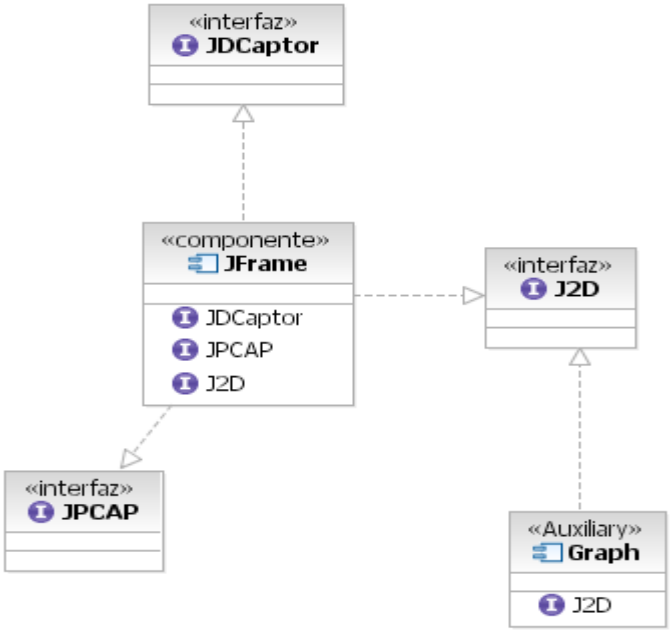


Diagrama de componentes Análisis Estadístico continuo

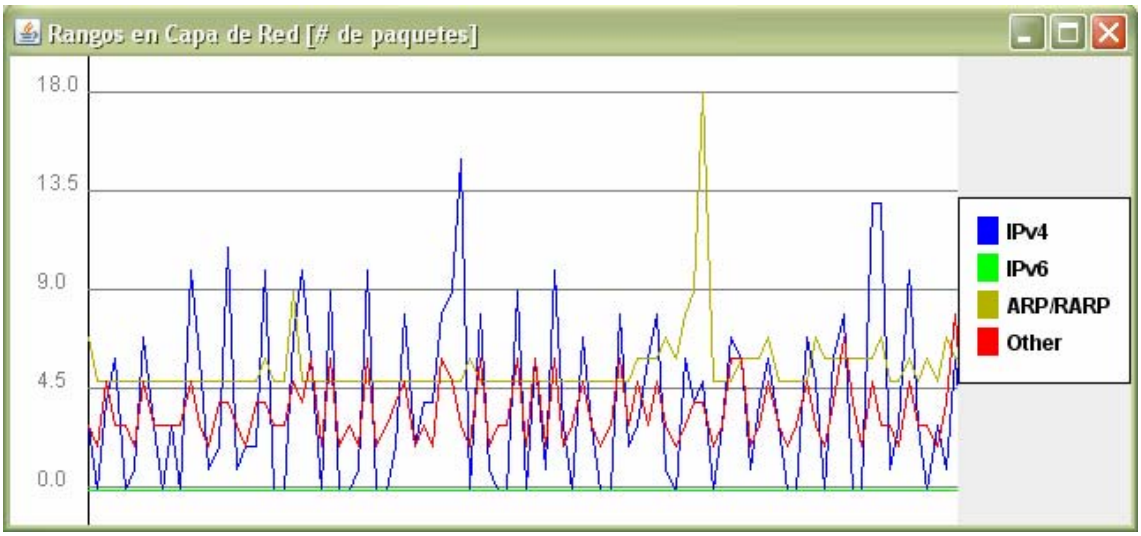


Rangos en capa de red

Diagrama de componentes Rangos en capa de red

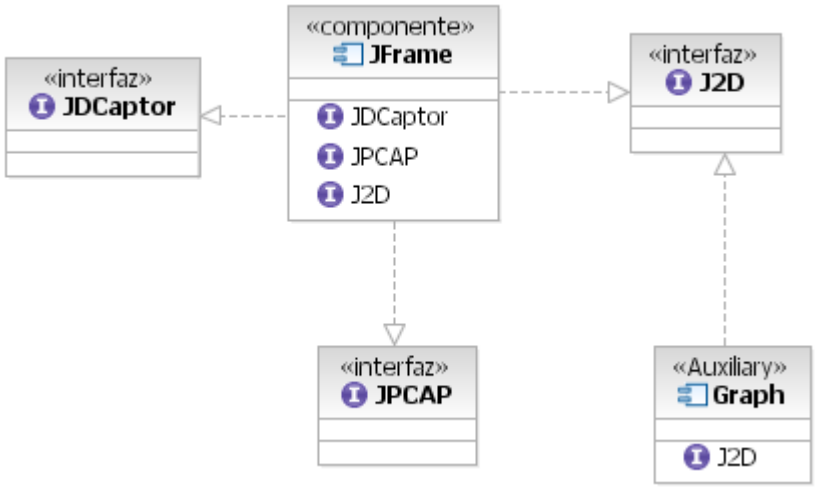


Diseño Rangos en capa de red

