PROTOTIPO DE DETECCIÓN DE ATAQUES DISTRIBUIDOS DE DENEGACIÓN DE SERVICIOS (DDOS[1]) A PARTIR DE MÁQUINAS DE APRENDIZAJE

Manuel Sebastián Hoyos Llanos

UNIVERSIDAD AUTÓNOMA DE MANIZALES

MAESTRIA EN GESTIÓN Y DESARROLLO EN PROYECTOS DE SOFTWARE

MANIZALES

2015

[1] Distributed Denial of Service

PROTOTIPO DE DETECCIÓN DE ATAQUES DISTRIBUIDOS DE DENEGACIÓN DE SERVICIOS (DDOS) A PARTIR DE MÁQUINAS DE APRENDIZAJE

Manuel Sebastián Hoyos Llanos

Trabajo de Grado para optar por el título de Magíster en Gestión y Desarrollo de Proyectos de Software

Tutor: Gustavo A. Isaza Echeverri, PhD

UNIVERSIDAD AUTONOMA DE MANIZALES

MAESTRIA EN GESTIÓN Y DESARROLLO EN PROYECTOS DE SOFTWARE

MANIZALES

2015

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

Los ataques Distribuidos de Denegación de Servicios (DDOS) afectan la disponibilidad de los servicios WEB por un periodo de tiempo indeterminado, inundando con peticiones fraudulentas los servidores de las empresas y denegando las solicitudes de los usuarios legítimos, generando pérdidas económicas por indisponibilidad de los servicios prestados. Por este motivo, el alcance de este documento es desarrollar un prototipo de detección de ataques DDOS a partir de máquinas de aprendizaje (SVM[2]),el cual captura el tráfico de red, filtra las cabeceras HTTP[3], normaliza los datos teniendo como base las variables operacionales: Tasa de Falsos Positivos, Tasa de Falsos Negativos, Tasa de Clasificación, y envía la información a la SVM para el respectivo entrenamiento y pruebas de detección, integrado con el software estadístico para minería de datos WEKA[4], permitiendo identificar efectivamente estos comportamientos anómalos en la capa superior a la sesión (Modelo de referencia OSI[5]), con el propósito de aumentar el tiempo de disponibilidad de los servicios. El experimento permitirá evaluar, validar y comparar la técnica del prototipo basado en un modelo supervisado SVM, contra un modelo tradicional basado en reglas como SNORT(Snort, 2008).

[2] Support Vector Machine
[3] Hypertext Transfer Protocol
[4] Waikato Environment for Knowledge Analysis
[5] Open System Interconnection

INTRODUCCIÓN

En un mundo tecnológicamente globalizado y conectado a la grande red que es Internet, las empresas están tomando muy en serio la disponibilidad de sus sistemas de información, teniendo en cuenta que en muchos casos los objetivos estratégicos de la compañía y su estabilidad financiera dependen directamente de los servicios ofrecidos a los clientes.

Los ataques DDOS son un problema crítico para las compañías que han venido integrando sus áreas tecnológicas a la gran autopista de la información, permitiendo que múltiples atacantes puedan acceder a los datos o dejar sin servicios a grandes compañías o países, como es el caso de Corea del Norte, donde se produjo un ataque DDOS el día 15 de Enero de 2015, haciendo que las comunicaciones a través de Internet quedarán por fuera de línea.(Cio, 2015)

En el presente trabajo de tesis se pretende implementar un prototipo computacional basado en modelos supervisados con técnicas de máquinas de aprendizaje SVM, que permitan detectar estos tipos de comportamientos anómalos a través del entrenamiento y respectiva evaluación del modelo generado. La metodología aplicada combinará 2 modelos, un modelo para la gestión de conocimiento KDD[6] e ingeniería de software, con el objetivo de tener un proceso iterativo en la captura, filtro, normalización, implementación y pruebas del artefacto computacional. En una primera etapa se realizará todo el proceso de captura del tráfico de la red, en la segunda etapa un filtro de las cabeceras HTTP teniendo como base la información descargada, en la tercera etapa el proceso de normalización donde se seleccionan las variables operacionales que serán utilizadas, como cuarta etapa un proceso de integración y entrenamiento del prototipo con un porcentaje del tráfico normalizado, y como último paso, la respectiva evaluación del prototipo en la clasificación de los registros anómalos o normales.

El resultado de esta investigación permitirá evaluar la efectividad de la técnica SVM comparada con la técnica tradicional basada en reglas y dejará una base técnica – funcional en el prototipo computacional, el cual detectará estos comportamientos de una manera automática y gestionada.

El documento está dividido de la siguiente manera: El Referente Contextual muestra el dominio del problema, la importancia del proyecto, revisión de algunas aplicaciones o modelos planteados

[6]Knowledge Discovery Databases

sobre máquinas de aprendizaje y la finalidad del proyecto (Numeral 1). La Estrategia Metodológica ofrece las etapas que se ejecutarán durante el desarrollo del prototipo y el valor de dicha ejecución (Numeral 2). El Desarrollo del proyecto, se enfocará en el marco teórico, las variables seleccionadas para el entrenamiento, evaluación y las etapas de ingeniería de software aplicadas al desarrollo del prototipo, así como el análisis de resultados del mismo (Numeral 3). Por último están las Conclusiones, Recomendaciones y Referencias bibliográficas (Numeral 4, 5, 6).

1. REFERENTE CONTEXTUAL

1.1. ÁREA PROBLEMÁTICA

La incursión de nuevas tecnologías nos ayuda a simplificar y presentar a los usuarios procesos flexibles y de fácil gestión; sin embargo, el riesgo de ataques que afecten la disponibilidad de los servicios es una realidad que diariamente se incrementa, debido a la conectividad a través de Internet y a la falta de previsión por parte de las compañías desarrolladoras.(Chan, Ng, Yeung, & Tsang, 2004, pág. 4252)

En la actualidad existen varias técnicas y modelos que buscan detectar y evitar ataques de Denegación de Servicios, a través de firmas, reglas o sistemas expertos. Las firmas buscan patrones exactos de un ataque, pero carecen de inteligencia a la hora de analizar un cambio en el comportamiento; las reglas definen parámetros fijos y carecen del dinamismo para la adaptación de los ataques; y los sistemas expertos requieren de un entrenamiento previo, haciendo que el desafío de evitar falsos positivos y aumentar la cantidad de ataques reales no se vea reflejado con la rapidez requerida. Algunas empresas buscan aumentar la disponibilidad a través de una infraestructura de hardware con escalamiento horizontal y vertical, haciendo que los costos operativos aumenten drásticamente y únicamente compañías con un gran musculo financiero puedan acceder a este tipo de soluciones. Los ataques distribuidos de Denegación de Servicios son tan críticos, que el 25 de Octubre de 2010 en un país del sudeste asiático, el servicio de Internet colapso, debido a un ataque masivo a los proveedores de Internet, donde se estima que la tasa de envío de paquetes oscilaba entre 10-15 Gbps[7], superando la capacidad de los enlaces terrestres y satelitales; por tal motivo, la métrica de detección de los mecanismos existentes, varía según la técnica y el modelo planteado por el autor.(Security, 2010)

El problema radica en que no existe un artefacto computacional, que permita detectar con eficacia y eficiencia los ataques Distribuidos de Denegación de Servicios en entornos corporativos, haciendo que la disponibilidad de los servicios ofrecidos sea comprometida, y por ende, las operaciones y peticiones de los usuarios legítimos sufran una degradación que se ve reflejada en términos de tiempo, calidad y finanzas.

[7] Giga bit por Segundo

1.2. ANTECEDENTES

El aumento de la conectividad de los equipos electrónicos a Internet, permitió que los ataques de Denegación de Servicios comenzaran a aumentar drásticamente a partir del año 2000, donde grandes compañías como Ebay, Yahoo, Amazon, Buy.com, E*Trade y CNN, fueron víctimas de este tipo de ataques, causando en un par de horas pérdidas financieras millonarias, debido a la falta de disponibilidad de los servicios. El impacto de este tipo de ataques a nivel mundial fue tan grande, que el Departamento de Defensa de los Estados Unidos (DARPA[8]) y el Centro Nacional de Seguridad Informática incluyeron las medidas y controles necesarios para proteger la infraestructura contra ataques de Denegación de Servicios, destacando que este problema afecta la seguridad de la información y debe ser manejando con una prioridad alta. (Chan et al. 2004, pág. 4252)

Desde el año 1998 y 1999, DARPA ha recogido y distribuido el primer estándar para evaluación de los sistemas de detección de intrusiones de redes informáticas. Se coordinaron las primeras evaluaciones formales, repetibles y estadísticamente significativa de los sistemas de detección de intrusos. Estas evaluaciones miden probabilidad de detección y falsa alarma para cada sistema sometido a prueba. (DARPA & AFRL/SNHS)

"Estas evaluaciones han contribuido de manera significativa al campo de la investigación de detección de intrusiones, proporcionando dirección a los esfuerzos investigativos y ajustando el estado actual de la técnica. Es de interés para todos los investigadores que trabajan en el problema general de detección de intrusiones en la red para estaciones de trabajo. La evaluación está diseñada para ser simple, para centrarse en las cuestiones fundamentales de la tecnología, y fomentar la participación más amplia posible mediante la eliminación de problemas de seguridad y privacidad, y proporciona tipos de datos que se usan comúnmente por la mayoría de los sistemas de detección de intrusos". (DARPA & AFRL/SNHS)

Los tipos de ataques que se evaluaron se dividieron en cuatro categorías(Mukkamala & Sung, 2003, pág. 1232):

[8] Defense Advanced Research Projects Agency

1. DOS: Denegación de Servicio.

2. R2L: Acceso no autorizado desde una máquina remota.

3. U2Su: Acceso no autorizado desde un usuario local a un usuario con privilegios de súper usuario (root).

4. Probing: Supervisión y otras pruebas.

El resultado de la evaluación permitió identificar nuevas técnicas de ataque con datos de prueba y disminuir la tasa de detección de falsas alarmas con datos reales. Los resultados de la evaluación sugirieron que la investigación futura debería estar orientada a desarrollar nuevos algoritmos que detecten nuevos ataques y no en la creación de reglas o firmas estáticas.

A partir de ese instante, diversos expertos comenzaron a trabajar en técnicas y modelos que permitieran resolver este inconveniente. A continuación se presentan algunas técnicas propuestas en los sistemas de detección de intrusos (IDS):

Rules: Se encarga de analizar el tráfico que pasa por el IDS[9], clasificando las tramas como normales o posibles intrusiones. Esta técnica utiliza una base de datos de conocimiento donde un conjunto de reglas es aplicado para comparar los patrones del tráfico con las reglas parametrizadas en la base de datos. (Chan et al. 2004, pág. 4253)

Según el modelo propuesto por (Liu W.-T. , 2008), sobre un sistema de detección de intrusos basado en agentes con un motor de reglas basado en XML[10], se puede argumentar:

Se plantea el modelo como un sistema distribuido de detección de intrusos, el cual consiste en tres agentes inteligentes que tienen funciones específicas e intercambian información XML a través de protocolos seguros cómo SSL[11] y una conexión punto a punto cómo IAP[12]. La decisión de hacerlo distribuido y con agentes inteligentes, se basa en que los sistemas de detección de

[9] Intrusion Detection System
[10] Extensible Markup Language
[11] Security Socket Layer
[12] Intrusion Alert Protocol

intrusos distribuidos tradicionales, presentan inconvenientes, tales como análisis de la jerarquía, refinamiento de datos, módulos voluminosos en todos los niveles y la interacción pasiva. El sistema de detección de intrusiones distribuida ofrece funciones y acciones más eficaces que los IDS individuales mediante el uso de agentes inteligentes. (Liu W.-T. , 2008, pág. 1401)

Los tres agentes inteligentes que se plantean en el modelo son: agente de tipo A, B y C. Cada agente tiene una tarea en el sistema que se correlaciona con los datos XML.

El agente de tipo A tiene la capacidad de monitorear los datos de la red y terminar el análisis de los protocolos de segmento. El protocolo de datos se almacena y describe a través de un XML por el agente A. El agente de tipo B es el más importante, debido a que tiene la responsabilidad de la detección de intrusiones utilizando las reglas descritas en XML. El agente de tipo C se encarga de las funciones de control y gestión. La función de un agente puede pertenecer a la seguridad del equipo donde se encuentre o a la red a la que está conectado.(Liu W.-T. , 2008, pág. 1401)

La comunicación e intercambio de información de los agentes se realiza a través de cuatro tipos de archivos XML: El protocolo XML de datos, información de control, las reglas de detección de intrusos y las respuestas obtenidas. En la transmisión se utiliza IAP y su tarea principal es la transferencia de la información entre todos los agentes en el sistema de intrusión distribuido. IAP intenta promover la interoperabilidad ubicua de todos los tipos de agentes de detección de intrusión en el entorno de Internet, y es importante para el sistema de detección de intrusiones distribuida. La seguridad de los datos XML de transporte es proporcionado por el protocolo de enlace SSL.

Un marco para la detección de intrusiones distribuida con agentes, permite obtener más independencia y elimina el análisis centralizado y tener un intercambio sin pérdidas de datos complejos entre agentes.(Liu W.-T. , 2008, pág. 1403)

Según el modelo propuesto por (Muthuregunathan, Siddharth, Srivathsan, & Rajesh, 2009, pág. 336), sobre un sistema de detección de intrusos basado en reglas utilizando la computación evolutiva, se puede argumentar:

Los autores definieron un modelo basado en técnicas de minerías de datos como Clusterización y Computación evolutiva, donde se utilizaron algoritmos genéticos para optimizar los agrupamientos (Clusters) e incrementar la eficiencia de la solución. Se planteó un diseño de arquitectura, cuyo objetivo principal es analizar el tráfico y utilizar la capacidad de procesamiento paralelo que ofrece el sistema distribuido y heterogéneo GRID. Los módulos planteados son los siguientes (Muthuregunathan et al. 2009, pág. 337):

1) Analizador de Tráfico de Red: Encargado de analizar los paquetes de red y generar un archivo de tráfico.

2) Planificador: Se encarga de recibir el archivo de tráfico y asignar una tarea de procesamiento al entorno distribuido.

3) Entorno distribuido y paralelo: Recibe la tarea del planificador y el archivo de tráfico, y ejecuta el proceso en paralelo, el cual generará un nuevo archivo de reglas y lo enviará nuevamente al analizador de tráfico, con el objetivo de actualizar las reglas del sistema de detección de intrusos.

Resultados del Experimento

Se realizaron un conjunto de experimentos donde se examinaron las capacidades del enfoque propuesto, los resultados fueron (Muthuregunathan et al. 2009, pág. 340):

Los algoritmos paralelos descritos fueron escritos utilizando librerías MPI en C++. Un recurso de clúster con un nodo maestro y cuatro nodos esclavos fueron utilizados para el propósito de la prueba. El conjunto de datos KDD 99 para la detección de intrusiones, se utilizó para generar paquetes y probar el nodo destino instalado con el sistema de detección de intrusos SNORT. Las características de la información de red capturados por el analizador de paquetes hicieron parte del conjunto de datos KDD 99. Algunas características fueron: Duración de la conexión, protocolo, servicio, número de bytes origen, entre otras.

El experimento se realizó para probar la eficacia de las reglas generadas para SNORT NIDS. El conjunto de datos se dividió en diez series, donde cada conjunto constaba de aproximadamente

500 mil registros. Herramientas para la construcción de paquetes como: Jpcap, Hping y ANTs, se utilizaron para generar los paquetes de acuerdo con el conjunto de datos KDD. La tasa de falsos positivos promedio fue de 0,43% y la tasa de verdaderos positivos fue de 64.02%.

