Detección Estadística de Intrusiones en Redes de Datos

López Mariano Gabriel.
malo.gabi@gmail.com

Abstract: *Dado un constante crecimiento de aplicaciones cada vez mas interconectadas en donde los datos se encuentran disponibles para ser consultados desde una computadora personal fuera de la oficina hasta un teléfono celular, se vuelve cada vez más necesarias aplicaciones que detecten y brinden respuestas antes la intromisión en tiempo repuestas online. Este trabajo se propone el estudio y realización de un software que utiliza diferentes clasificadores basados en redes neuronales orientados a la detección de ataques del tipo denegación de servicio.*

Keywords: IDS, *Redes Neuronales, Ataques a Redes, Denegación de Servicio, Kolmogorov – Smirnov.*

1. Introducción

Es el desafío de todo administrador de redes trabajando en una empresa asegurar que los servicios informáticos que brinda mediante el uso de servidores (pagina web, correo, acceso al sistema interno, etc.) se mantengan siempre disponibles.

Cuando se implementa la posibilidad de brindar información entre miembros de una empresa surge la necesidad de que solo aquellos que estén autorizados puedan acceder y no todos, esto se realizaba así, hasta que con el advenimiento de Internet, el número de usuarios posibles se volvió incontrolable, los ataques se volvieron una verdadera pesadilla para los administradores.

Los ataques pasivos (que consiste en el monitoreo del tráfico saliente y entrante teniendo como objetivo obtener la mayor cantidad de información de mensajes transmitidos) son muy difíciles de reconocer y son un intento previo al ataque activo los cuales afectan y comprometen los pilares básicos de las prácticas de seguridad: la confidencialidad, la integridad y la disponibilidad.

Ante la posibilidad de que se presenten, los Firewall son la defensa, que como su traducción lo indica, son la pared de filtrado para el acceso al sistema. Si bien son herramientas muy poderosas, no se pueden adaptar a los cambios de proceder de personas que quieran atacar la red, y además se necesita de un administrador que actué (según se observe) agregando o quitando condiciones de acceso.

Con este problema surgen los IDS (Intrusion Detection Systems) son programas que tienen la función de detectar un posible ataque o intrusión, existen muchos tipos,

que se especializan en observar diferentes conductas que puedan ser el principio de un ataque. Una de las conductas que más se presenta es el de encontrarse con picos de tráficos en la red que no son normales, la pregunta es: … *¿cómo se sabe que tan anormal es un tráfico?,* en realidad, es un análisis muy complejo en el cual, hay considerar, que hora del día es, si el día es feriado o si es laborable, etc.

2. Descripción del proyecto, Antecedentes.

Existen numerosos trabajos donde se han tenido que analizar sistemas tan heterogéneos en su uso como son los tráficos en una red de datos; un servicio que se asemeja en su funcionamiento es el servicio de luz eléctrica.

Donde un IDS cumpliría la función de una alarma que detecte que algo no está bien, como por ejemplo, puede existir una demanda de corriente que no es acorde al día, al clima o a la hora, deduciendo que existe un exceso de consumo o un robo de energía[1].

En este caso un IDS no sería aprovechado, debido que la pérdida de algunos watt por algunos segundos, no se compara con la perdida de información por robo o destrucción de algunos megas-bytes. En la actualidad se conocen estudios con sus posteriores implementaciones en redes neuronales, sobre simulación del consumo eléctrico de una ciudad, discriminado por hora, día (ya sea feriado o no) y clima [2].

Se agrega a estos antecedentes, lo expuesto en su trabajo de investigación por C. Manikopoulos y S. Papavassiliou[3], que proponen un método para detectar intrusiones estadísticamente, usando como herramienta la función de Kolmogorov-Smirnov y las redes neuronales para modelar y detectar ataques.

Además el posterior trabajo de investigación de J. D. Britos[4], donde se implementa lo presentado por Papavassiliou en la Universidad Nacional de Córdoba con excelentes resultados.

Desde estos antecedentes, se aborda el presente trabajo de investigación y desarrollo de un IDS con arquitectura similar a la analizada e implementada por el Ingeniero Britos, sumando la posibilidad de:

- Realizar de manera automática comunicación con el Firewall para ser una herramienta de detección de intrusos que también realice el rechazo de amenazas sin la necesidad de la intervención de un administrador de red.
- Acceder desde el exterior, para constatar online el estado de la red, las medidas tomadas por la aplicación, historiales, etc.
- Implementar en Java para lograr una multiplataformidad del sistema.
- Contar con un soporte de Base de datos que almacene historiales de medidas y de tráficos, ataques detectados, medidas tomadas tanto por el administrador como por la red, para ser utilizadas en el momento que se desee efectuar el reentrenamiento de la red neuronal.

- Disponer de herramientas estadísticas, graficas y comparativas de los historiales.
- Controlar el logueo de usuarios, configurando diferentes perfiles y accesos.

3. Aspectos Metodológicos Relevantes.

El proyecto vincula disciplinas variadas y distintas, por esta razón se utilizarán diferentes metodologías de desarrollo para cada parte del sistema. Para identificar cada una podemos clasificar las actividades los objetivos.

3.1. Etapa de Simulación. (Modelos sacado de Wiston – Hillier. Investigación de Operadores[5]). En esta etapa se implementará las fases de prueba y experimentación de los diferentes modelos de redes neuronales. Iniciando con la presentación del problema hasta llegar a la implementación mediante software, y pruebas de verificación y validación.