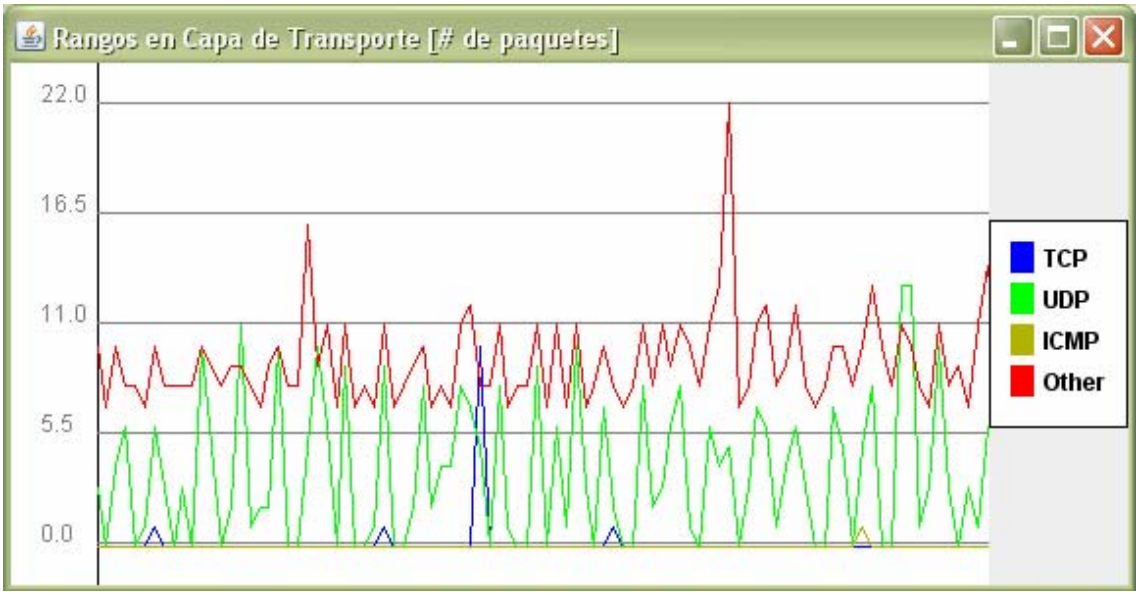


Rangos en capa de transporte

Diagrama de componentes Rangos en capa de transporte

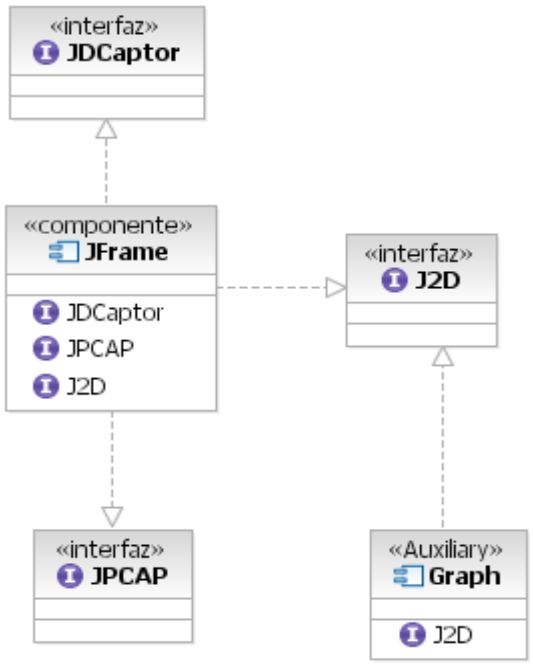


Diseño Rangos en capa de transporte

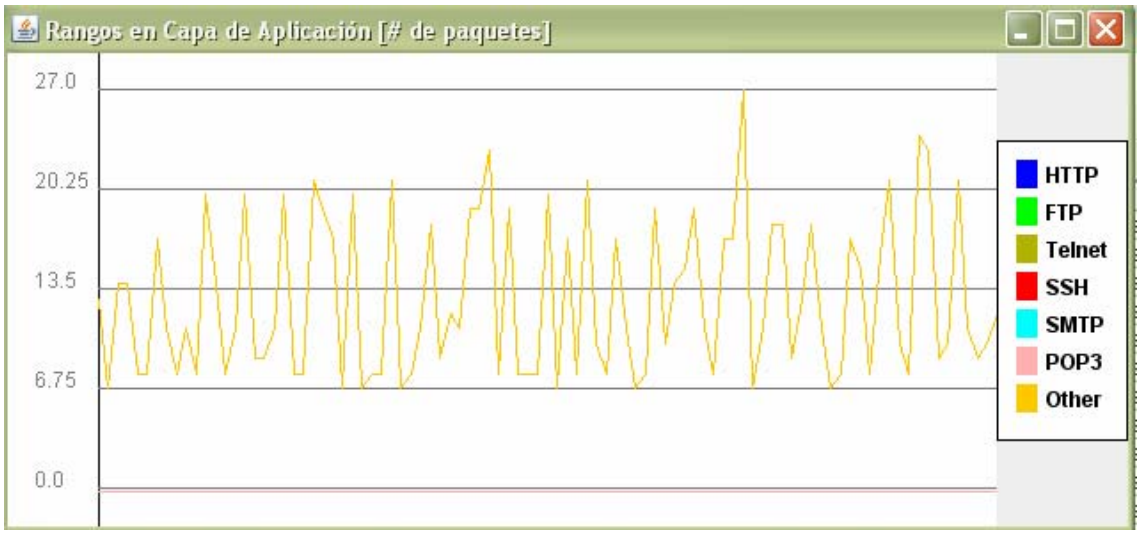


Rangos en capa de aplicación

Diagrama de componentes Rangos en capa de aplicación