Se demostró que la hibridación de la computación evolutiva utilizando algoritmos genéticos y computación en paralelo, podría producir nuevos resultados, aprovechando el poder de la computación proporcionada por las redes heterogéneas.

Neural Networks: Las redes neuronales artificiales (ANN[13]) se inspiran en el comportamiento de las neuronas del mundo biológico, buscando emularlas a nivel tecnológico.(Kartalopoulos, 1996, pág. 61)

Según el modelo propuesto por(Seufert & O' Brien, 2007), sobre un sistema automático de defensa contra ataques distribuidos de Denegación de servicios, se puede argumentar:

Los autores proponen un modelo de defensa automático que evita la interacción humana, basado en técnicas de redes neuronales artificiales y cuyas características principales son(Seufert & O' Brien, 2007, pág. 1219):

1) Enfoque de aprendizaje totalmente automatizado, sin intervención humana.

2) Monitoreo de la utilización de recursos en lugar del tráfico entrante en la detección de ataques.

3) Uso de redes neuronales artificiales para la detección de anomalías (otros algoritmos de aprendizaje automático se pueden integrar con facilidad).

4) Extensible, recopilación de datos distribuida y sistema de filtrado.

5) No se requiere conocimiento experto excepto la información básica sobre estructuras de protocolo.

[13] Artificial Neural Networks

Se planteó un diseño de arquitectura donde se puedan adaptar fácilmente los protocolos en cada capa del modelo de referencia OSI y los algoritmos de las máquinas de aprendizaje. Los módulos son los siguientes(Seufert & O' Brien, 2007, pág. 1219):

- Detección de ataques: Se utilizan agentes distribuidos donde se reportan los niveles de utilización de los recursos de la máquina a los maestros en intervalos de tiempo, comparando los umbrales configurados por el administrador. El concepto que se plantea no es reaccionar ante el primer ataque, sino cuando el sistema tenga un porcentaje de sobrecarga considerable.

- Extracción de las características: Se utiliza la métrica de las cabeceras de los paquetes IP (OSI Nivel 3), encabezados TCP (OSI Nivel 4) y los datos de la capa de aplicación, tales como peticiones HTTP (nivel OSI 7). Esto proporciona un espectro mucho más amplio de datos para analizar y por lo tanto una mayor probabilidad de encontrar una característica que se puede utilizar para la clasificación exitosa del tráfico.

- Fase de entrenamiento: Una vez que un ataque se ha detectado, es necesario clasificar las solicitudes entrantes como normales o anormales. Durante esta etapa, las técnicas de aprendizaje automáticas se utilizan para entrenar de forma dinámica los algoritmos, con el objetivo de aprender las diferencias entre el funcionamiento normal y las recibidas durante el ataque. Sólo la mitad de los datos se utilizan para entrenar el algoritmo. La otra mitad se utiliza para evaluar el resultado de la formación. Si la formación tiene éxito (es decir, el algoritmo puede discriminar entre los dos conjuntos), el algoritmo se puede utilizar para clasificar las peticiones entrantes y para filtrar el tráfico malicioso; de lo contrario, se supone que el algoritmo no es capaz de detectar el ataque usando este conjunto de características específicas y no se utiliza para el filtrado.

- Filtro del tráfico: El clasificador se puede utilizar directamente para filtrar el tráfico o puede ser utilizado para generar reglas para el filtrado de un marco existente para bloquear el tráfico de fuentes que han enviado recientemente solicitudes maliciosas.

Se implementó un clasificador basado en redes neuronales artificiales (ANN). Las ANN constituyen un método general y práctico para aprender las funciones reales y discretas valoradas a partir de ejemplos. Esto los hace una buena opción para clasificar el tráfico en la red, mientras que están bajo el ataque. La robustez a errores en los datos de entrenamiento es especialmente importante porque las muestras recogidas durante la fase de ataque contendrán peticiones legítimas. Del mismo modo el tráfico de la línea de base puede contener peticiones maliciosos de ataques anteriores.(Seufert & O' Brien, 2007, pág. 1220)

Resultados del modelo propuesto

Se realizaron un conjunto de experimentos donde se examinaron las capacidades del enfoque propuesto, los resultados fueron(Seufert & O' Brien, 2007, pág. 1221):

A. Rendimiento de la Clasificación: Los resultados mostraron que el algoritmo de clasificación puede funcionar eficazmente con presencia de información errada. Como era de esperar la capacidad de clasificación se reduce ligeramente y se aceptan algunos paquetes maliciosos, mientras que se eliminan algunos paquetes benignos. Los errores permanecen por debajo del 10% del sistema, lo cual sigue siendo eficaz.

B. Resultados del ataque simulado: El sistema sin protección muestra los tiempos de respuesta alta. El sistema protegido con el enfoque propuesto, regresa a los tiempos de respuesta normales después de que el ataque haya sido detectado y las contramedidas se han aplicado.

Según el modelo propuesto por (Li, Liu, & Gu, 2010, pág. 196), sobre un sistema de detección de ataques distribuidos de Denegación de servicios basado en redes neuronales, se puede argumentar:

Los autores proponen un modelo de detección basado en un método supervisado de redes neuronales llamado: Learning Vector Quantisation (LVQ), el cual utiliza patrones de salida conocidos para cada patrón de entrada, y puede ser aplicado para el reconocimiento de patrones, la clasificación multi-clase y tareas de compresión de datos.

La implementación del sistema se dividió en 5 fases tales como: recolección del conjunto de datos, pre procesamiento, determinación de la arquitectura de la red neuronal, entrenamiento y pruebas (Li et al. 2010, pág. 197):

1. Recolección del conjunto de datos: El experimento utilizó las herramientas requeridas en la topología de red. Se utilizó 1 servidor web, 4 clientes, plataformas para Apache, Mysql y PHP. Se utilizó Wast como software de prueba de esfuerzo para simular el tráfico normal. Debido a que varias clases de servidores tienen una función de respuesta diferente cuando fueron atacados, se recogieron los parámetros del servidor: 1) Uso de CPU, 2) Uso de la memoria, 4) Hora del sistema; 5) Conexiones TCP; 6) Envío bytes; 7) Recepción de bytes; 8) Conexiones TCP pasiva. Los datos se recogieron de un archivo CSV o directamente en la base de datos relacional. El experimento recogió un total de cuatro tipos de datos: 1) Normal (Sin ataques); 2) Land Attack; 3) Ping of Death Attack; 4) Smurf Attack.

2. Pre Procesamiento del conjunto de datos: Este conjunto de datos se compone de características numéricas, de modo que se puede dar como entradas a la red neuronal. Se normalizó la información reemplazando de manera numérica el significado del conjunto de datos, por ejemplo: Se reemplazó la coma, por punto y coma; y se categorizo normal con 0 y ataques con 1.

3. Determinación de la arquitectura de la red neuronal: Hay sólo dos tipos de resultados de detección de intrusos, la primera categoría cómo tráfico normal y la segunda como ataque. La capa de entrada LVQ tenía inicialmente 8 vectores dimensionales, después de las pruebas, las neuronas de la capa aumentaron a 20 para un mejor desempeño.

4. Entrenamiento: Los datos recogidos (normal 2700, ataque 2700), después del pre procesamiento se selecciona al azar el 70% como el conjunto de datos del entrenamiento, el otro 30% como prueba.

5. Pruebas: Después del entrenamiento, el 30% del conjunto de datos en la red neuronal entrenada se prueba y se obtienen los resultados.

Resultados de la comparación del modelo propuesto LVQ vs el modelo BP (Li et al. 2010, pág. 198)*:*

Con el fin de mejorar la autenticidad de los resultados experimentales, se realizó el mismo experimento 10 veces y se realizó una comparación entre la red neuronal Back Propagation (BP) y LVQ. A través de los resultados del experimento, podemos ver la utilización de redes neuronales para el sistema de detección de intrusos basado en la detección de anomalías, obteniendo una buena tasa de reconocimiento para detección de ataques. La red neuronal BP puede alcanzar el índice de reconocimiento del 89,9%, pero podemos ver fácilmente caer en mínimo local a través de resultados. La red neuronal LVQ puede alcanzar el índice de reconocimiento de 99,732%, una tasa muy alta de reconocimiento y estabilidad. Como conclusión, se puede argumentar que a través del método LVQ se puede mejorar la tasa de reconocimiento del sistema de detección de intrusos.

Support Vector Machine (SVM): Es una técnica basada en máquinas de aprendizaje, donde se clasifican los datos mediante la determinación de un grupo de vectores de soporte y se describen las características que se van a cuantificar.(Mukkamala & Sung, 2003)

Según el modelo propuesto por (Chan et al. 2004, pág. 4253), sobre un sistema Hibrido de Detección de Intrusos (HIDS), tomando como base las máquinas de aprendizaje y específicamente la técnica SVM, se puede argumentar: Consta de dos componentes: Módulo Regla-Refinamiento (RRM) y el módulo de detección basada en reglas (RBDM).

Módulo Regla-Refinamiento (RRM): En este módulo, el SVM es utilizado para clasificar los patrones de los ataques DOS y no DOS con los respectivos umbrales. Primero se obtiene un conjunto de datos aleatorios donde se divide en dos grupos con igual de condiciones en entrenamiento y pruebas. Cada característica del conjunto de entrenamiento determinado se realiza de manera individual a través del SVM y se registra con la relevancia de la métrica obtenida. El conjunto de características con una precisión superior a un umbral, se selecciona como una característica pertinente.

Los diferentes valores, desde el mínimo al valor máximo de esta función de entrada, se alimentan a la SVM y los umbrales son encontrados si la clasificación de la SVM cambia con respecto a los valores introducidos. Este proceso se repite para cada una de las características pertinentes. Después que el SVM ha encontrado las características pertinentes y los umbrales correspondientes, las normas se definen y almacenan en el módulo de detección basada en reglas (RBDM).

Módulo de detección basada en reglas (RBDM): Utiliza las reglas generadas a partir del módulo (RRM) para detectar ataques de denegación. El RBDM sólo supervisa las variables seleccionadas por el mecanismo de refinamiento (RMM), lo que ahorra los recursos y permite el monitoreo de la red con una mayor utilización.

En base a las reglas definidas, si una característica en el tráfico alcanza los umbrales preseleccionados, se envía un mensaje de alerta al administrador para que ejecute las acciones pertinentes.

Variables que se midieron:

Se analizaron las siguientes variables (Chan et al. 2004, pág. 4254):

- Cantidad de conexiones del mismo equipo.
- Número de datos enviados desde el origen al destino.
- Número de datos enviados desde el destino al origen.
- Tipo de servicio utilizado (HTTP).

Resultados de la comparación del modelo propuesto HIDS vs el modelo basado en reglas SNORT (Chan et al. 2004, pág. 4255)**:**

El modelo HIDS tuvo una mejoría significativa (aproximadamente un 20% más) en la precisión de la clasificación cuando se comparó con el SNORT utilizando las reglas establecidas por los expertos de dominio.

Por otra parte, HIDS fue más rápido en la detección de ataques y utilizó menos recursos para controlar la red, debido a la reducción del número de funciones para la definición de los ataques. (Chan et al. 2004, pág. 4255)

Según el modelo propuesto (Subbulakshmi, Shalinie, GanapathiSubramanian, BalaKrishnan, AnandKumar, & Kannathal, 2011, pág. 17), sobre un sistema de detección de ataques DDOS, utilizando un enfoque mejorado de la técnica SVM y un conjunto de datos generado en tiempo real, se puede argumentar:

Los autores proponen un nuevo tipo de SVM denominado EMCSVM (Enhanced Multi Class Support Vector Machines), el cual utiliza pesos adicionales para los registros del conjunto de datos y para la detección de los ataques DDOS en varias clases. Se aplicó una metodología para la generación del conjunto de datos real para 10 ataques DDOS, analizando 14 atributos derivados del paquete de datos obtenido. Los ataques se generan en el entorno distribuido utilizando un grupo de pruebas experimental, después los ataques se recogen en el servidor y los datos DDOS se crean utilizando scripts. EMCSVM se utiliza para clasificar el tráfico de ataques y normal, utilizando el entrenamiento y las pruebas (Subbulakshmi et al. 2011, pág. 18). A continuación se presentan los pasos para la generación del conjunto de datos (Subbulakshmi et al. 2011, pág. 19):

1) Singularidad del Enfoque:

El conjunto de datos DDOS utilizado se concentró tanto en la capa de aplicación y la capa de red. Dado que la mayoría de los parámetros se derivan de los paquetes de red durante el tiempo de ataque, la inferencia de diversos ataques puede distinguirse claramente. El tráfico normal se actualiza en intervalos regulares para hacer que el perfil del usuario pueda tener más influencia en la detección.

2) Colección Normal de Datos:

El comportamiento de usuario normal es diferente de la atacante, por ese motivo los parámetros recogidos para el usuario normal y atacante muestran variaciones distintas. El comportamiento de los usuarios normales es lineal y regular, mientras que el comportamiento atacante es fluctuante y completamente irregular. Los parámetros como la tasa de solicitud HTTP, tasa de Sesión, el tiempo pasado en la página, el número de paquetes TCP, el número de paquetes UDP, el número de paquetes ICMP y el protocolo se derivan del tráfico recogido .

3) Generación del Ataque:

Los ataques se generan con una base de datos de pruebas experimental utilizando scripts de ataque. Para generar el ataque, el usuario debe especificar el número de atacantes y la dirección IP de la víctima y la duración de los ataques. Los tipos de ataques experimentales son ICMP Flooding, UDP Flooding, TCP Flodding, SmurfFlooding, escaneo de puertos y HTTP Flooding. Cada tipo de ataque masivo es generado en intervalos de tiempos distintos y recogido a través de un archivo.

4) Recopilación de los Datos del Tráfico:

La recolección de datos se realiza a través de una base de datos que contiene los aplicativos conectados a la red. Los datos normales y de ataque se recogen en el mismo entorno. El tráfico de datos se recoge mediante los programas de captura de paquetes.

5) Cálculo de los Parámetros:

Los archivos son separados cada treinta minutos y 14 parámetros se calculan utilizando el analizador de paquetes de red. Se clasifica el tráfico en 10 tipos de clases, como Normal, ICMP, TCP, UDP, SMURF, escaneo de puertos, LAND, HTTP, IP y sesión.

6) Conjunto de Datos Final:

El conjunto de datos final contiene diez tipos de clases, tales como: Normal, ICMP, TCP, UDP, SMURF, escaneo de puertos, LAND, HTTP, IP y sesión. El entrenamiento y las pruebas se recogen en archivos separados. Cada archivo contiene los 14 atributos y un número diferente de

registros. Los archivos de entrenamiento se utilizan para entrenar los algoritmos de aprendizaje automático y los archivos de prueba para evaluar el rendimiento.

Después de obtener el conjunto de datos normalizado con los tipos de ataque, se procede a clasificarlo a través del EMCSVM. Durante el entrenamiento, EMCSVM aprende los patrones normales y de ataque. En las pruebas, aprende a diferenciar los ataques y el tráfico normal usando los patrones aprendidos (Subbulakshmi et al. 2011, pág. 20).