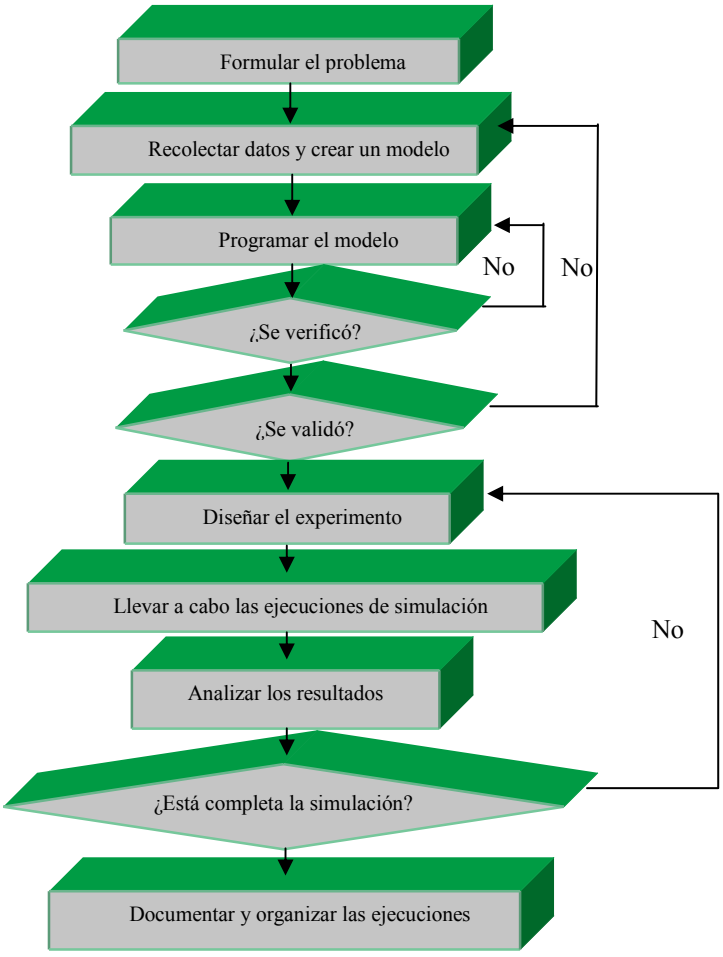


Fig. 1. Metodología de Simulación de Wiston – Hillier

3.2. **Etapa de confección de la red neuronal.** Como base se utilizó el modelo metodológico presentado en el Libro Redes Neuronales y Sistemas Borrosos [1], donde se plantea la división del trabajo en 9 pasos bien definidos a los que se le suman 2 pasos más con el objetivo de lograr la integración con la GUI del sistema. Para lograr dicha integración entre las partes, se dividió las etapas de desarrollo de la red, en los flujos de trabajo planteados en UML [6]. Así de esta manera se logra paralelismo en el desarrollo de los diversos componentes, lográndose mitigar los riesgos y minimizar los tiempos de desarrollo.

Modelo de Requerimiento

a. Planteamiento del problema. Divide y vencerás.

Es necesario realizar una descripción detallada del problema para analizar si algún aspecto podrá ser resuelto mediante un módulo neuronal. No se debe pensar que las ANS (Redes Neuronales) son la mejor solución en todos los casos [1].

b. Requerimiento del sistema.

En este punto se debe responder de la forma más concreta posible a las especificaciones que se debe cumplir en el sistema. Por ejemplo: es necesario plantear la cota de error que se desea alcanzar, el tipo de formulación que se va aplicar (predicción, clasificación, series temporales, ajustes funcionales, procesado de señales, etc.), la forma en la que se dispondrán los datos, el tiempo de respuesta requerido, los equipos informáticos necesarios disponibles, etc.

c. Revisión bibliográfica.

Una buena práctica es realizar un sondeo en busca de alguna aplicación parecida a la que se plantea. Observando la estrategia seguida por otras personas en problemas similares, obteniendo un punto de arranque para el trabajo con datos propios.

Modelo de Diseño

d. Elección del modelo de ANS.

Una vez se ha especificado con detalles las características del problema, se elegirá un modelo de ANS, para comenzar con las pruebas.

e. Datos disponibles y selección de variables relevantes.

Es preciso saber la forma en la que se va a disponer de los datos, viendo la posibilidad de contar con un procesamiento de datos en Línea o por lotes, además se necesitara conocer la cantidad de ejemplos que se van a poder emplear en el entrenamiento. En el caso de ser un número pequeño quizás haya que aplicar alguna técnica específica de entrenamiento para pocos patrones.

f. Elección de los conjuntos de aprendizaje y test.

En este punto se elige cual será el conjunto que se destina al entrenamiento y cual a la verificación. Una de las decisiones más difíciles en el desarrollo de una aplicación de red neuronal es saber cuándo detener el entrenamiento, ya que después de una determinada interacción, la red empieza a memorizar los datos con la que se la entrena.

Cometiendo luego errores, cuando se deben analizar con los que no fue entrenada, es por esto que se elige un pequeño grupo de entrenamiento y un grupo mayor de "test", este segundo grupo se utiliza para cotejar la capacidad de reconocimiento de nuevos casos.