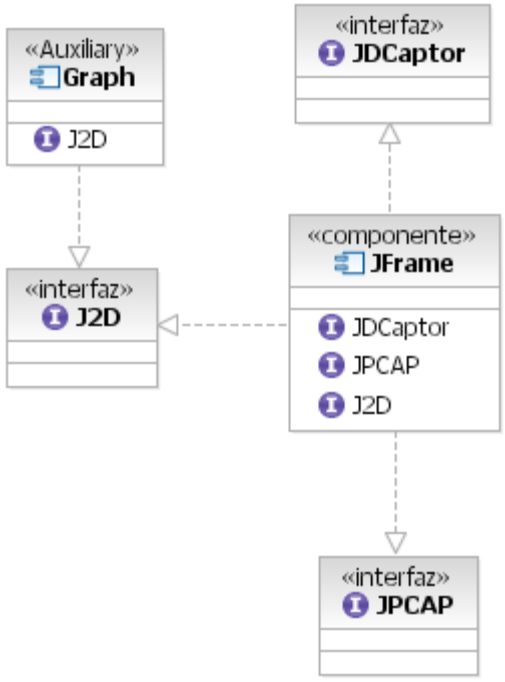


Diseño Rangos en capa de aplicación

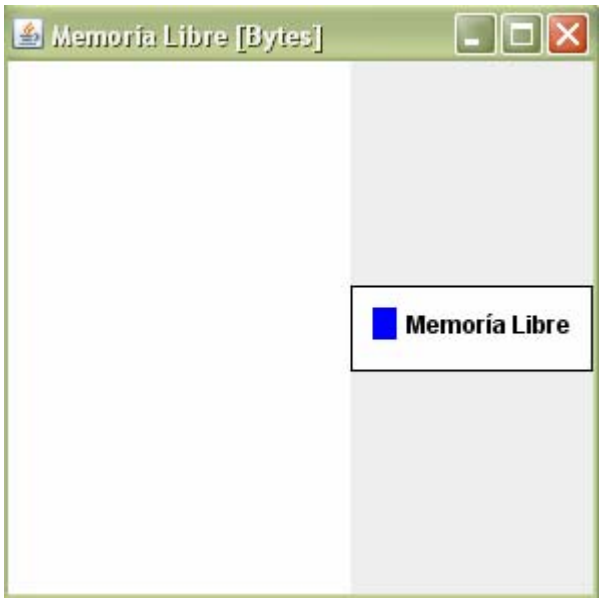


Memoria libre

Diagrama de componentes Memoria libre

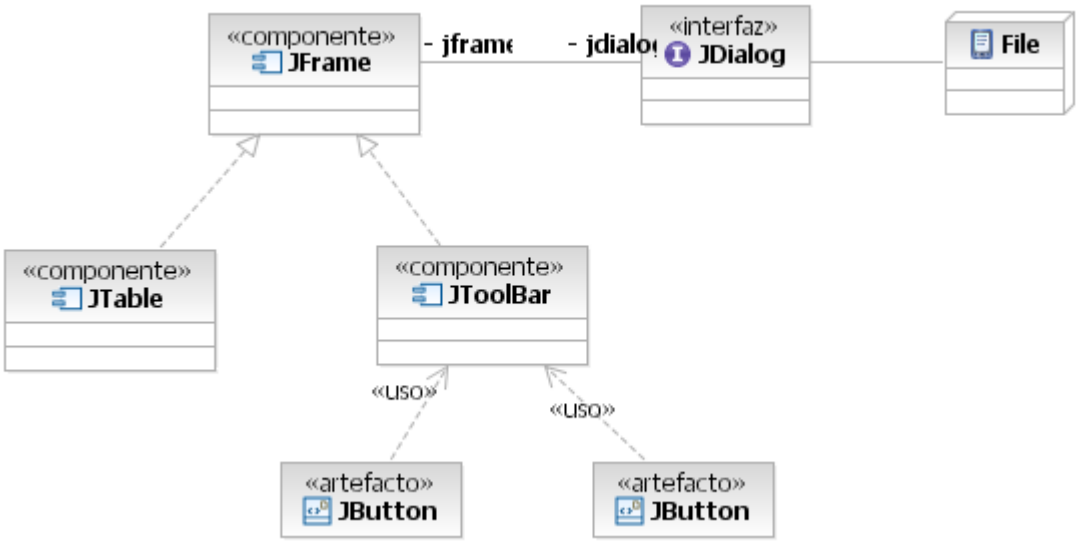


Diseño Memoria libre



Detección por Reglas