Como conclusión, se puede argumentar que el EMCSVM utilizado añade valor en la mejora de la precisión en la detección de varios ataques en comparación con la SVM ordinaria. (Subbulakshmi et al. 2011, pág. 21)

1.3. JUSTIFICACIÓN

La conectividad a través de redes privadas o públicas nos exige que la confidencialidad, integridad y disponibilidad de la información, sean una prioridad en el momento de desarrollar aplicativos informáticos, ya que en un entorno globalizado como el que estamos viviendo, una de las diferencias competitivas se brinda a través de los tiempos de respuesta y disponibilidad de los servicios ofrecidos. La seguridad de la información es un tema relevante, debido a la cantidad de equipos de cómputo que cada día se unen a la gran red mundial que es Internet, haciendo que la disponibilidad de los servicios sea un elemento de máxima prioridad, procurando el acceso de los mismos de manera permanente.

Los sistemas de información constituyen la mejor herramienta para fomentar el desarrollo social, económico y administrativo de la sociedad, debido al impacto que generan en el mejoramiento de la calidad de vida y eficacia en la ejecución de las tareas cotidianas; por este motivo, es necesario que unos de los pilares de la seguridad de la información como la disponibilidad, no se vea comprometida debido a las peticiones fraudulentas que diariamente los atacantes envían con el fin de degradar el servicio prestado por las corporaciones, haciendo que el usuario final sufra directamente las consecuencias de estos actos. Por tal motivo, es necesario el desarrollo de nuevas técnicas y prototipos que permitan detectar de manera eficaz y eficiente los ataques fraudulentos de peticiones concurrentes, con el fin de evitar la indisponibilidad del servicio y las pérdidas económicas, como las que se presentaron en Febrero del año 2000, donde grandes compañías vieron comprometidas sus servicios, generando pérdidas en tiempo y dinero. (Chan et al. 2004, pág. 4252)

La implementación de este prototipo computacional ayudará a complementar un nuevo enfoque en la búsqueda de la solución a los ataques de denegación de servicios, debido a que el diseño de las variables a medir, estará definido en la capa superior a la sesión (Modelo de referencia OSI) y se generarán reportes de detección periódicos, permitiendo complementar los esfuerzos realizados por la comunidad investigativa en la capa de red, buscando aumentar la eficacia en la detección de peticiones fraudulentas.

A través del prototipo se aporta conocimiento en la búsqueda de un modelo que permita evitar la cantidad de falsos positivos y aumentar la cantidad de ataques reales detectados, con el fin de minimizar esta anomalía que diariamente causa tantos inconvenientes a nivel local, regional y mundial.

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL

Implementar un prototipo computacional que detecte ataques de Denegación de Servicios a partir de técnicas basadas en máquinas de aprendizaje, con el propósito de minimizar este tipo de comportamientos anómalos en entornos corporativos.

1.4.1.1. OBJETIVOS ESPECIFICOS

- Definir la técnica de Máquina de Aprendizaje y los parámetros que se aplicarán en el prototipo.
- Diseñar e implementar el modelo de arquitectura de software a partir de la técnica, parámetros y necesidades del problema identificado.
- Diseñar e implementar el proceso de entrenamiento y afinamiento de la máquina de aprendizaje, y el prototipo desarrollado a partir del conjunto de datos normalizado.
- Evaluar y validar el modelo a partir de un conjunto de métricas medibles en el problema de detección de intrusos, entre otras: Tasa de Falsos Positivos, Tasa de Falsos Negativos, Tasa de Clasificación, curvas ROC[14].

[14] Receiver Operating Characteristic

1.5. RESULTADO ESPERADO

Se realizarán 2 entregables: El prototipo computacional que detecte ataques DDOS y un cuadro comparativo entre la técnica SVM y el modelo tradicional utilizado por SNORT:

- Prototipo computacional que detecte los comportamientos anómalos de los ataques DDOS, con todas las fases desarrolladas en la metodología: Diseño, Implementación, Entrenamiento, Evaluación y Pruebas.

- Cuadro comparativo entre la técnica tradicional basada en reglas y la técnica basada en máquinas de vectores SVM, utilizando como comparación las variables operacionales: TFP (Tasa de falsos positivos), TFN (Tasa de falsos negativos), TVN (Tasa de verdaderos negativos), TVP (Tasa de verdaderos positivos), Porcentaje de Clasificación, Curvas ROC.

2. ESTRATEGIA METODOLÓGICA

2.1. METODOLOGIA

La metodología propuesta estará orientada al descubrimiento del conocimiento de la información (KDD) y al análisis, diseño e implementación del prototipo. Por lo tanto, para la explotación del conocimiento se tomará como referencia el marco de trabajo propuesto por (Roy, 2001, pág. 180):

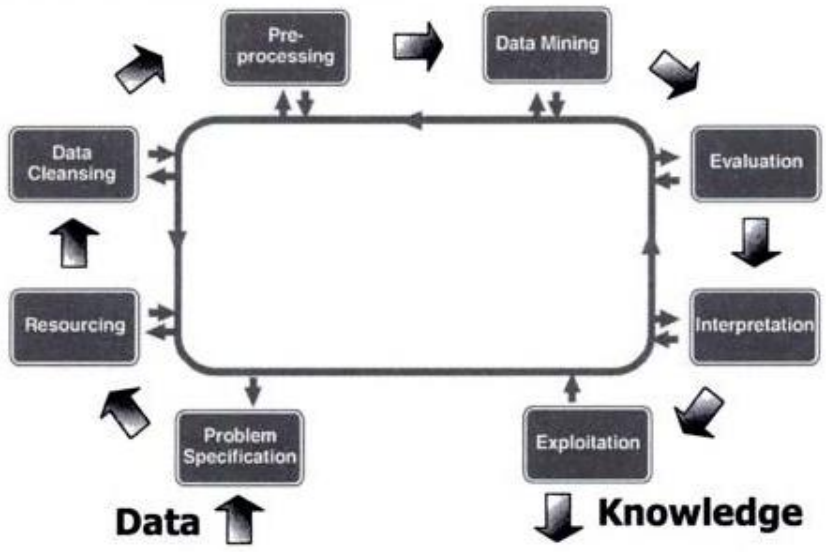


Figura 1: Marco de trabajo. (Roy, 2001, pág. 181)

Los pasos que tiene este marco de trabajo son los siguientes(Roy, 2001, pág. 181):

1. *Especificación del problema:* Entender el dominio del problema, las tareas, objetivos y la factibilidad del problema a resolver.
2. *Recursos:* Consiste en recoger y seleccionar el conjunto de datos originales o un subconjunto apropiado, para resolver el problema identificado.
3. *Limpieza y pre procesamiento*: Tomar decisiones con respecto a valores faltantes, atípicos, erróneos. La etapa de preparación y limpieza busca normalizar los datos, con el propósito de evitar errores en los resultados reales.

4. *Técnica de Minería de Datos:* Escoger la herramienta adecuada para resolver el problema, teniendo en cuenta el objetivo (predecir, explicar, clasificar, agrupar). Establecer los parámetros de las redes utilizadas (arquitectura de la red, datos de entrenamiento, validación y pruebas).
5. *Evaluación:* Llevada a cabo por un analista con un nivel de profundidad en el área de conocimiento básica.
6. *Interpretación de los datos*: Llevada a cabo por un grupo de expertos en el dominio del problema.
7. *Explotación:* Consolidación del conocimiento adquirido, probando los modelos creados contra los resultados obtenidos de la aplicación.

Este marco de trabajo es naturalmente iterativo y permite tener una serie de pasos que nos llevan a explotar e interpretar los datos en conocimiento; sin embargo, KDD está orientado a descubrir conocimiento y no tiene implícito etapas definidas de ingeniería de software que son requeridas para el desarrollo de este proyecto.

Las fases mostradas a continuación mezclan el marco de trabajo para el descubrimiento de conocimiento aplicado a las máquinas de aprendizaje con el desarrollo del prototipo de software:

FASE 1: DISEÑO

- Definición de la técnica de Máquina de Aprendizaje.
- Definición de los parámetros.
- Diseño del prototipo computacional.

FASE 2: IMPLEMENTACIÓN

- Implementación del prototipo computacional.

FASE 3: ENTRENAMIENTO Y EVALUACIÓN

- Recolección y selección del conjunto de datos originales y subconjunto de datos para el problema que vamos a resolver.
- Limpieza y pre procesamiento de los valores de los parámetros.
- Entrenamiento de la máquina de aprendizaje con un 60% de datos normalizados.

- Evaluación del modelo a partir de los resultados obtenidos y comparación del prototipo implementando contra un sistema de detección convencional.

FASE 4: PRUEBAS

- Realización de las pruebas funcionales al prototipo.
- Realización de las pruebas funcionales al clasificador con un 40% de datos normalizados.

2.2. PRESUPUESTO

Rubros	Valor ($)
Material	$3.650.000
Viajes y Transporte	$1.000.000
Publicación	$500.000
Total	$5.150.000

Tabla 1: Presupuesto total

MATERIALES, SUMINISTROS Y BIBLIOGRAFÍA:

Materiales	Cantidad	Valor Unidad (S)	Total (S)
Servidor en la nube por 6 meses.	1	$1.000.000	$1.000.000
Servicio de energía eléctrica por 6 meses	1	$300.000	$300.000
Servicio de internet por 6 meses	1	$350.000	$350.000
Dispositivos (USB, CD, Papelería)	1	$300.000	$300.000
Libros (Seguridad de la información e Ingeniería de conocimiento)	2	$300.000	$600.000
Computador	1	$1.100.000	$1.100.000
Total			$3.650.000

Tabla 2: Presupuesto materiales y suministro

VIAJES Y TRANSPORTE:

Viaje / Transporte	Justificación	Pasajes	Estadía	Total (S)
Viajes				$500.000
Transporte				$500.000
Total				$1.000.000

Tabla 3: Presupuesto viajes y transporte

3. DESARROLLO

3.1. MARCO TEÓRICO

Existen varios modelos y técnicas que se han desarrollado con el fin de minimizar estas anomalías computacionales; sin embargo, es importante que antes de analizar las técnicas de detección, se analice la definición y los diferentes métodos de ataques que pueden afectar la disponibilidad de los servicios.

Ataque de Denegación de Servicio:

Se caracteriza por un intento explícito de un atacante para evitar que los usuarios legítimos de un servicio utilicen los recursos deseados. Algunos ejemplos de ataques de denegación incluyen (Lau, Rubin, Smith, & Trajkovic, 2000):

- Intentos de inundación de una red, evitando de este modo los intentos de tráfico de red legítimos.
- Intento de interrumpir conexiones entre dos máquinas, evitando de este modo el acceso a un servicio.
- Intentos de evitar a un individuo en particular el acceso a un servicio.
- Intentos de interrumpir el servicio a un sistema o persona específica.

Ataque Distribuido de Denegación de Servicio:

Se caracteriza por un conjunto de computadores coordinados y distribuidos geográficamente que envían miles de peticiones concurrentes a la víctima, con el objetivo de generar una indisponibilidad de los servicios prestados. A continuación se presentan los pasos que toman lugar durante un ataque distribuido (Lau et al. 2000):

1. El verdadero atacante envía un mensaje de ejecutar al programa de control principal.

2. El programa principal de control recibe el mensaje de ejecución y propaga el comando para el ataque que está bajo su control.

3. Al recibir la orden, los dispositivos conectados al programa principal comienzan el ataque a la víctima.

En este tipo de ataques distribuido, el programa principal se encarga de brindar las instrucciones a los dispositivos que se encuentran conectados a la red de ataque, haciendo que la coordinación se encuentre centralizada.

Métodos de Ataque de Denegación de Servicio:

A continuación se describen algunos métodos básicos de ataques de Denegación de Servicios (Lau et al. 2000):

- **Ataque Smurf:** Un atacante envía una gran cantidad de mensajes (ICMP[15]) a un conjunto de direcciones de protocolo de transmisión de Internet (IP[16]). La víctima es objeto de una gran cantidad de tráfico que los amplificadores generan. Este ataque tiene el potencial de sobrecargar una red completa.

- **Ataque SYN Flood:** También se conoce como el ataque al protocolo (TCP[17]) y se basa en la explotación de la negociación TCP en tres vías en la capa de transporte. En la negociación es necesario un intercambio de tres paquetes antes de que un cliente pueda utilizar oficialmente el servicio. El ataque consiste en enviar un paquete SYN inicial sin enviar el ACK correspondiente, dejando al servidor en espera y evitando el ingreso de nuevas peticiones ACK al servidor.

- **Ataque UDP Flood:** El atacante utiliza paquetes UDP[18] falsos para conectarse al servicio de escucha, donde el generador envía caracteres a otra máquina. El resultado es que los

[15] Internet Control Message Protocol
[16] Internet Protocol
[17] Transmission Control Protocol
[18] User Datagram Protocol

dos servicios consumen todo el ancho de banda de red disponible entre las máquinas ya que intercambian caracteres entre sí.

Métodos Distribuidos de Ataques de Denegación de Servicio:

A continuación se describen algunos métodos distribuidos de ataques de Denegación de Servicios (Lau et al. 2000):

- **Ataque Trinoo:** Utiliza una comunicación TCP entre el atacante y el control principal. El programa principal comunica el ataque a los demonios utilizando paquetes UDP y realiza un ataque UDP Flood.

- **Ataque TribeFlood Network (TFN):** Utiliza una interfaz de línea de comandos para comunicar el atacante y el control principal. La comunicación es realizada a través de la respuesta de paquetes ICMP. El método TFN implementa ataques como Smurf, SYN Flood, UDP Flood e ICMP Flood.

- **Ataque Stacheldraht:** Se basa en el ataque TFN, con la diferencia que la comunicación TCP entre el atacante y el control principal se encuentra encriptado. El método Stacheldraht implementa ataques como Smurf, SYN Flood, UDP Flood e ICMP Flood.

- **Ataque Shaft:** La comunicación y ataque entre el control principal y los demonios se realiza a través de paquetes UDP. El control principal y el atacante se comunica a través del protocolo TCP y una conexión telnet. Lo distintivo de este método es la capacidad para cambiar el control de los servidores maestros y los puertos en tiempo real, haciendo que la detección sea compleja.

- **Ataque TFN2K:** Utiliza los protocolos TCP, UDP e ICMP para entre el control principal y los demonios. La comunicación entre el atacante real y el control maestro está encriptado con el algoritmo CAST-256. El método TFN2K implementa ataques como Smurf, SYN Flood, UDP Flood e ICMP Flood.

Métodos de Defensa contra Ataques:

En la actualidad no existe un método que evite en su totalidad un ataque de Denegación de Servicios; sin embargo, existen varias técnicas de seguridad que hacen el trabajo de los atacantes más complejo (Lau et al. 2000):

- **Filtro en los enrutadores:** Filtrado de todos los paquetes que entran y salen, protege la red de ataques que se llevan a cabo en las redes vecinas, y evita que la propia red sea un atacante. Esta medida requiere filtros de entrada y salida de paquetes en todos los enrutadores.

- **Deshabilitar IP Broadcast:** Al deshabilitar el envío de mensajes de Broadcast, los ordenadores ya no pueden ser utilizados como amplificadores para ataques ICMP Flood y Smurf. Sin embargo, para defenderse de este ataque, todas las redes vecinas deben deshabilitar IP Broadcast.

- **Aplicar parches de seguridad:** Para protegerse de los ataques de Denegación de Servicios, los ordenadores deben estar actualizados con los últimos parches de seguridad.

- **Deshabilitar los servicios no utilizados:** En general si los servicios de red son innecesarios o no utilizados, los servicios deben ser desactivados para evitar la manipulación y ataques.

- **Medida de detección de intrusos:** La supervisión de la red es una manera buena de protección contra ataques de Denegación de servicio. Por supervisión de patrones de tráfico, una red puede determinar cuando está bajo ataque, y puede tomar los pasos necesarios para defenderse.