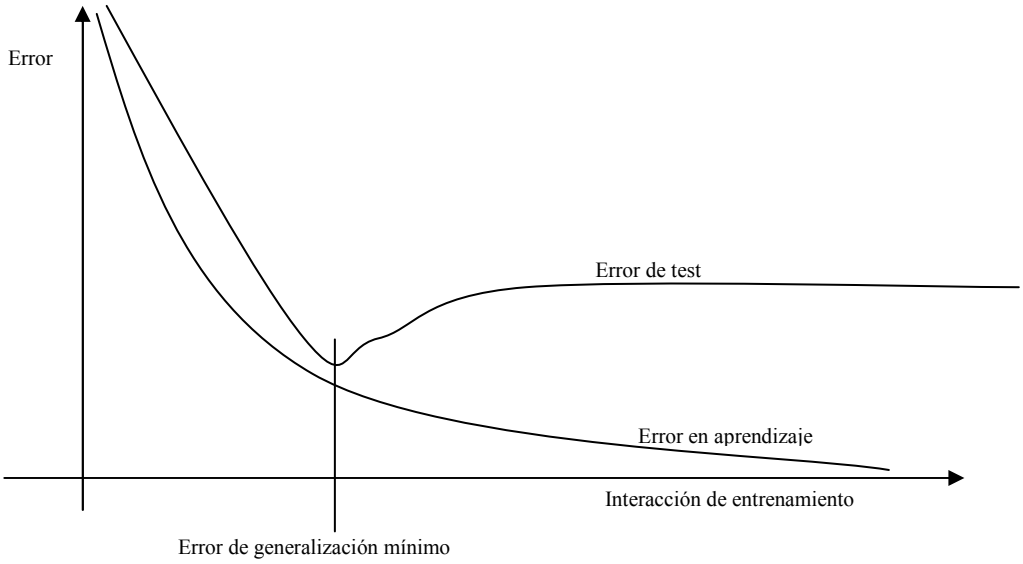


Fig. 2. Errores de test y aprendizaje

Cuando se realiza el proceso de entrenamiento se debe convivir con dos tipos de errores, el error de test y error de aprendizaje, el primero define la magnitud del error en el proceso de entrenamiento, y el segundo define como la red reacciona a nuevos ejemplos que no conoce. Es el objetivo de esta etapa reconocer el punto mínimo en que el error de test y aprendizaje se encuentran (Error de generalización) [1].

g. Pre-procesamiento.

Se denomina pre-procesamiento al tratamiento previo de los datos de entrada y salida para adecuarlos a su tratamiento por la red neuronal, para trabajar con ANS es aconsejable conseguir que los datos que se proporcionan a la red posean las siguientes cualidades [1]:

- Buena Distribución.
- Rangos de valores parecidos para todas las entradas.
- Rangos acotados dentro del intervalo de trabajo de la función de activación empleada en las capas ocultas y de salida ANS.

Proceso de entrenamiento.

Es el punto determinante de todo el desarrollo de aplicación, en este proceso existe una interacción entre los dos conjuntos de patrones (aprendizaje y test), la estructura del ANS y el experimentador, se debe ensayar con diferentes topologías y con distintos parámetros de aprendizaje, medir el error de aprendizaje y el de generalización, hasta alcanzar los deseados.

h. Evaluación de los resultados.

Finalizada la fase de entrenamiento y almacenados los pesos ideales en la anterior etapa, ya se está en disposición de aplicar el ANS sobre nuevos casos (no empleados en el entrenamiento) para medir su eficacia de forma objetiva. Si se comprueba que se siguen obteniendo resultados dentro del margen de error deseado, se puede proceder a emplear el ANS dentro del entorno de trabajo.

3.3 **Etapa de desarrollo de la Interfaz de Usuario (GUI).**

Para esta etapa se emplea la metodología utilizada por excelencia para el desarrollo de software, el desarrollo orientado a objetos. Tomando como metodología el proceso dirigido por casos de usos, centrado en la arquitectura e interactivo e incremental. Y como herramienta de modelado se utiliza el Lenguaje Unificado de Desarrollo o UML (por sus siglas en ingles). Cabe aclarar que no se utilizará el total de las herramientas y modelados que UML propone, solo las que poseen un significado al desarrollo del sistema, según cada uno de los flujos de trabajo se utiliza las siguientes herramientas:

- Modelado de Requerimientos.
  - Modelo de Casos de uso.
  - Descripciones de casos de uso (nivel sistema)
  - Listado de requerimientos funcionales.
  - Listado de requerimientos no funcionales.

- Modelado de Análisis
  - Modelado de objeto del dominio del negocio.
  - Diagrama de clases del análisis.
  - Diagramas de colaboración del análisis.
  - Identificación de los diversos paquetes.

- Modelado de Diseño

- Modelado de clases del diseño.
- Modelo de despliegue. Determinación del ambiente de implementación.
- Identificación y descripción del almacenamiento.
- Diagramas de estados.
- Relaciones entre Objeto – Entidad.

❖ Modelo de implementación:
- Manual de usuario.
- Ejecutable.

4. Descripción de los elementos del sistema.

Dentro del recuadro se puede observar un pequeño diagrama conceptual del sistema, teniendo como ambientes internet, una base de datos y Firewall. La herramienta más utilizada para la configuración del Firewall es IP-Tables, el proyecto se encuentra distribuido en 5 capas bien identificadas con objetivos bien definidos para cada una.