Diagrama de componentes Detección por Reglas

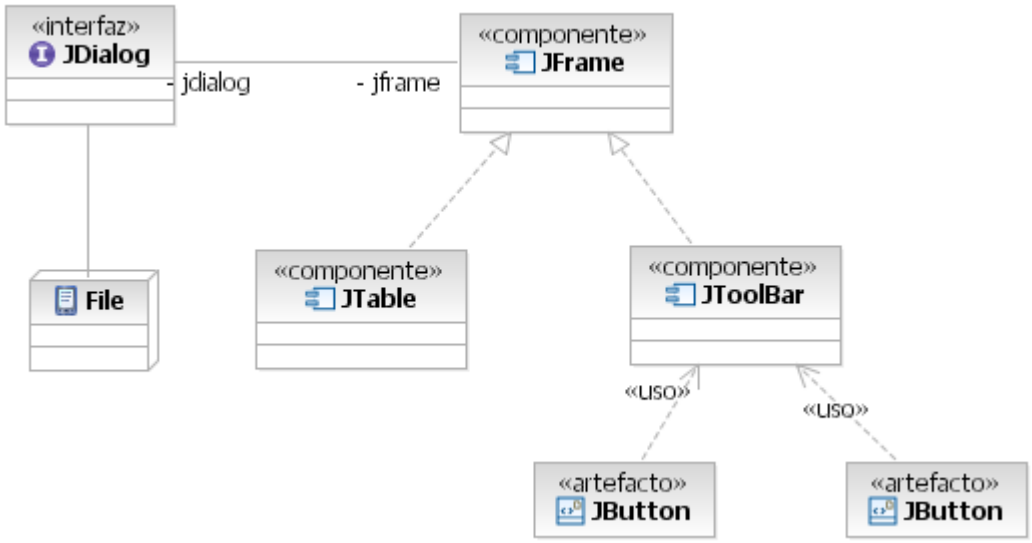


Diseño Detección por Reglas


Reglas para detección de intrusos
Puerto
Protocolo
Descripción
6 39 SubSARI
6 41 Deep Throat, Foreplay
6 44 Arctic
6 48 DRAT
6 50 DRAT
6 53 ADM worm, Lion
6 58 DMSetup
6 59 DMSetup
6 69 BackGate