Máquinas de Aprendizaje:

A continuación se presentan algunas técnicas de clasificación que intentan generar aprendizaje computacional automático y son una rama de la inteligencia artificial(Vijaya, Jamuna, & Karpagavalli, 2009, pág. 402):

- **Perceptrón Multicapa** (MLP)**:** El perceptrón multicapa (MLP) de red es el más ampliamente utilizado clasificador de redes neuronales. Las redes MLP son de uso general, los modelos flexibles, no lineales que consisten en un número de unidades organizadas en múltiples capas. La complejidad de la red de MLP se puede cambiar variando el número de capas y el número de unidades en cada capa. Teniendo en cuenta las unidades ocultas y suficientes datos, se ha demostrado que las MLP puede aproximarse prácticamente a cualquier función a cualquier precisión deseada. Son herramientas valiosas en problemas cuando se tiene poco o ningún conocimiento acerca de la forma de la relación entre los vectores de entrada y sus correspondientes salidas.

- **Árbol de Decisión:** La clasificación genera la salida como un árbol binario como estructura llamada un árbol de decisión, en el que cada nodo de rama representa una elección entre un número de alternativas, y cada nodo hoja representa una clasificación o decisión. Un modelo de árbol de decisión, con reglas para predecir la variable objetivo. Este algoritmo funciona bien, incluso cuando hay un número variable de ejemplos de entrenamiento y un considerable número de atributos en bases de datos grandes. Un modelo de árbol de decisión se construye mediante el análisis de los datos de entrenamiento y el modelo se utiliza para clasificar los datos que no se alcanzan a observar.

- **Clasificador Naive Bayes:** Es un clasificador simple pero eficaz que se ha utilizado en numerosas aplicaciones de procesamiento de información incluyendo, procesamiento del lenguaje natural, la recuperación de información, entre otras. La técnica se basa en el teorema Bayesiano y es particularmente adecuado cuando la dimensionalidad de las entradas es alta. El Clasificador supone que el efecto de un valor de la variable en una clase dada es independiente de los valores de la otra variable. El inductor Naive Bayes calcula las probabilidades condicionales de las clases impartidas en la instancia y escoge la clase con probabilidad más alta. Dependiendo de la naturaleza exacta del modelo de probabilidad, los clasificadores pueden ser entrenados de manera muy eficiente en un entorno de aprendizaje supervisado.

- **Support Vector Machine (SVM):** Es un nuevo enfoque para la clasificación de patrones supervisado que se ha aplicado con éxito a una amplia gama de problemas de reconocimiento de patrones. Máquina de soporte vectorial es un algoritmo de entrenamiento para el aprendizaje de la clasificación y reglas de regresión de los datos. SVM es la más adecuada para trabajar con precisión y eficiencia con espacios que cuentan con alta dimensionalidad. SVM se basa en fundamentos matemáticos y los resultados en algoritmos simples, pero muy poderosos

Técnicas y Modelos contra Ataques:

A continuación se describen algunas técnicas propuestas contra ataques de Denegación de Servicios (Lau et al. 2000):

- **Rules:** Se encarga de analizar el tráfico que pasa por el IDS[19], clasificando las tramas como normales o posibles intrusiones. Esta técnica utiliza una base de datos de conocimiento donde un conjunto de reglas es aplicado para comparar los patrones del tráfico con las reglas parametrizadas en la base de datos. El IDS alertará al administrador de la red o jefe de seguridad sobre los patrones encontrados y un posible fallo de seguridad. El experto en el dominio define las reglas que deberán ser aplicadas en el IDS, tomando como referencia la experiencia y las observaciones en diferentes tipos de ataques. Las falencias detectadas en este tipo de técnicas es que siempre se requiere de un experto humano que constantemente actualice las reglas con los nuevos esquemas de ataque. (Chan et al. 2004, pág. 4253)

- **Support Vector Machine (SVM):** Es una técnica basada en máquinas de aprendizaje, donde se clasifican los datos mediante la determinación de un grupo de vectores de soporte y se describen las características que se van a cuantificar. SVMs proporcionan un mecanismo genérico para adaptarse a un hiper plano para llevar a cabo una clasificación lineal de los patrones a través de la utilización de una función kernel. El usuario puede proporcionar una función (por ejemplo, lineal, polinómica o sigmoide) durante el proceso

[19] Intrusion Detection System

de formación, que selecciona vectores de soporte. El número de parámetros libres utilizados en la SVMs depende del margen que separa los puntos de datos pero no sobre el número de funciones de entrada, por lo tanto SVMs no requieren una reducción en el número de características con el fin de evitar un sobre ajuste, una ventaja aparente en aplicaciones tales como detección de intrusos. Otra de las principales ventajas de este método es la baja probabilidad de error esperado de generalización.(Mukkamala & Sung, 2003)

La detección de DDOS utilizando SVM se compone de tres fases(Mukkamala & Sung, 2003):

- Pre procesamiento: Un analizador automatizado se utiliza para procesar los datos de volcado de TCP / IP sin procesar en la forma adecuada.

- Formación: SVM es entrenado en diferentes tipos de ataques y los datos normales. Tenemos 41 características y dos clases, uno es normal (-1) y el otro es DOS datos de ataque (Cl).

- Pruebas: Se analiza el rendimiento de la SVM entrenada, para asegurar que ha adquirido la capacidad adecuada de clasificación.

Herramientas de Ataque:

A continuación se describen las herramientas más importantes para realizar ataques Distribuidos de Denegación de Servicios (InfoSec, 2013):

- **LOIC (Low Orbit Ion Canon):** Es una de las herramientas más populares de ataques DOS disponibles en Internet. Se puede utilizar simplemente por un solo usuario para llevar a cabo un ataque DOS en servidores pequeños. Esta herramienta realiza un ataque DOS enviando UDP, TCP o peticiones HTTP al servidor víctima, únicamente se necesita

saber la URL de la dirección IP del servidor y la herramienta realizará el resto. Lo más importante que debes saber es que la herramienta no oculta la dirección IP.

- **XOIC:** Desarrolladores de XOIC afirman que es más poderoso que LOIC de muchas maneras. Al igual que LOIC, viene con una interfaz gráfica de usuario fácil de usar, por lo que un principiante puede utilizar fácilmente esta herramienta para realizar ataques a otros sitios web o servidores.

  En general, la herramienta viene con tres modos de ataque. El primero, conocido como el modo de prueba, es muy básico. El segundo es el modo de ataque DOS normal. El último es un modo de ataque DOS que viene con un mensaje / HTTP / UDP / ICMP TCP.

- **HULK (HTTP Unbearable Load King):** La herramienta genera una respuesta única para cada una de las solicitudes realizadas, generando tráfico ofuscado en un servidor web. Esta herramienta utiliza otras técnicas para evitar la detección de ataques a través de patrones conocidos. Utiliza la falsificación remitente y puede pasar por alto los motores de almacenamiento en caché, por lo que impacta directamente los recursos del servidor.

- **DDOSIM—Layer 7 DDOS Simulator:** Como su nombre indica, se utiliza para llevar a cabo ataques DDOS mediante la simulación de varios grupos de ataque. Todos los ordenadores crean conexiones TCP al servidor de destino. Esta herramienta está desarrollada en C++ y se ejecuta en los sistemas Linux. Estas son características principales de DDOSIM:

  - Simula varios ataques en grupo.
  - Direcciones IP aleatorias.
  - Ataques basados en conexiones TCP.
  - Ataques DDOS en la capa de aplicación.
  - Ataques DDOS HTTP con las solicitudes válidas.
  - Ataques DDOS HTTP con las solicitudes inválidas.

- Ataques DDOS SMTP.
- Inundaciones TCP con conexión en el puerto aleatorio.

- **R-U-Dead-Yet:** Es una herramienta de ataque HTTP/POST DOS. Se lleva a cabo un ataque DOS con el envío de información en un formulario WEB mediante el método POST. Esta herramienta viene con un menú en la consola interactiva. Detecta formularios de una URL determinada y permite al usuario seleccionar qué formas y campos se deben utilizar para un ataque DOS basado en el método POST.

- **Tor's Hammer:** Es una herramienta escrita en Python, la cual cuenta con una gran ventaja: Se puede ejecutar a través de una red TOR y ser anónimo mientras se realiza el ataque. Es una herramienta eficaz que puede matar a los servidores Apache o IIS en pocos segundos.

- **PyLoris:** Con esta herramienta se pueden utilizar servidores proxy SOCKS y conexiones SSL para realizar un ataque DOS en un servidor. Se puede apuntar a varios protocolos, incluyendo HTTP, FTP, SMTP, IMAP y TELNET. La última versión de la herramienta viene con una interfaz gráfica de usuario sencilla y fácil de usar. A diferencia de otras herramientas tradicionales de ataque DOS, esta herramienta golpea directamente el servicio.

3.2. FASE 1: DISEÑO

3.2.1. DEFINICIÓN DE LA TÉCNICA DE LA MÁQUINA DE APRENDIZAJE

En el campo de la investigación de reconocimiento de patrones, las máquinas de aprendizajes utilizando SVM han sido empleadas con gran éxito en diferentes herramientas de clasificación y regresión. Algunos núcleos utilizados internamente por la SVM son: lineal, polinómica, radial (RBF) y sigmoide. A continuación se presentará la representación matemática de los núcleos anteriores (NachifFernandes, Pilastri, Pereira, Goncalves Pires, & Papa, 2014, pág. 260):

1. Lineal:

$$K(x_i, x_j) = (x_i \times x_j)$$

Figura 2: Fórmula lineal

2. Polinomial:

$$K(x_i, x_j) = (\gamma x_i \times x_j + c)d$$

Figura 3: Fórmula polinomial

3. Función Básica Radial(RBF):

$$K(x_i, x_j) = e\gamma \left|\left|x_i - x_j\right|\right| \ \gamma > 0$$

Figura 4: Fórmula función básica radial RBF

4. Sigmoide:

$$K(x_i, x_j) = \tan^{-1}(\gamma x_i \times x_j + c),\ \gamma > 0\ c < 0$$

Figura 5: Fórmula sigmoide

En la que (xi) y (xj) soporte para las muestras, (d) es el grado y: (γ) es la varianza de Gauss. La selección de cada núcleo dependerá del problema que se busca solucionar, para este proyecto

utilizaremos la función básica radial (RBF), teniendo en cuentas las características presentadas por los autores (Nachif Fernandeset al.2014, pág. 260):

- Las habilidades de generalización de RBF lo convierten en la primera función del núcleo seleccionada.
- En caso de datos multi-dimensionales, RBF se comporta mejor que el núcleo lineal.
- En datos no linealmente separables, el núcleo RBF es a menudo preferible, ya que utiliza menos hiper-parámetros que el núcleo polinomial.

Es importante resaltar que un buen comportamiento del núcleo RBF, depende de los valores configurados en sus parámetros.

3.2.2. DEFINICIÓN DE LAS VARIABLES OPERACIONALES

El trabajo a realizar parte de la información recopilada en el experimento realizado en la fase de implementación, adicionalmente del interés en determinar qué factores influyen en la detección de ataques distribuidos de denegación de servicios que sean aplicados en la búsqueda de la solución definitiva; por tal motivo, las parámetros a trabajar parten de la investigación que se realice sobre estos factores. Las variables que tendrá el prototipo son las siguientes:

a) Entradas de las variables de clasificación a la SVM:
    - DURATION (Tiempo en segundos de cada petición)
    - IP_SRC (Dirección IP origen)
    - IP_DEST (Dirección IP destino)
    - IP_SRC_COUNT (Cantidad de peticiones desde la IP origen).
    - BYTES (Número de datos enviados).
    - CLASS (Clasificación del tráfico: Normal, Anómalo).
b) Salidas de las variables de clasificación de las métricas de desempeño de la SVM:
    - TFP (Tasa de falsos positivos).
    - TFN (Tasa de falsos negativos).
    - TVN (Tasa de verdaderos negativos).

- TVP (Tasa de verdaderos positivos).
- Porcentaje de Clasificación.
- Sensibilidad.

3.2.3. DISEÑO DEL PROTOTIPO DE DETECCIÓN DE ATAQUES:

El diseño estará enfocado en mostrar las diferentes capas y niveles de alto nivel que tendrá el prototipo computacional, así como los componentes que intervienen en la extracción, filtro, normalización, entrenamiento y evaluación.

El diseño del prototipo computacional expone la siguiente arquitectura lógica (Figura 6):

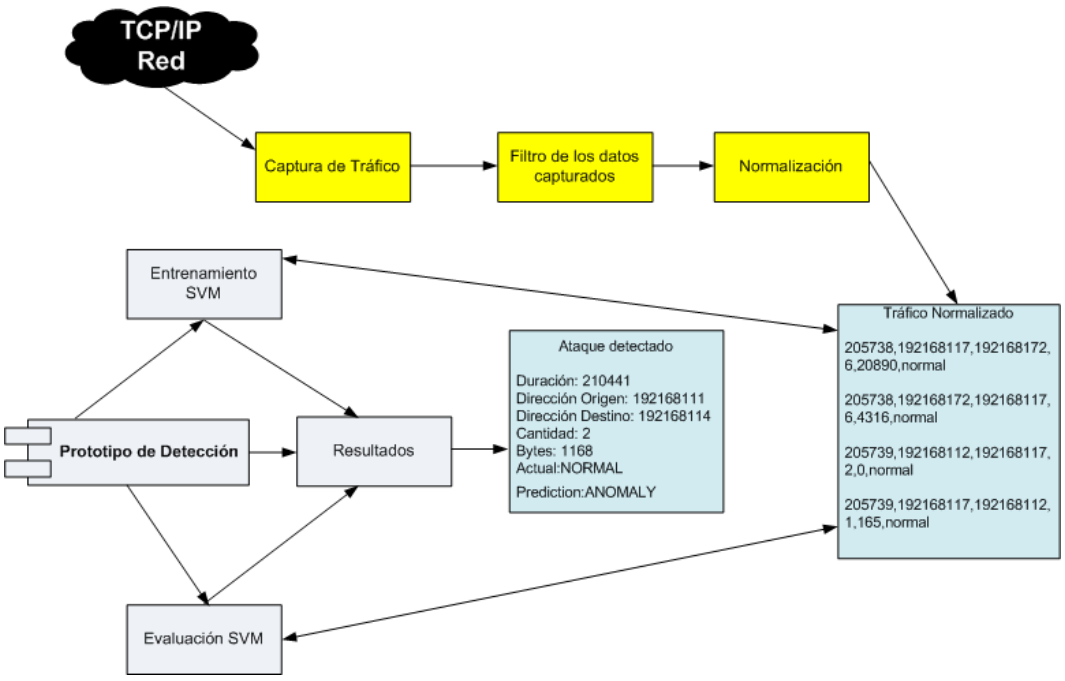


Figura 6: Arquitectura de alto nivel del Prototipo de Detección SVM

En el diagrama de arquitectura anterior se puede observar como el prototipo tiene separado los componentes de la captura, filtro y normalización de la información, con la utilización de un lenguaje que permita aplicar eficientemente las expresiones regulares requeridas para el volumen de información estimado. La evaluación y entrenamiento se realizarán en un lenguaje de alto nivel, que permita la integración de la librería SVM y maneje el concepto de multiprocesamiento de una manera eficaz y eficiente independiente del sistema operativo utilizado.