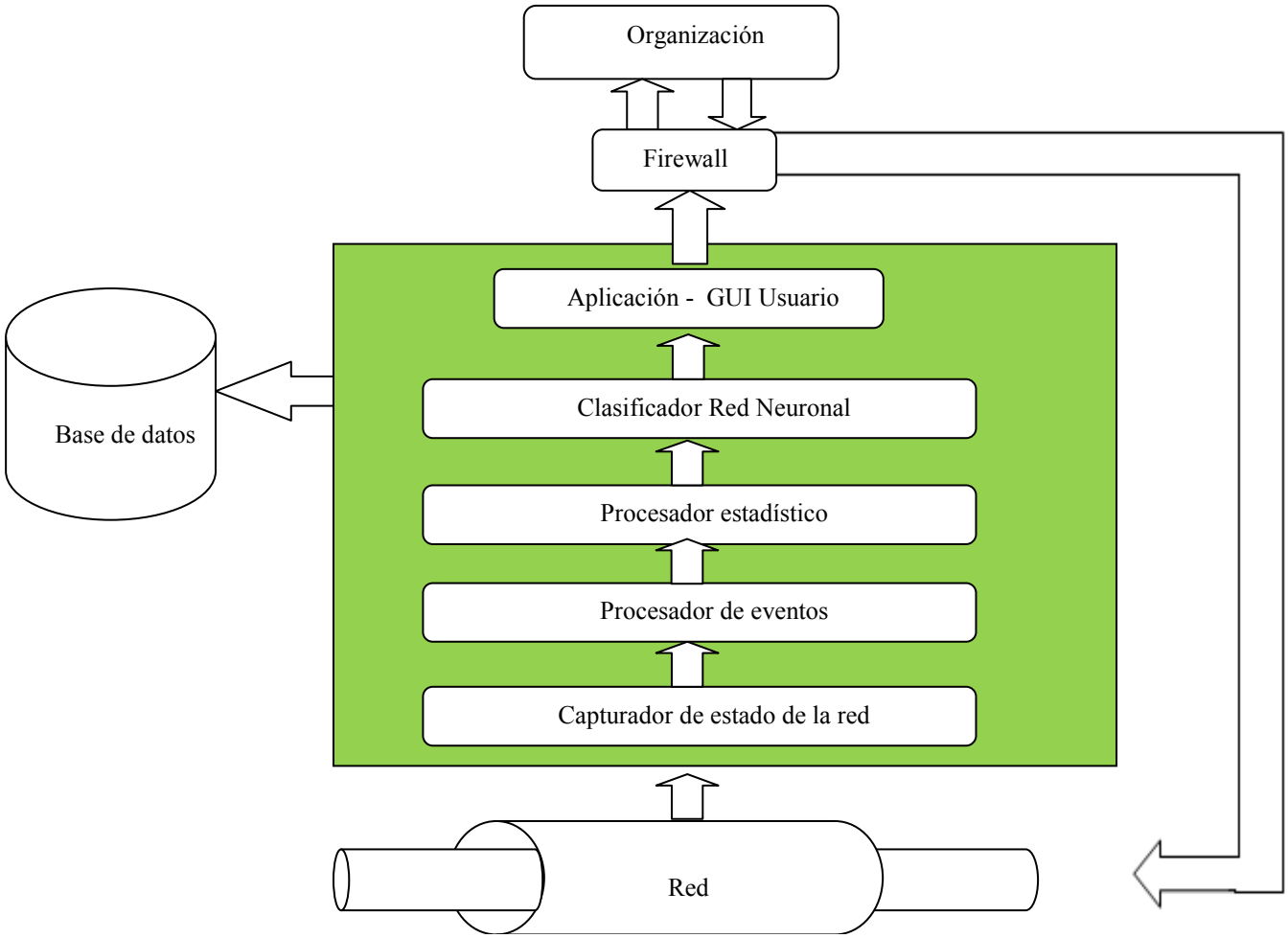


Fig. 3. Esquema de elementos del Sistema

4.1 **Capturador de estado de la red**

El objetivo de esta capa, es capturar en tiempo real, el estado de tráfico de la red. El mismo deberá capturar muestras de los siguientes datos:

Tab. 1. Parámetros a Capturar

Parámetro	Protocolo	Unidades
Tráfico	UDP	Mbps
Longitud de paquete	UDP	Bytes
Tráfico	UDP	Paquetes por segundo
Tráfico	IP	Mbps
Longitud de paquetes	IP	Bytes
Tráfico	IP	Paquetes por segundo

Son un total de 6 parámetros, discriminando de acuerdo al protocolo UDT o IP, las siguientes variables:

Tráfico: se emplean dos variables para medir el tráfico, en megabits por segundos y la segunda variable mide la cantidad de paquetes por segundo sin importar el tamaño de estos. Ambas tienen como función medir la cantidad de información en un determinado momento.

Longitud de paquete: Mide el tamaño de paquetes en byte.

4.2 **Procesador de eventos:**

Se toma como entrada una matriz de capturas de la etapa anterior, el cálculo es simple y se trata del consiente entre el máximo valor observado en cada uno de los parámetros divididos la cantidad de ellos. Este número representa un peso estadístico de cada uno de los parámetros.

4.3 **Procesador estadístico:**

Los ataques pueden durar no solo unos pocos segundos, sino que pueden durar varios minutos, incluso horas, es necesario un esquema que permita relacionar las diversas muestras en forma conjunta y en diversos momentos de tiempo. En esta etapa recibimos los parámetros actuales de la anterior capa, en diversos intervalos de tiempo y los enviamos a la capa subsiguiente.

4.4 **Clasificador Red Neuronal:**

En esta etapa comparamos cada uno de los parámetros que recibimos de la capa anterior y los sometemos a evaluación de la red neuronal, la red comparará las diferentes entradas con las que reconoce, posible ataque y tráfico normal dando una salida de entre 0 y 1 (1 como absoluta certeza de ataque y un valor de 0 como absoluta certeza de tráfico normal).

4.5 **Aplicación – GUI Usuario:**

En esta etapa es donde se efectúa el control de usuarios, almacenamiento de los diferentes parámetros en base de datos, teniéndo como objetivo dos importantes salidas; por un lado el nivel de amenaza y por el otro la regla de ip-table que brinde la solución. Como objetivos se puede enumerar la generalización de informes históricos de medidas, tráfico, estados de la red y amenazas detectadas. También se ofrece un esquema de reentrenamiento de la red de manera automática.