Detectar intrusos por reglas

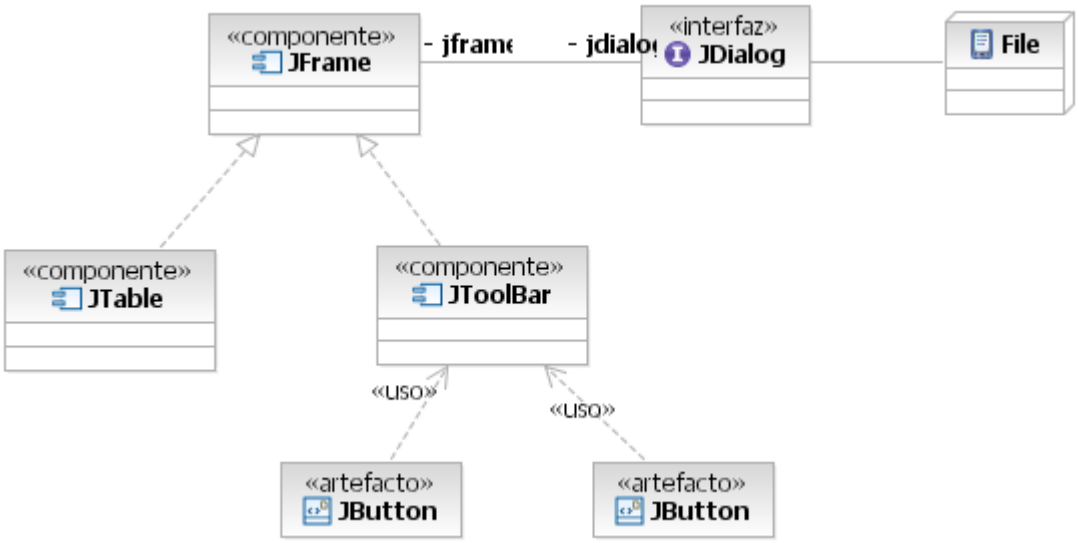
Diagrama de componentes Detectar intrusos



Detectar intrusos por reglas

Detección por entrenamiento neuronal

Diagrama de componentes Detección por entrenamiento neuronal



Diseño Detección por entrenamiento neuronal