La captura, filtro y normalización de las peticiones realizadas a través de Internet, hace que el proceso tenga un nivel de complejidad alto, debido a la cantidad de información enviada y recibida durante la negociación entre el cliente y el servidor, por lo tanto, es necesario diseñar este flujo de información a través de una serie de componentes que permitan clasificar la información de una manera correcta. A continuación se presenta el diagrama de actividades del proceso de normalización del prototipo y la respectiva interacción con los diferentes módulos planteados (Figura 7):

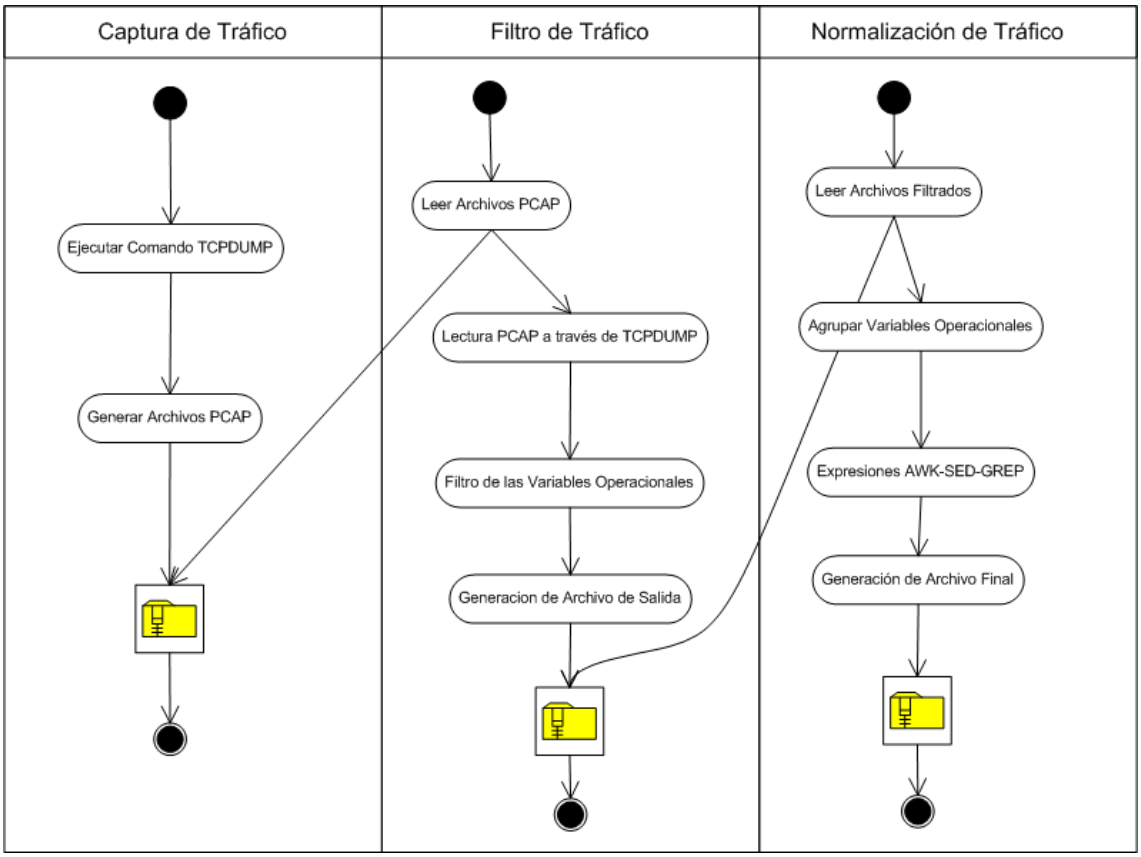


Figura 7: Arquitectura lógica de la carga, filtro y normalización

1. ***Captura del tráfico***: Esta etapa es la encargada de recuperar todo el tráfico enviado por los usuarios a través de peticiones WEB y guardarlos en una aplicación de programación utilizada para la captura de paquetes llamada PCAP.
2. ***Filtro del tráfico***: Esta etapa es la encargada de recuperar los archivos PCAP y realizar un filtro teniendo como base las variables operacionales seleccionadas en la etapa de diseño (Ver: *Definición de las variables Operacionales*).

3. ***Normalización del tráfico***: Esta etapa es la encargada de recuperar la información filtrada y realizar un proceso de agrupamiento teniendo como base los factores de tiempo definidos en las variables operacionales.

La evaluación y entrenamiento tendrán como punto de partida para el prototipo desarrollado la información normalizada, la cual permitirá la generación del modelo en la etapa de entrenamiento y la respectiva carga del mismo en la etapa de evaluación. El diagrama de flujo del componente estará representado de la siguiente manera (Figura 8):

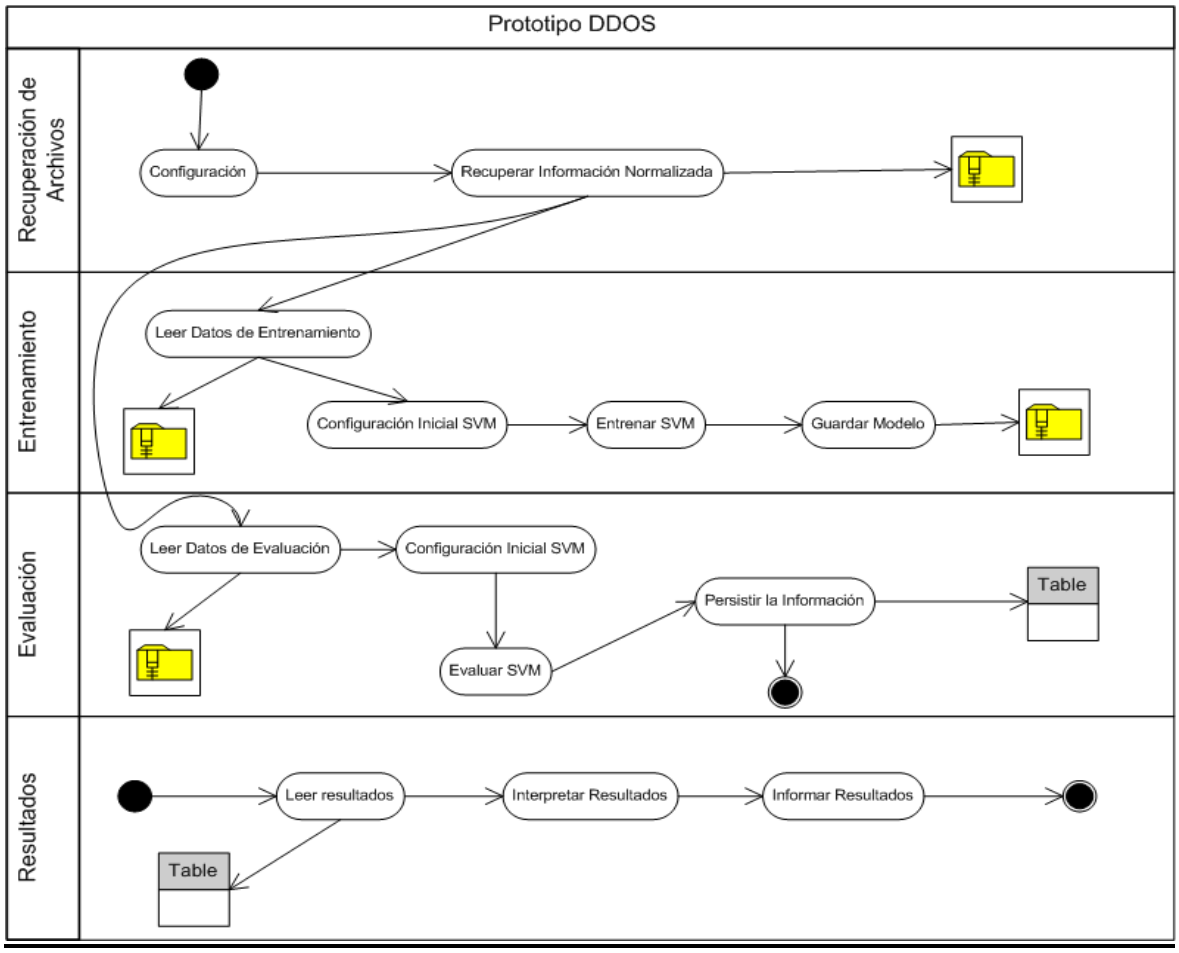


Figura 8: Arquitectura lógica del entrenamiento, evaluación y resultados

1. ***Recuperación de la información Normalizada***: Esta etapa es la encargada de recuperar el tráfico normalizado, después de haber realizado el proceso de captura y filtro en las etapas anteriores.

2. ***Entrenamiento***: Es la encargada de cargar la información normalizada, configurar los parámetros del modelo, según el núcleo aplicado y generar el modelo estadístico con la información de los comportamiento anómalos y normales.
3. ***Evaluación***: Esta etapa carga el modelo generado y generar un proceso de comparación a nivel estadístico del proceso de entrenamiento contra el proceso de evaluación actual, permitiendo detectar si la petición es anómala o normal.
4. ***Resultados***: En esta etapa se leen los resultados de la etapa de evaluación, se interpretan y se presentan para la respectiva interpretación.

3.3. FASE 2: IMPLEMENTACIÓN

En la fase de implementación se mostrarán los componentes más relevantes del generador de ataques y el prototipo computacional de detección, así como el flujo de interacción y comunicación entre los diferentes módulos que intervienen en el proceso.

3.3.1. PROTOTIPO GENERADOR DE ATAQUES DDOS:

Realizando el proceso de investigación se pudo observar que muchas de las herramientas implementadas para realizar ataques DDOS (LOIC, XOIC, HULK) no mostraban información técnica del desarrollo, haciendo que una parte de la solución del prototipo quedará con vacíos en el momento de su ejecución, por lo cual se tomó la decisión de realizar un diseño simple de un herramienta que hiciera múltiples peticiones a una dirección WEB, con el objetivo de comprender este tipo de comportamientos y mejorar la implementación del prototipo de detección.

El prototipo se desarrolló en el lenguaje de programación JAVA teniendo en cuenta las características multiplataforma y el manejo multihilos de la máquina virtual, permitiendo tener un control sobre cada una de las peticiones enviadas al servidor WEB atacado y generando estadísticas de tiempo en base a la respuesta, con el fin de conocer un poco más sobre la degradación del servicio. A continuación se muestra un diagrama de implementación del prototipo de ataques (Figura 9):

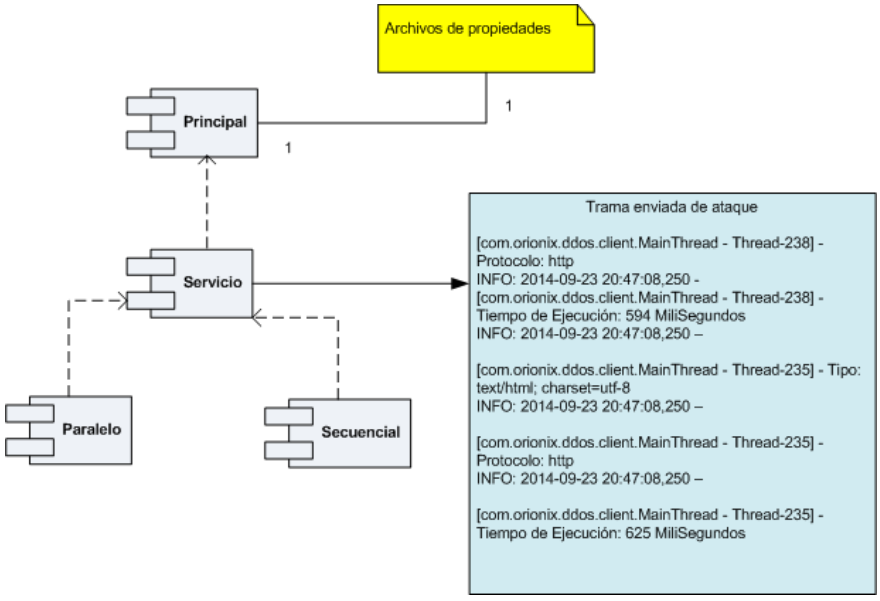


Figura 9: Componentes del prototipo de Ataques

La funcionalidad del prototipo es simple es su definición e implementación, ya que no es el objetivo específico del proyecto, pero aporta a la comprensión de la solución, por lo tanto cabe resaltar que el generador de ataques tiene 2 opciones de ejecución: En modo paralelo y en modo secuencial.

Modo Paralelo: En este modo se levanta un hilo por cada petición sin tiempo de espera, haciendo que el procesador del servidor WEB tenga que atender múltiples solicitudes y los recursos físicos de la máquina se vean comprometidos paulatinamente debido al gran flujo de tráfico.

Modo Secuencial: En este modo cada petición que se envía debe esperar la respuesta de la anterior para poder ejecutarse, haciendo que el procesador del servidor WEB reciba una solicitud al tiempo y los recursos físicos de la máquina reciban un flujo de tráfico normal.

En la siguiente figura se puede observar un fragmento de código, con la ejecución en paralelo instanciando la clase de Thread de JAVA, la cual maneja el procesamiento de los hilos, y la ejecución secuencial envía una solicitud siempre esperando la respectiva respuesta (Figura 10):

```
//Opcion de ejecutar el desarrollo en paralelo
Paralelo = Boolean.parseBoolean(Prop.getProperty("Paralelo"));
//*************************************************************
//Enviar tramas en paralelo (Hilos). mshll - Agosto - 2014
//*************************************************************
if (Main.Paralelo != false) {
    logger.info("Procesando en Paralelo.......");
    for (int j = 0; j < Iteraciones; j++) {
            Thread t = null;
            for (int i = 0; i < Peticiones; i++) {
                t = new Thread(new MainThread(Host, Espera));
                t.start();
                logger.info("Peticiones..." + i);
            }
        }
} //*************************************************************
//Enviar tramas secuenciales. mshll - Agosto - 2014
//*************************************************************
else if (Main.Paralelo != true) {
    for (int j = 0; j < Iteraciones; j++) {
        logger.info("Procesando secuencialmente.......");
        for (int i = 0; i < Peticiones; i++) {
            MainSequential.getUrlDDOS(Host, Peticiones);
        }
    }
}
```

Figura 10: Código del procesamiento paralelo y secuencial del generador de ataques

Básicamente el prototipo toma la decisión de enviar peticiones paralelas o secuenciales, teniendo como base un archivo de configuración donde se definen los parámetros de ejecución: Dirección WEB a atacar, Puerto destino, Tiempo de espera en la conexión al puerto, modo de ejecución (Paralelo o Secuencial), cantidad de peticiones en cada iteración y número de iteraciones totales. En la siguiente figura (11) se muestra la configuración:

```
#############################################################################
# Manuel Sebastián Hoyos Llanos
#############################################################################
#parámetros de conexión al aplicativo
#############################################################################
Host_proxy=http://192.168.1.17:8080/conciliatega/login.xhtml
Puerto_proxy=8080
Timeout_proxy=50000
#############################################################################
Paralelo=true
Peticiones=500
Iteraciones=5
#############################################################################
```

Figura 11: Configuración del prototipo generador de ataques

Cabe resaltar de la configuración anterior la decisión de diseño de definir cantidad de peticiones por iteración: Teniendo en cuenta que la ejecución en paralelo consume tantos recursos físicos tanto en el cliente como en el servidor atacado, es necesario definir una cantidad de peticiones por iteración, con el objetivo que la máquina virtual cargue en memoria peticiones agrupadas y desde el cliente no se generen excepciones de tipo OutOfMemory. A continuación se muestran peticiones realizadas al servidor WEB, así como el tiempo de ejecución promedio de una petición (Figura 12):

```
[com.orionix.ddos.client.utility.Propiedades - main] - Leyendo propiedades de aplicación...
[com.orionix.ddos.client.utility.Propiedades - main] - Carga exitosa propiedades de aplicación
[com.orionix.ddos.client.Main - main] - Iniciando el aplicativo DDOSCLiente...
[com.orionix.ddos.client.Main - main] - Leyendo los parametros de conexion del Cliente DDOS...
[com.orionix.ddos.client.Main - main] - Host: http://192.168.1.17:8080/conciliatega/
[com.orionix.ddos.client.Main - main] - Puerto: 8080
[com.orionix.ddos.client.Main - main] - Tiempo de Espera: 50000
[com.orionix.ddos.client.Main - main] - Peticiones: 200
[com.orionix.ddos.client.Main - main] - Iteraciones: 50
[com.orionix.ddos.client.Main - main] - Procesando en Paralelo.......
[com.orionix.ddos.client.Main - main] - Peticiones...45
[com.orionix.ddos.client.MainThread - Thread-17] - Respuesta: 200
[com.orionix.ddos.client.MainThread - Thread-17] - Tipo: text/html; charset=utf-8
[com.orionix.ddos.client.MainThread - Thread-17] - Protocolo: http
[com.orionix.ddos.client.MainThread - Thread-17] - Tiempo de Ejecución: 485 MiliSegundos
[com.orionix.ddos.client.MainThread - Thread-15] - Respuesta: 200
```

Figura 12: Log del prototipo generador de ataques

3.3.2. PROTOTIPO DE DETECCIÓN DE ATAQUES:

El prototipo de detección utilizó la misma arquitectura lógica planteada en la etapa de diseño (Figura 6), por lo tanto se explicará a nivel técnico la implementación del mismo, permitiendo mostrar el nivel de detalle requerido en este tipo de proyectos.