5. Conclusión

Al momento de redacción del presente informe de investigación se ha logrado avanzar en los siguientes puntos:

1) Desarrollo del Módulo Capturador de Estado de la Red.
2) Desarrollo de Módulo Procesador de Eventos.
3) Entrenamiento de la Red Neuronal en un ambiente de prueba (MATLAB).
4) Desarrollo del Módulo Clasificador Red Neuronal.

Como resultado parcial se observan un muy buen comportamiento del sistema a pesar de no encontrarse completamente desarrollado, esta etapa del trabajo se ha centrado principalmente sobre el desarrollo de la Red Neuronal (ANS). Se puede concluir el logro de presentar de manera objetiva a las ANS, no como la solución de todos los problemas sino como una herramienta útil para encarar problemas muy puntuales.

Son de extrema utilidad para enriquecer un sistema pero no son un sistema en sí mismas, pueden verse como un módulo dentro de un sistema más amplio en donde se realicen diferentes análisis con el objetivo de lograr reconocer ante un conjunto de entradas cual es la salida más correcta aun cuando no se hayan entrenado para ese caso en concreto.

6. Bibliografía

[1] Redes Neuronales y Sistemas Difusos. 2 edición. Bonifacio Martin del Brío y Alfredo Sanz Molina. 2002.

[2] Martín del Brío, B. N Medrano, I. Ramírez, J.A. Domínguez, J. Barquillas, J. Blasco, J. García. Short-term electric power load-forescasting using artificial neural networks. Part I: Self-organizing networks for classification of day-types. 14th IASTED Int. Conf. on Modelling, Identif and Control, Igst, Austria, febrero 1995.

[3] C. Manikopoulos and S. Papavassiliou, .Network intrusion and fault detection: a statistical anomaly approach,. Communications Magazine, vol. 40, pp. 76.82, Oct. 2002.

[4] Detección Estadística de Intrusiones en Redes de Datos. J. D. Britos. 2001.

[5] Apuntes cátedra Simulación de sistemas 4to año Ingeniería en sistemas prof: Ingeniero Cristina Sandri. Año 2006 U. T. N. Villa María.

[6] El proceso Unificado de Desarrollo de Software. Ivar Jacobson, Grady Booch, James Rumbaugh. 1999.

[7] Fundamento de Seguridad en Redes Aplicaciones y Estándares. Willian Stalling. 2005.

[8] Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering. 1996 Massachusetts Institute of Technology. Nikola K. Kasabov.

[9] Machine Learning, Neural and Statistical Classification. Editors: D. Michie, D.J. Spiegelhalter, C.C. Taylor. February 17, 1994.

[10] Classification, Parameter Estimation and State Estimation. F. van der Heijden, R.P.W. Duin, D. de Ridder and D.M.J. Tax. 2004 John Wiley & Sons.

[11] Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives. Dolores DeGroff. 07/01/94.

[12] C++ Neural Networks and Fuzzy Logic. Valluru B. Rao. 06/01/95.

[13] Fuzzy Inference Neural Network for Fuzzy Model Tuning. Keon – Myung Lee, Dong – Hoon Kwak, and Hyung Lee – Kwang. Agosto 1996.

[14] COMPUTATION AND NEURAL SYSTEMS SERIES. California Institute of Technology. Christof Koch. 1991.

[15] Inteligencia Artificial y Redes Neuronales. M. Daniela López De Luise-EnterpriseWare SRL-Junio de 2001.

[16] Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications. Lakhmi C. Jain; N.M. Martin. 11/01/98.

[17] Artificial Neuronal Networks. Michigan State University. Jianchang Mao, K. M. Mohiuddin y IBM Alamden Research Center. 1996.

[18] Sistemas Inteligentes baseados em Redes Neurais Artificiais aplicados ao Processamento de Imagens. Fernando Osório & João Ricardo Bittencourt. 2000.

[19] J. P. Anerson, "Computer security technology planing study," ESD-TR-73-51 Electronics System Division(AFSC), vol. 1, pp. 1–32, October. 1972

[20] S. F. Kenneth Ingham, "A history and survey of network firewalls," ACM, vol. V, no.˜N, pp. 1–42, 2002.

[21] Apuntes cátedra Probabilidad y estadísticas 2do año Ingeniería en sistemas prof: Ingeniero Carlos Colazo. Año 2004. U. T. N. Villa María.

[22] Sistema de Detección de Intrusiones y Anomalías en redes informáticas mediante una aproximación estadística. Javier Bongiovanni, José Luis Regali. 2005.

Anexo

Introducción Experimentaciones realizadas:

A continuación se especificaran el conjunto de pasos realizados durante el proceso de entrenamiento. Se verán en una serie de 5 puntos en donde se especificaran las herramientas utilizadas, el ambiente en donde se especificaron las pruebas, actividades, resultados obtenidos y metodología empleada.

Antes que nada cabe agradecer a los aportes desde el Centro de Informático de Comunicaciones (CICOM) de la Facultad Regional Villa María, por prestar sus instalaciones y equipo, a la hora de dar lugar a las pruebas test de la red interna de la facultad. También cabe agradecer los aportes del Ing. Britos al momento de de brindar información de manera desinteresada sobre el desarrollo de entrenamientos similares, siendo el utilizado como base para la presentación el trabajo realizado por los Ingenieros Javier Bongiovanni y José Luis Regale en su trabajo final de grado de Ingeniería Electrónica en la Universidad Nacional de Córdoba[22].

Ambiente

Como requerimiento indispensable para la realización del entrenamiento de la red neuronal, era necesario poseer un conjunto suficientemente grande de muestras de tráficos (discriminadas por protocolo y en las unidades necesarias) de carácter "normal" con el objetivo de poseer una base de conocimiento. Como definición de "lugar", se llego a la conclusión que debía poseer las siguientes características:

- Un flujo de tráfico superior a 500 kb.
- Una red de datos con una gran cantidad de servicios corriendo en un mismo instante.
- Un conjunto de usuarios que utilizaran la mayor cantidad de servicios posibles al mismo tiempo.
- Una gran variabilidad en los flujos de tráfico.

- La existencia de un punto en donde pasara todo el tráfico tanto de entrada como de salida (llamaremos a este punto como "X").
- Poseer acceso al punto "X" en varios instantes de tiempo buscando poseer una base de conocimiento en cuando la red se encuentre siendo infra usada, en una "meseta" o en cuello de botella.

Los parámetros analizados (como anteriormente se mencionó) son:

Parámetro	Protocolo	Unidades
Tráfico	UDP	Mbps
Longitud de paquete	UDP	Bytes
Tráfico	UDP	Paquetes por segundo
Tráfico	IP	Mbps
Longitud de paquetes	IP	Bytes
Tráfico	IP	Paquetes por segundo

A continuación se ampliara como mediante la utilización de pcap se logro la captura de los parámetros necesarios.

Biblioteca de funciones PCAP.

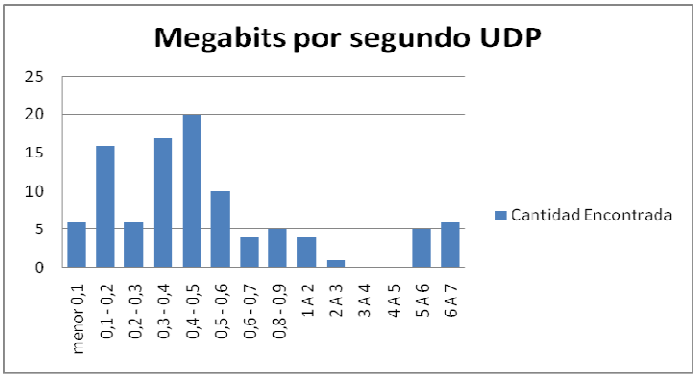
Pcap provee una interfaz de alto nivel para sistemas de captura de paquetes. Todos los paquetes de la red, incluso aquellos destinados a otros anfitriones son accesibles a través de este mecanismo.

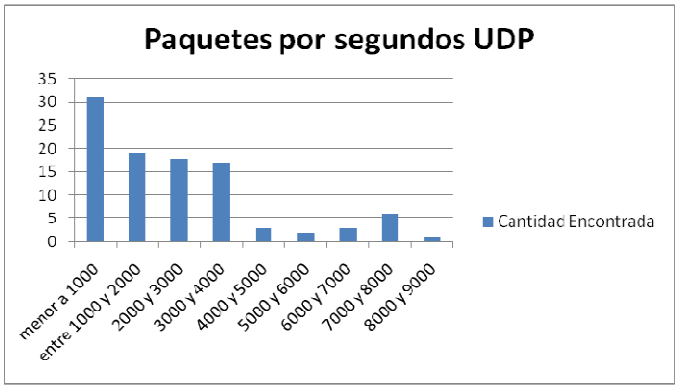
Existen 3 parámetros que regulan el comportamiento de programa:

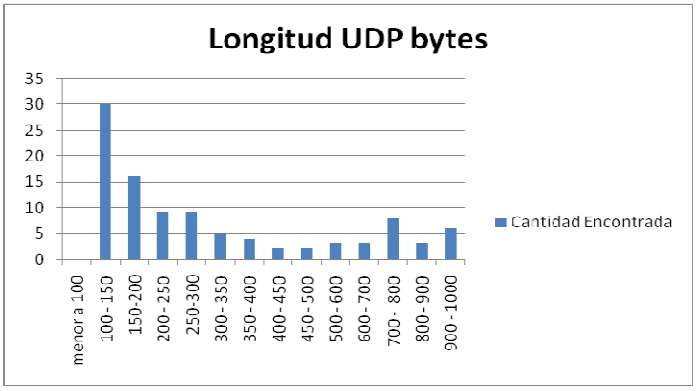
PAQUETES_POR_MUESTRA = 10
Muestras_Por_PDF = 20
NUM_PDF = 10

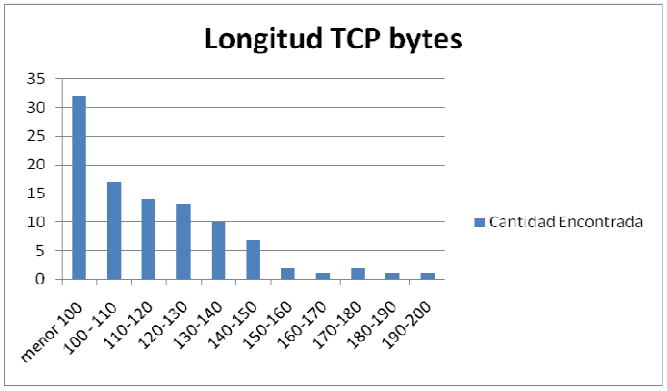
Los cuales hacen que el capturador, captura 10 paquetes en la red, luego con ellos calcule 1 muestra y con 20 muestras calcule un total de 10 PDF. Este esquema jerárquico simplifica no solo los números con los que se trabajan en el programa sino que además efectúa una salida lo suficientemente general que no provoca una congestión en la maquina, haciendo de este un proceso liviano en el servidor.