El prototipo de detección se encuentra dividido técnicamente de la siguiente manera: La recopilación, filtro y normalización del tráfico se realizó en SHELL SCRIPTING, teniendo en cuenta el rendimiento de las expresiones regulares y tuberías ofrecidas por el lenguaje. El entrenamiento y evaluación se realizó en el lenguaje de programación JAVA, teniendo en cuenta que la librería SVM está en dicho lenguaje y por las características mencionadas en el generador de ataques.

El artefacto computacional tiene definido los siguientes componentes en el diagrama de implementación (Figura 13):

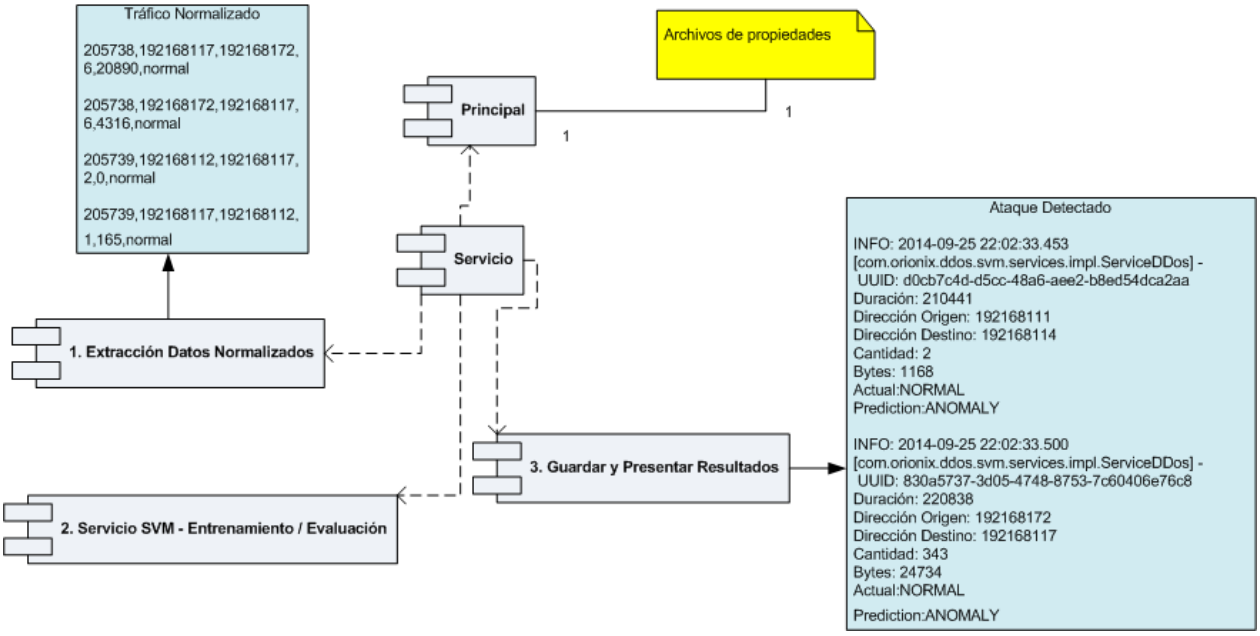


Figura 13: Componentes del prototipo de Detección de Ataques

En el diagrama de componentes anterior se puede observar como el prototipo tiene una etapa inicial de captura de la información normalizada, después realiza un proceso de entrenamiento y genera el modelo estadístico, se realiza la evaluación o detección a partir del modelo creado, se guardan los resultados y presentan para su respectiva interpretación.

3.3.2.1. RECOPILACIÓN DE LOS DATOS, FILTRO Y NORMALIZACIÓN:

Esta etapa es la encargada de recuperar el tráfico de red, filtrar y normalizar la información, teniendo en cuentas las variables operacionales definidas en la etapa de diseño. A continuación se describe el diagrama de componentes realizado para la recopilación de la información, filtro y normalización (Figura 14):

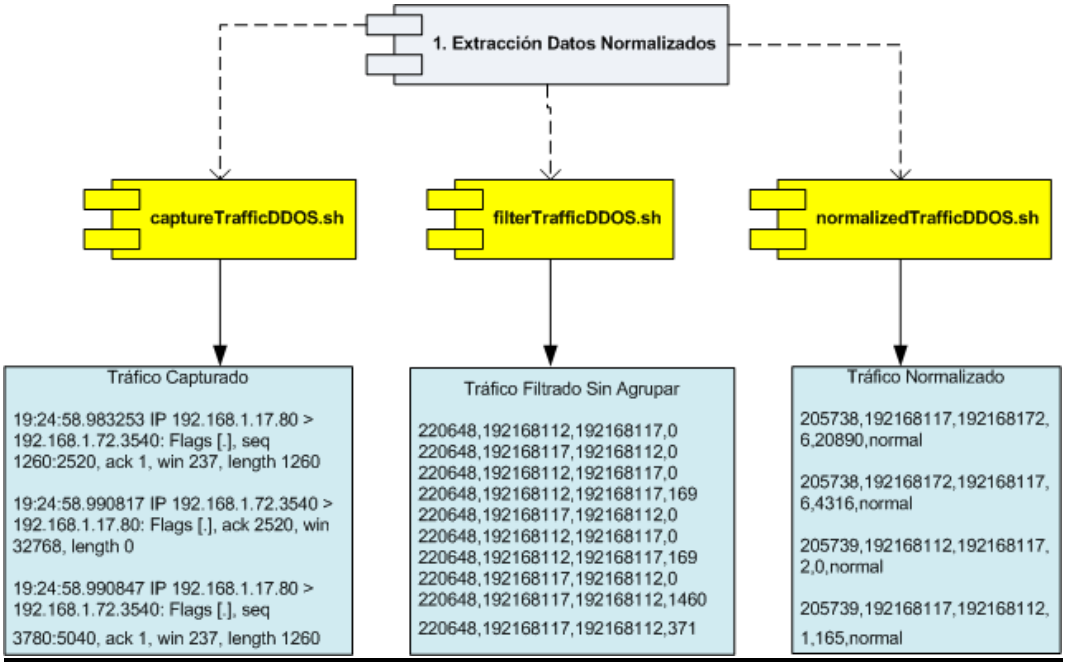


Figura 14: Componentes de la extracción, filtro y normalización

Recolección del conjunto de datos: Para la captura del tráfico de red, se utilizó el analizador TCPDUMP en el servidor y se filtraron todas las cadenas HTTP en un fichero PCAP, permitiendo manipular la información de una manera más clara.

La captura de dicho tráfico se encuentra en el script: captureTrafficDDOS.sh mostrado en la figura (14) y el objetivo es enviar todo el tráfico de las peticiones HTTP a los archivos con extensión PCAP, con el propósito que el proceso de limpieza y procesamiento capture la información de manera automática y lo analice. A continuación se muestra el tráfico capturado en los archivos PCAP (Figura 15):

```
354 0.529145   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
355 0.529273   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
356 0.529397   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
357 0.529522   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
358 0.530771   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
359 0.530897   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
360 0.532146   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
361 0.533396   192.168.1.17    192.168.1.72    TCP      1314 [TCP segment of a reassembled PDU]
362 0.533522   192.168.1.17    192.168.1.72    HTTP      534 HTTP/1.1 200 OK  (text/html)
363 0.534774   192.168.1.17    192.168.1.72    TCP        66 80→3686 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
364 0.537769   192.168.1.72    192.168.1.17    TCP        54 3655→80 [ACK] Seq=159 Ack=8821 Win=65536 Len=0
365 0.538146   192.168.1.72    192.168.1.17    TCP        54 3655→80 [ACK] Seq=159 Ack=11341 Win=65536 Len=0
366 0.543892   192.168.1.72    192.168.1.17    TCP        66 3688→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1260 WS=2 SACK_PERM=1
367 0.543913   192.168.1.17    192.168.1.72    TCP        66 80→3688 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
```

Figura 15: Estructura del archivo PCAP en la captura de tráfico

La información almacenada en los archivos PCAP contiene la negociación que se realiza a nivel del modelo OSI filtrado por las cadenas HTTP enviadas a través de Internet. A continuación se muestra parte del código de esta etapa (Figura 16):

```
function captureTraffic {

    x=0;
    x=`ps -ef|grep "tcpdump"|grep -v "grep"|awk '{print $2}' | head -1`

    #Verifica si el proceso tcpdump se está ejecutando - mshoyosll
    if [[ $x -eq 0 ]]; then
        echo  "$(date +%Y/%m/%d-%H:%M:%S) Nodo: $NODE " >> $FILE_LOG
        echo  "$(date +%Y/%m/%d-%H:%M:%S) Archivo de Directorio pcap: $FILE_DIR " >> $FILE_LOG
        echo  "$(date +%Y/%m/%d-%H:%M:%S) Comando tcpdump: sudo tcpdump -i $NETWORK -w $FILE_DIR/capture-%s.pcap
        echo  "$(date +%Y/%m/%d-%H:%M:%S) Archivo de Log: $FILE_LOG " >> $FILE_LOG
        #Envía el tráfico a los archivos pcap - mshoyosll
        `sudo -b tcpdump -i $NETWORK -w $FILE_DIR/capture-%s.pcap -G 600 -C $FILES -W $SIZE port $PROTOCOL `
        echo  "$(date +%Y/%m/%d-%H:%M:%S) El proceso TcpDump se ha iniciado. PID: $x " >> $FILE_LOG
    else
        echo  $x  > $FILE_PID
        echo  "$(date +%Y/%m/%d-%H:%M:%S) El proceso TcpDump ya se está ejecutando. PID: $x " >> $FILE_LOG
    fi

}
```

Figura 16: Código de la captura del tráfico

Después de descargar la información de la ubicación física del servidor, se realiza la respectiva limpieza, filtro y normalización en la siguiente etapa del proceso.

Limpieza y pre procesamiento de los valores de los parámetros: Este conjunto de datos se compone de 2 etapas de filtro y normalización, las cuales se describen a continuación:

1. **Etapa de Filtro:** El filtro del tráfico se encuentra en el script: filterTrafficDDOS.sh mostrado en la figura (14) y consiste en depurar los campos que no se utilizarán y filtrar las variables operacionales definitivas. Los parámetros de salida sin agrupar fueron los siguientes (Figura 17):

```
220648,192168112,192168117,0
220648,192168117,192168112,0
220648,192168112,192168117,0
220648,192168112,192168117,169
220648,192168117,192168112,0
220648,192168112,192168117,0
220648,192168117,192168112,0
220648,192168112,192168117,0
220648,192168112,192168117,169
220648,192168117,192168112,0
220648,192168117,192168112,1460
220648,192168117,192168112,371
220648,192168112,192168117,0
220648,192168117,192168112,0
```

Figura 17: Estructura del archivo con el filtro tráfico

La información almacenada en el archivo muestra las peticiones enviadas por los usuarios en cada instante de tiempo sin agrupar, haciendo que un mismo periodo puedan llegar desde 1 hasta N peticiones. A continuación se muestra parte del código de esta etapa (Figura 18):

```
#captureTraffic: Captura de tráfico
function filterTraffic {


    ##################################
    #mover los archivo base a la entrada del filtro
    ##################################
    moveTrafficBaseFiles $FILE_INPUT_BASE $FILE_INPUT_FILTER

    ##################################
    #Realizar el filtro de las cadenas
    ##################################

    echo  "$(date +%Y/%m/%d-%H:%M:%S) Nodo: $NODE " >> $FILE_LOG
    echo  "$(date +%Y/%m/%d-%H:%M:%S) Directorio pcap Base: $FILE_INPUT_BASE " >> $FILE_LOG
    echo  "$(date +%Y/%m/%d-%H:%M:%S) Directorio pcap Filter: $FILE_INPUT_FILTER " >> $FILE_LOG
    echo  "$(date +%Y/%m/%d-%H:%M:%S) Directorio pcap Temp: $FILE_INPUT_TEMP " >> $FILE_LOG
    echo  "$(date +%Y/%m/%d-%H:%M:%S) Archivo de Salida Filter: $FILE_OUTPUT_FILTER " >> $FILE_LOG
    echo  "$(date +%Y/%m/%d-%H:%M:%S) Archivo de Log: $FILE_LOG " >> $FILE_LOG

    cd $FILE_INPUT_FILTER
    FILE_INPUT_FILTER_TEMP=$FILE_INPUT_TEMP$(date +%Y-%m-%d)-filterTemp.log
```

Figura 18: Código del filtro del tráfico

2. **Etapa de Normalización:** La normalización o agrupación del tráfico se encuentra en el script: normalizedTrafficDDOS.sh mostrado en la figura (14) y consiste en agrupar en un

periodo de tiempo (Segundos), la cantidad de peticiones y envío de información (Bytes) para las direcciones IP origen y destino. Los parámetros de salida agrupados fueron los siguientes (Figura 19):

```
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
220838,192168172,192168117,343,24734,normal
220839,192168112,192168117,100,1690,normal
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
220838,192168172,192168117,343,24734,normal
220839,192168112,192168117,100,1690,normal
```

Figura 19: Estructura del archivo normalizado

La información normalizada tiene un proceso de agrupamiento complejo a nivel de recursos físicos del sistema operativo, debido a que por cada petición realizada, se agrupan las solicitudes y el tráfico recibido, con el objetivo que se genere un único registro en un instante de tiempo. A continuación se muestra parte del código de esta etapa (Figura 20):

```
#Se recorre el archivo generado, para crear un nuevo archivo con las variables. mshoyos11

echo  "$(date +%Y/%m/%d-%H:%M:%S) Archivo Final: $FILE_OUTPUT_NORMALIZED " >> $FILE_LOG
for lineFile in `cat $capfile | sort | awk -F"," '{print $1","$2","$3}' | uniq -c  | awk -F" "
do
    count=$(echo $lineFile | awk -F"," '{print($1)'})
    date1=$(echo $lineFile | awk -F"," '{print($2)'})
    ip_src=$(echo $lineFile | awk -F"," '{print($3)'})
    ip_dest=$(echo $lineFile | awk -F"," '{print($4)'})
    filter=$date1","$ip_src
    bytes=`cat $capfile | awk -F"," '{print $1","$2","$4}'   | grep "$filter" | awk -F"," '{a[$
    lineTemp=$date1","$ip_src","$ip_dest","$count","$bytes
    echo -e $lineTemp >> $FILE_OUTPUT_NORMALIZED
done
```

Figura 20: Código del tráfico normalizado

Después de tener la información normalizada, el prototipo captura los parámetros a través de una conexión segura SFTP[20] y sube la información para entrenamiento y evaluación en un directorio

[20] SSH File Transfer Protocol

físico del servidor. El código que realiza el procedimiento de extracción se muestra en la siguiente Figura (21):

```
//Valida si utilizará el sftp - mshoyos11
if (sftp!= null){
    if (sftp.equals("true")){
        logger.info("Acceder por los archivos a través de SFTP ");
        FacadeSftp facadeSftp = new FacadeSftp();
        facadeSftp.executeSftp(server, user, password, port, source, sourceBackup, evaluation);
        logger.info("Acceder por los archivos a través de SFTP Terminado ");
    }
}
```

Figura 21: Código de la extracción de la información por SFTP

Al cargar la información con las variables operacionales en el archivo físico, se genera el punto de partida para que la SVM realice el proceso de entrenamiento o evaluación del prototipo.

3.3.2.2. ENTRENAMIENTO DEL PROTOTIPO:

La etapa de entrenamiento consiste en recibir la información normalizada y enviarla a la SVM, para que realice el proceso de agrupamiento estadístico de las peticiones anómalas y normales en el modelo generado. A continuación se presenta el diagrama de componentes del proceso de entrenamiento (Figura 22):

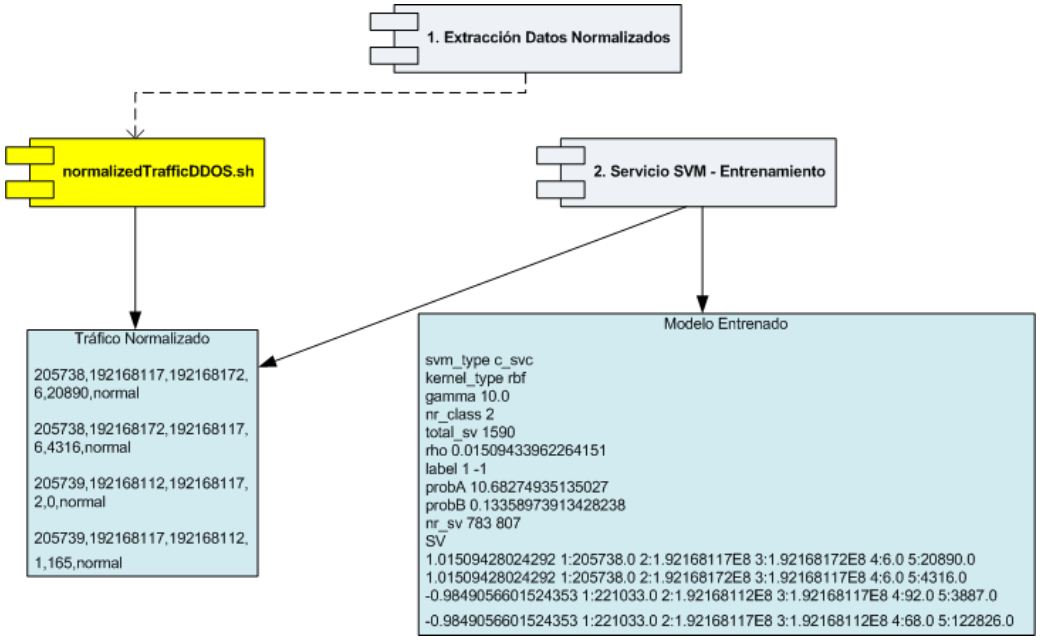


Figura 22: Componentes en la etapa de entrenamiento

El prototipo integra la librería *libsvm*, la cual permite entrenar, evaluar y generar un modelo estadístico, teniendo como base la información normalizada. La estructura interna del modelo generado se presenta a continuación (Figura 23):

```
svm_type c_svc
kernel_type rbf
gamma 10.0
nr_class 2
total_sv 1590
rho 0.01509433962264151
label 1 -1
probA 10.68274935135027
probB 0.13358973913428238
nr_sv 783 807
SV
1.01509428024292 1:205738.0 2:1.92168117E8 3:1.92168172E8 4:6.0 5:20890.0
1.01509428024292 1:205738.0 2:1.92168172E8 3:1.92168117E8 4:6.0 5:4316.0
-0.9849056601524353 1:221033.0 2:1.92168112E8 3:1.92168117E8 4:92.0 5:3887.0
-0.9849056601524353 1:221033.0 2:1.92168117E8 3:1.92168112E8 4:68.0 5:122826.0
```

Figura 23: Modelo generado por la SVM en el prototipo

El modelo es un matriz donde agrupa las variables operacionales ingresadas y genera un valor real de clasificación para los resultados: (Normal: 1.0 y Anómalo: -0.0).El modelo es almacenado físicamente en una ruta del servidor y es la base en la fase de evaluación o detección. El código que realiza el procedimiento de entrenamiento se muestra en la siguiente figura (24):

```
//mshoyosll - 2014-10-03
problem.x = svm_nodes;
problem.y = labels;
problem.l = labels.length;
svm_model model = svm.svm_train(problem, parameter);
try {
    svm.svm_save_model(pathFile,model);
} catch (IOException e) {
    throw new ExceptionDDos("Error guardando el modelo" + e.toString());
}
return model;
```

Figura 24: Código del proceso de entrenamiento

A nivel técnico es importante resaltar que los cálculos de las métricas de desempeño las realiza la librería *(libsvm)* y no las librerías propias de la herramienta WEKA, como gráficamente se puede observar al momento de hacer la clasificación.

3.3.2.3. EVALUACIÓN DEL PROTOTIPO:

La etapa de evaluación consiste en recibir el tráfico normalizado y enviarlo a la SVM, para que realice el proceso de detección del ataque, teniendo como base el modelo generado en la etapa de entrenamiento. A continuación se presenta el diagrama de componentes del proceso de evaluación (Figura 25):

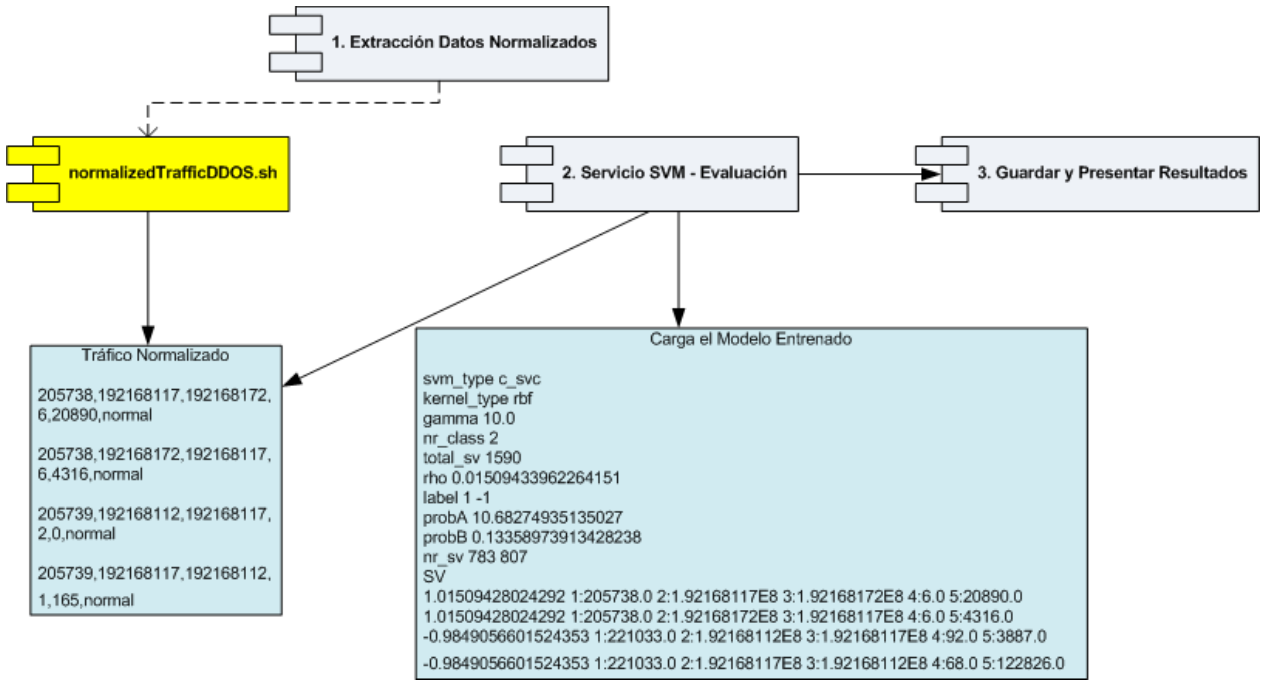


Figura 25: Componentes en la etapa de evaluación o detección

El proceso de evaluación tiene como punto de partida el tráfico enviado por los usuarios y normalizado en la etapa inicial, el prototipo carga el modelo generado y analiza cada registro de manera individual, con el objetivo de compararlo con el modelo estadístico generado en la etapa

de entrenamiento e identificar si efectivamente se está generando un ataque. A continuación se presenta el código que realiza el procedimiento de evaluación (Figura 26):

```
//mshoyosll - 2014-10-03
int totalClasses = 2;
int[] labels = new int[totalClasses];
svm.svm_get_labels(model,labels);

double[] prob_estimates = new double[totalClasses];
double result_prob = svm.svm_predict_probability(model, nodes,prob_estimates);
double result_normal = svm.svm_predict(model, nodes);
System.out.println("Result with prob " + result_prob);
System.out.println("Result normal " + result_normal);
System.out.println("Probability " + prob_estimates[0] + "\t"
        + prob_estimates[1]);

for (int i = 0; i < prob_estimates.length; i++) {
    System.out.println("(" + labels[i] + ":" + prob_estimates[i] + ")");
}
```

Figura 26: Código del proceso de evaluación

En el código anterior se reciben las etiquetas utilizadas para realizar el proceso (*labels: Valor 1.0, -0.0*), el modelo generado en el entrenamiento (*model*) y la información normalizada para la evaluación (*nodes*), tomando cada registro y verificando si efectivamente hay un ataque.

3.3.2.4. RESULTADOS DEL PROTOTIPO:

La información capturada en el proceso de evaluación es almacenada en una base de datos relacional y tienen la siguiente estructura:

TABLA TBLEXECUTION				
INFORMACIÓN BÁSICA DE LA EJECUCIÓN.				
NAME	**DATATYPE**	**NULL OPTION**	**PRIMARY KEY**	**FOREIGN KEY**
UUID	**VARCHAR2(50)**	**NOT NULL**	**YES**	**NO**
NAME	**VARCHAR2(200)**	**NULL**	**NO**	**NO**

TABLA TBLEXECUTION				
INFORMACIÓN BÁSICA DE LA EJECUCIÓN.				
NAME	**DATATYPE**	**NULL OPTION**	**PRIMARY KEY**	**FOREIGN KEY**
DATEINSERT	DATE	NULL	NO	NO

Tabla 4: Tabla física ejecución (TBLEXECUTION)

TABLA TBLRESULTOPERATIONVARIABLES				
INFORMACIÓN DE LA VARIABLES OPERACIONALES DE ENTRADA, MUESTRA EL COMPORTAMIENTO ACTUAL DEL REGISTRO EN LA FASE DE ENTRENAMIENTOY EL VALOR DE PREDICCIÓN EN EJECUCIÓN DE LA SVM.				
NAME	**DATATYPE**	**NULL OPTION**	**PRIMARY KEY**	**FOREIGN KEY**
UUID	VARCHAR2(50)	NOT NULL	YES	NO
DURATION	VARCHAR2(50)	NULL	NO	NO
IP_SRC	VARCHAR2(50)	NULL	NO	NO
IP_DEST	VARCHAR2(50)	NULL	NO	NO
IP_SRC_COUNT	VARCHAR2(50)	NULL	NO	NO
BYTES	VARCHAR2(50)	NULL	NO	NO
CLASS1	VARCHAR2(3)	NULL	NO	NO
ACTUAL	VARCHAR2(10)	NULL	NO	NO
PREDICTION	VARCHAR2(10)	NULL	NO	NO
DATEINSERT	DATE	NULL	NO	NO
UUIDEXECUTION	VARCHAR2(50)	NOT NULL	NO	YES

Tabla 5: Tabla física (TBLRESULTOPERATIONVARIABLES)

TBLRESULTOPERATIONVARIABLESTATISTICS

INFORMACIÓN DE LA VARIABLES OPERACIONALES DE SALIDA, MUESTRA LAS MÉTRICAS DE DESEMPEÑO DEL MODELO EN LA EVALUACIÓN.				
NAME	**DATATYPE**	**NULL OPTION**	**PRIMARY KEY**	**FOREIGN KEY**
UUID	**VARCHAR2(50)**	**NOT NULL**	**YES**	**NO**
TFP	**FLOAT**	**NOT NULL**	**NO**	**NO**
TFN	**FLOAT**	**NOT NULL**	**NO**	**NO**
TTN	**FLOAT**	**NOT NULL**	**NO**	**NO**
TTP	**FLOAT**	**NOT NULL**	**NO**	**NO**
CLASIFICATION	**FLOAT**	**NULL**	**NO**	**NO**
SENSIBILITY	**FLOAT**	**NULL**	**NO**	**NO**
DATEINSERT	**DATE**	**NULL**	**NO**	**NO**
UUIDEXECUTION	**VARCHAR2(50)**	**NOT NULL**	**NO**	**YES**

Tabla 6: Tabla física (TBLRESULTOPERATIONVARIABLESTATISTICS)

El modelo relacional mostrado en las figuras anteriores, básicamente resume la funcionalidad del prototipo a través de la generación de 2 tipos de reporte para la respectiva interpretación:

1. Evaluación del modelo predictivo vs entrenado: Se encarga de analizar registro por registro, si la información ingresada de las variables operacionales es coherente con el proceso de entrenamiento, arrojando un valor predictivo y comparándolo respectivamente con el actual. A continuación se presenta un ejemplo del modelo predictivo almacenado(Figura 27):

DURATION	IP_SRC	IP_DEST	IP_SRC_COUNT	BYTES	CLASS1	ACTUAL	PREDICTION
210441	192168111	192168114	2	1168	1	NORMAL	ANOMALY
220839	192168112	192168117	100	1690	1	NORMAL	ANOMALY

Figura 27: Almacenamiento de las variables operacionales de entrada

Se almacenan las variables operacionales definidas en las estructura de entrada (Tabla 5) y se realiza la respectiva comparación entre el valor entrenado (Columna ACTUAL) contra el valor

predictivo (Columna PREDICTION) realizado por la SVM, permitiendo realizar una detección detallada del ataque DDOS.

2. Variables Operacionales por defecto: El prototipo invoca la librería (*libsvm*) y calcula los valores de salida, tales como: TFP (Tasa de falsos positivos), TFN (Tasa de falsos negativos), TVN (Tasa de verdaderos negativos), TVP (Tasa de verdaderos positivos) y Porcentaje de Clasificación. A continuación se presenta un ejemplo de las métricas de desempeño almacenadas (Figura 28):

UUID	TFP	TFN	TTN	TTP	CLASIFICATION	SENSIBILITY	DATEINSERT	UUIDEXECUTION
6ba51774-9316-4...	0.2	0	0	0.98	0.99	0.98	2015-04-04	6ba51774-9316-4103...
6ba51009-9316-1...	0.3	0	0	0.98	0.98	0.98	2015-04-04	087d5974-4175-4c68...