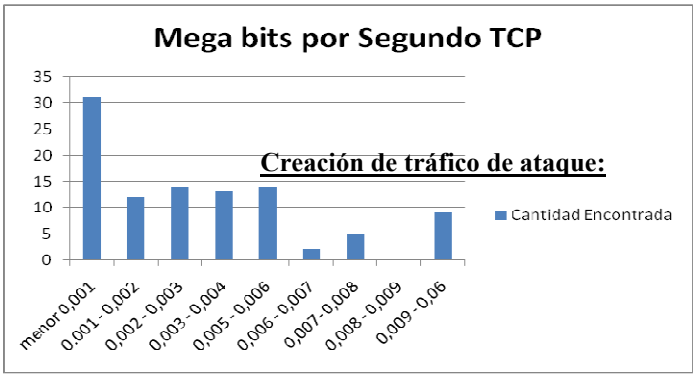
Trafico encontrado:



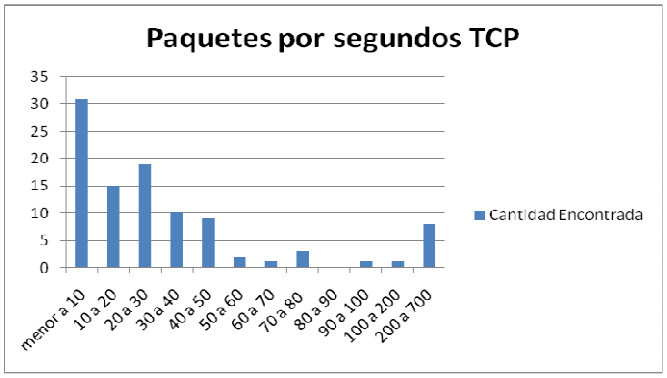








Creación de tráfico de ataque:



Basándose en los resultados obtenidos por la tesis de grado de Javier Bongiovani y José Luis Regali, utilizamos mediante cálculo algebraico el tráfico resultante de un ataque. El cálculo se puede resumir de la siguiente manera:

1. Tomar el tráfico normal como base.
2. Siendo el trafico de inundación constante en el tiempo (tanto en velocidad de información como cantidad) se puede expresar al tráfico resultante como:

   t -> infinito

   Long_Paquete(t) $\simeq$ Longitud de paquetes de ataque.

   BitsXSegundo(t) $\simeq$ Velocidad de ataque

   PaquetesxSegundo(t) $\simeq$ Velocidad de ataque de paquetes.

   Siendo:

   Long_Paguete(t) = función en el tiempo de longitud en bytes en un determinado tiempo.

   BitsXSegundo(t) = función en el tiempo de velocidad en megabits por segundo.

   PaquetesxSegundo(t) = función en el tiempo de paquetes por segundos encontrados.

3. Siendo que el programa captura al alcanzar un número fijo de paquetes, la muestra obtenida bajo ataque se puede definir como:

   Siendo:

   TraficoTotal = TraficoNormal + TraficoAtaque.

   O en termino Matemático

   MuestraTotal = FunciónOcurrenciaNormal(t) + FunciónOcurrenciaAtaque(t)

Siendo el objetivo del sistema captar cuando existe una "FunciónOcurrenciaAtaque(t)" a través de la muestra total.

4. Con motivo de simulación se puede tomar a la funciónOcurrenciaAtaque(t) como una simple función monotamente creciente (como en la mayorías de los programas generadores de ataque) en donde:

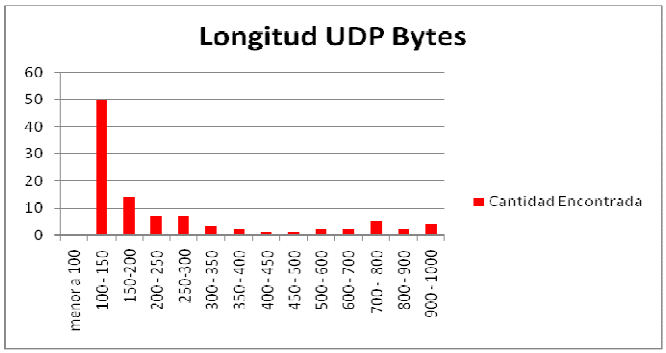
FunciónMonotonaAtaque(t) = MuestraTotal(t)

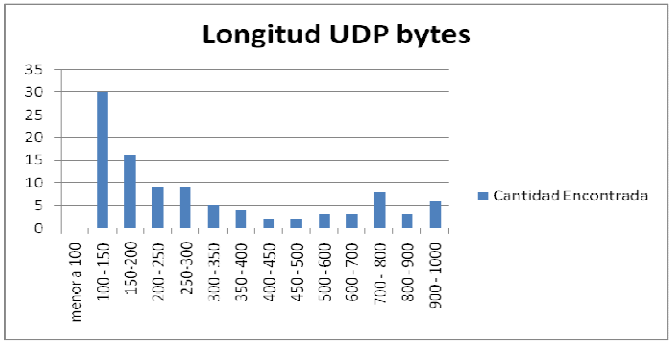
Cuando t = infinito y FuncionOcurrenciaNormal(t) ≃ 0.

En donde estamos en presencia de una red sobre saturada ya que todo el tráfico entrante es información de ataque.

5. Siendo ciertos los puntos 1 al 4, podemos definir una función lineal de pendiente k como nuestra función de ataque.

Tomando las 5 supuestos se calculo los gráficos de tráficos de ataque con un ataque con una longitud 105 bytes y 0,45 Megabits por segundo:





Arriba se observa dos gráficos, en el primero se observa un tráfico de red que se encuentra atacado (bajo simulación) y en el segundo se observa la misma red con tráfico normal (color azul)

Creación de la red neuronal:

- Como primer experimento se introdujo a la red neuronal un conjunto de 400 muestras con entrenamiento supervisado invocando a que sea 0 en sus salidas. Después de corrido el entrenamiento se obtuvo los siguientes pesos:

Primer Capa:

	Neur 1	Neur 2	Neur 3	Neur 4	Neur 5	Neur 6.
Neur. 1	0.0002	0.0491	-0.0006	-0.0165	-434.5888	0.0534
Neur. 2	0.0021	-0.1785	-0.0079	0.0116	-57.1800	0.0102
Neur. 3	-0.0057	1.2649	0.0000	0.0219	-149.3827	-0.1288
Neur. 4	0.0059	0.9607	0.0005	0.0050	-205.3640	0.0161
Neur. 5	0.0048	2.1613	0.0025	-0.0476	307.3191	0.0094
Neur. 6	0.2169	1.7610	-0.0052	0.0008	207.9566	0.0435

Segunda Capa:

	Primer capa					
	Neur1	Neur2	Neur3	Neur4	Neur5	Neur6
Nuer1 (segunda capa)	0.0000	-0.0000	0.0000	0.0000	0.0000	0.3123

- Como segundo experimento se tomo otras 50 muestras de las que fue entrenada para observar el nivel de memoria de la red. Obteniéndose los siguientes resultados:

0.0005	0.0004	-0.0001	0.0002	0.0000	0.0000	0.0002	0.0004	0.0002
0.0001	0.0002	0.0002	0	0.0011	0.0000	0.0001	-0.0007	0.0004
0.0002	0.0003	0.0001	0.0006	0	-0.5175	0.0010	-0.0000	-0.0005

-0.0000	-0.0003	0.0000	0.0005	0.0001	0.0001	0.0005	0.0002	-0.0107
0.0002	-0.3439	0.0000	0.0005	-0.0000	0.0001	-0.0007	0.0004	0.0002
0.0000	-0.0002	0.0001	-0.0001	0.0001				

Dando como salida promedio "-0.017308", siendo 0 el valor ideal se obtuvo un buen reconocimiento de entradas.

- Como tercer paso se introdujeron tráficos reconocidos como ataque para observar la reacción del sistema. Cabe aclarar que se realizo primeramente sin entrenar la red para que los reconociera, a lo que se obtuvo una breve suba en el promedio pero no lo suficientemente significativa. Pero luego de reentrenar la red neuronal con ejemplos de ataque y sin ataque (400 sin ataque y 100 de ataque) se obtuvieron los siguientes resultados en los pesos:

Primer Capa:

	Neur 1	Neur 2	Neur 3	Neur 4	Neur 5	Neur 6.
Neur. 1	0.0156	-7.4228	-0.0003	-0.0426	-434.5747	0.0115
Neur. 2	0.0347	-0.2095	0.0452	0.0204	-57.1797	0.0011
Neur. 3	-0.3588	1.3545	0.0204	0.0788	-149.3768	0.0552
Neur. 4	1.7246	0.9643	-1.1263	0.5123	- 205.3649	0.3249
Neur. 5	0.0246	-1.6050	0.0004	-0.0000	307.6344	-0.0285
Neur. 6	0.2169	1.7610	-0.0052	0.0008	207.9566	0.0435

Segunda Capa:

	Primer capa					
	Neur1	Neur2	Neur3	Neur4	Neur5	Neur6
Nuer1 (seg capa)	0.41129	-0.016446	-0.0041005	-0.018341	-4.8063	2.922

- Como comprobación final se introdujo en la red 100 muestras en donde se encontraban las primeras 50 de ataque y el resto tráfico normal con motive de observar los resultados.

Salidas de la red Neuronal después de ingresada muestras de ataque:

0.998270305024771	0.779487829016372	0.76975996615913
0.675694931517698	0.992695190253866	0.99998808670692
0.975454488291288	0.999919664682642	0.980873484359869
0.99962752351076	0.996729134590256	0.828227755382083
0.999765784837078	0.940354336632012	0.921340005036941
0.999949327021927	0.999449030785898	0.977425458245032
0.99997457951581	0.995726089607996	0.997974415373526
0.999986807246816	0.997176966069491	0.993709606374983
0.998975111274559	0.657124247956089	0.229229913077919
0.591239407753064	0.66043498824742	0.999990283013933
0.832737040225924	0.992076475602219	0.905861579990861
0.9881998545095	0.996319686513532	0.946134056362785
0.990685958217756	0.995654973144262	0.960220205763501
0.781538795315338	0.952491062891743	0.676451663589111
0.986676611337182	0.324032769915699	0.730871435046238
0.659882765751311	0.781335009292268	0.843605523867672
0.864418447250361	0.664764672985025	

Promedio = 0,8766

Salidas de la red Neuronal después de ingresadas muestras normales:

0.6849	-0.00050879	0.22953	0.0079781	0.3437	0.066339	0.02898
0.16923	0.039047	0.017415	0.014598	0.58442	0.6604	0.007694
0.61139	0.52365	0.1431	0.072176	0.041123	0.25921	0.00857

0.54799 0.66043 0.66044 0.11163 0.025087 0.013423 0.03133
0.01077 0.20286 0.0088198 0.12555 0.00052041 0.0076968 0.007738
0.18756 -0.028982 0.0054258 0.36377 0.22803 0.0809 0.015424
0.039377 0.66014 0.0014111 0.67997 -0.00050862 0.0038941 0.66049
0.20885

Promedio = 0,17328

Se puede observar un resultado muy bueno a pesar de la utilización de solo 7 neuronas (6 en la primer capa y solo 1 en la capa oculta).