Figura 28: Almacenamiento de las métricas operacionales del modelo

Las métricas de desempeño anterior, permiten evaluar el modelo en el momento de la ejecución, analizando el porcentaje de peticiones que se pueden tomar como falsos positivos (TFP), a partir del modelo generado en el entrenamiento. Entre más alto sea el porcentaje de TFP y más bajo sea el valor de clasificación, menos confiable será el modelo generado para la detección de ataques.

En la fase de pruebas se realizará un experimento donde se mostrará el detalle, definición, formulas y resultados de las métricas de desempeño, como la detección del ataque por parte del prototipo.

3.4. FASE 3: PRUEBAS Y RESULTADOS

Para la recopilación de la información del ataque, entrenamiento y pruebas del prototipo, se realizó un experimento, el cual utilizó las herramientas de red necesarias para la simulación, utilizando la siguiente arquitectura física (Figura 29):

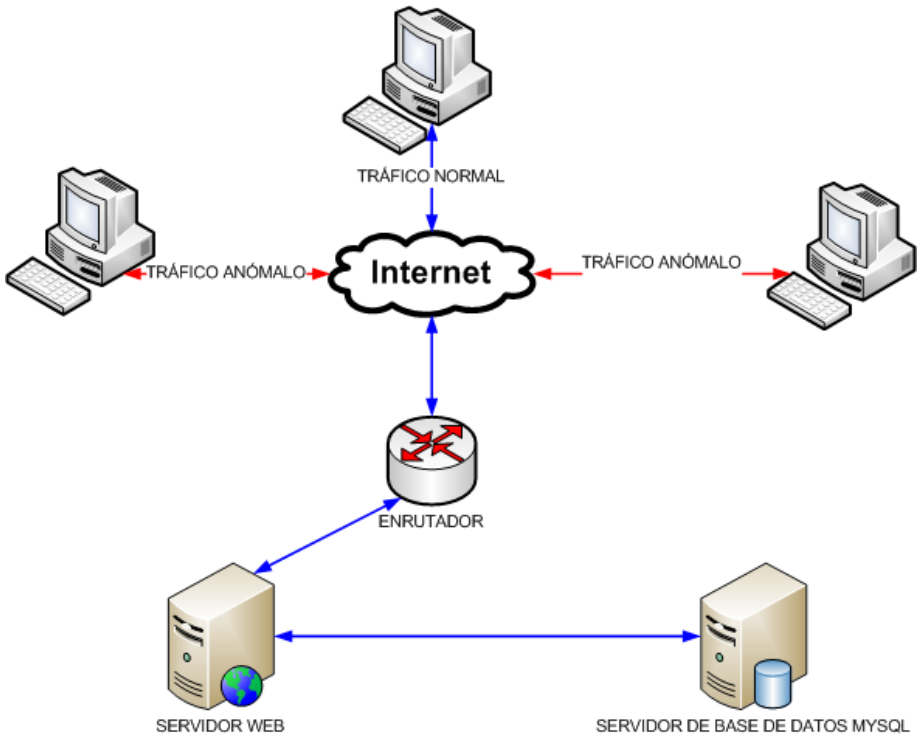


Figura 29: Arquitectura física del experimento

Se utilizó 1 servidor con sistema operativo UBUNTU, 3 clientes que simulaban el tráfico normal y anómalo, a través de un sistema de información WEB instalado en un servidor de aplicaciones TOMCAT y una base de datos MYSQL.

3.4.1. RECOPILACIÓN DE LA INFORMACIÓN:

Con el objetivo de comprender mejor la información recuperada y por ende el ataque, la información utilizada para las pruebas no fue el conjunto de datos conocido como *KDD99* (DARPA & AFRL/SNHS), sino que generamos un conjunto de datos propio a partir de la simulación de un ataque según la arquitectura física planteada en la Figura (29). La estructura

final del archivo normalizado está representada por las variables operacionales definidas en la etapa de Diseño (Tabla 7):

Variable	Descripción	Tipo
DURATION	Tiempo en segundos de cada petición.	Continuo
IP_SRC	Dirección IP origen	Continuo
IP_DEST	Dirección IP destino.	Continuo
IP_SRC_COUNT	Cantidad de peticiones desde la IP origen.	Continuo
BYTES	Cantidad de bytes enviados desde un origen hacia un destino.	Continuo
CLASS	Clasificación del tráfico: (Normal, Anómalo).	Discreto

Tabla 7: Estructura de atributos normalizados

En la ejecución del experimento para la fase de recopilación de la información, se enviaron un conjunto de peticiones normales y anómalas de manera paralela y secuencial con el prototipo generador de ataques (*3.3.1 Prototipo generador de ataques DDOS*) al sistema de información de pruebas, enviando 5 iteraciones de 500 peticiones en un periodo de tiempo determinado de 15 minutos. A continuación se muestra el sistema de información de pruebas atacado (Figura 30):

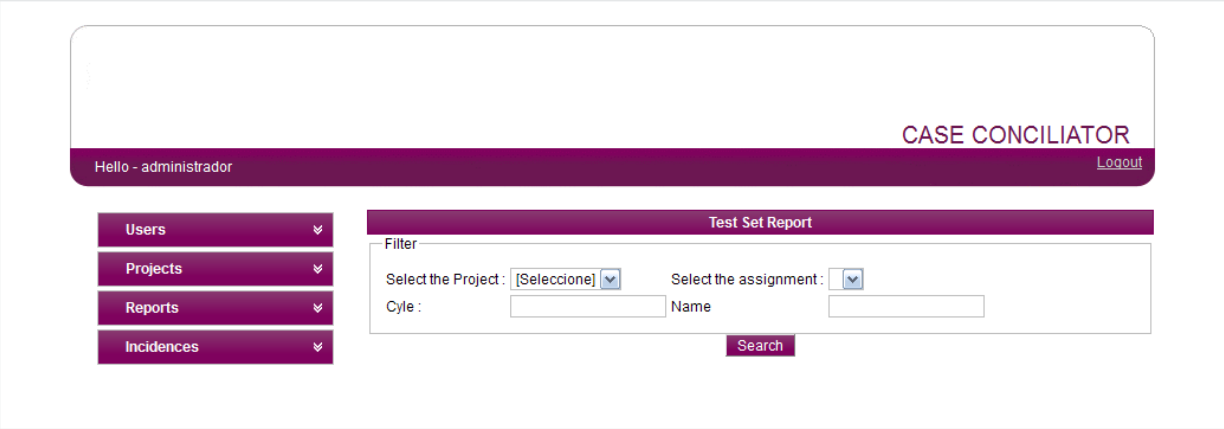


Figura 30: Sistema de información de pruebas

En total se agruparon y almacenaron 2698 registros entre tráfico normal y anómalo. Los datos completos del tráfico normalizado (normal 1349, anómalo 1349), son divididos en la etapa de entrenamiento, seleccionando al azar el 60% del conjunto de datos (normal 809, anómalo 809), el otro 40% (normal 539, anómalo 539) para la evaluación. A continuación se presenta un ejemplo de la información almacenada en el archivo normalizado (Figura 31):

```
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
220838,192168172,192168117,343,24734,normal
220839,192168112,192168117,100,1690,normal
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
220838,192168172,192168117,343,24734,normal
220839,192168112,192168117,100,1690,normal
```

Figura 31: Información del archivo normalizado en pruebas

Con el porcentaje de entrenamiento y evaluación definido, se procede a realizar las respectivas pruebas al prototipo, con el objetivo de determinar si tiene un nivel de detección aceptable. Antes de mirar los resultados de la fase de entrenamiento y evaluación, debemos entender las definiciones y fórmulas expuestas(Echeverri, O., Trujillo, & Marulanda, 2010):

– *VP: verdaderos positivos:* Paquetes correctamente detectados. La Tasa de Verdaderos Positivos está determinada por:

$$TasaVerdaderosPositivos(TVP) = \frac{\sum_{i=1}^{n} VP_i}{\sum_{i=1}^{n} A_i}$$

Figura 32: Fórmula de tasa de verdaderos positivos

– *FN: falsos negativos*: Paquetes que no fueron detectados, determinada por:

$$TasaFalsos\,Negativos(TFN) = \frac{\sum_{i=1}^{n} FN_i}{\sum_{i=1}^{n} A_i}$$

Figura 33: Fórmula de tasa de falsos negativos

– *FP: falsos positivos:* Paquetes que el modelo marcó como objetivos detectados, pero realmente no lo son.

$$TasaFalsos\,Positivos(TFP) = \frac{\sum_{i=1}^{n} FP_i}{\sum_{i=1}^{n} N_i}$$

Figura 34: Fórmula de tasa de falsos positivos

La sensibilidad (S) puede determinarse en función de los verdaderos positivos, falsos positivos y verdaderos negativos como:

$$S = \frac{VP}{VP + FN}$$

Figura 35: Fórmula de sensibilidad

La precisión se determina por:

$$\text{Pr}ecisiónCla\,sificación = \frac{\sum_{i=1}^{n} VP_i}{\sum_{i=1}^{n} A_i} \qquad Precisión = \frac{VP}{VP + FP}$$

Figura 36: Fórmula de precisión

Este tipo de métricas de desempeño, permiten la toma de decisiones y facilitan la visualización de los verdaderos positivos (VP) y falsos positivos (FP) usando curvas ROC. Una gráfica ROC es: "*Una técnica que permite visualizar, organizar y seleccionar clasificadores basada en su propio funcionamiento. Este tipo de gráfica es de dos dimensiones en la cual la fracción de VP se dispone en el eje Y y la fracción de FP en el eje X. Una curva ROC presenta una compensación entre los beneficios (VP) y los costos (FP) de un modelo*"(Sanchez, Henao, & Mauricio, 2007, pág. 44).

3.4.2. ENTRENAMIENTO DEL PROTOTIPO:

Seleccionamos al azar el 60% del conjunto de datos (normal 809, anómalo809), y realizamos el respectivo entrenamiento al prototipo, guardando la información en un archivo y clasificando los datos manualmente con el tráfico normal y anómalo. Este proceso debe realizarse manualmente debido a que la SVM es un método supervisado, que requiere de un previo entrenamiento para la generación del modelo (*3.3.2.2 Entrenamiento del Prototipo*) y respectiva evaluación del ataque. A continuación se presenta un ejemplo de la información clasificada manualmente en el archivo normalizado para la etapa de entrenamiento (Figura 37):

```
220838,192168117,192168112,295,58872,anomaly
220838,192168117,192168172,200,58872,anomaly
220838,192168172,192168117,343,24734,anomaly
220839,192168112,192168117,100,1690,anomaly
205738,192168117,192168172,6,20890,normal
205738,192168172,192168117,6,4316,normal
205739,192168112,192168117,2,0,normal
205739,192168117,192168112,1,165,normal
205741,192168112,192168117,23,7998,normal
205741,192168117,192168112,25,12102,normal
```

Figura 37: Información del archivo de entrenamiento en pruebas

Después de tener la información de entrenamiento, el prototipo genera el modelo y analiza las métricas de desempeño, las cuales permitirán determinar si el nivel de detección es aceptable o no.

Con el fin de garantizar y evidenciar la funcionalidad del prototipo, se realizó una revisión de los parámetros de desempeño obtenidos a través del software gráfico estadístico para minería de datos WEKA contra los valores obtenidos con el prototipo para la fase de entrenamiento y evaluación.

3.4.2.1. VALIDACIÓN DE LOS RESULTADOS CON WEKA:

A continuación se presentan los resultados de las pruebas de entrenamiento tomando el 60% de la información normalizada con la herramienta WEKA (Figura 38):

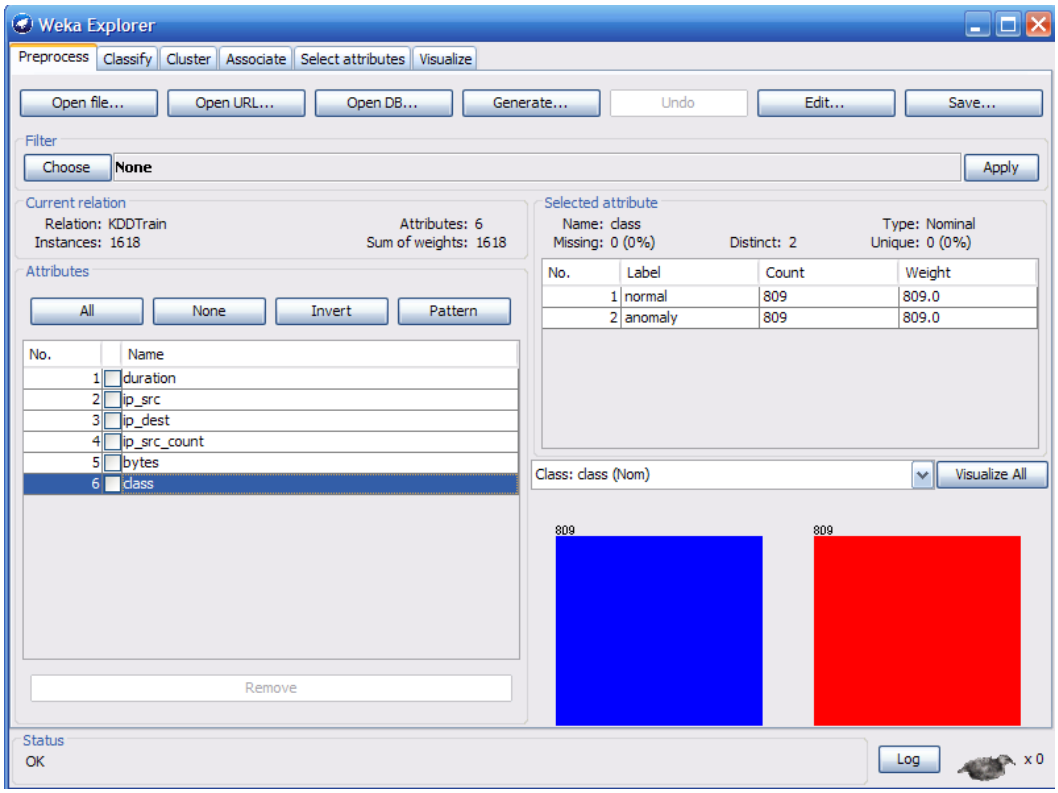


Figura 38: Pre procesamiento de la información en WEKA

Se carga el archivo normalizado con 1618 instancias entre tráfico normal y anómalo con la estructura requerida por WEKA (Figura 39):

```
@relation 'KDDTrain'
@attribute 'duration' real
@attribute 'ip_src' real
@attribute 'ip_dest' real
@attribute 'ip_src_count' real
@attribute 'bytes' real
@attribute 'class' {'normal' , 'anomaly'}
```

Figura 39: Estructura requerida en los archivos en WEKA

Se realiza el proceso de clasificación o entrenamiento teniendo como base las variables operacionales y el núcleo RBF definido en etapas anteriores. (Figura 40):

```
=== Summary ===

Correctly Classified Instances        1616               99.8764 %
Incorrectly Classified Instances         2                0.1236 %
Kappa statistic                          0.9975
Mean absolute error                      0.0012
Root mean squared error                  0.0352
Relative absolute error                  0.2472 %
Root relative squared error              7.0316 %
Coverage of cases (0.95 level)          99.8764 %
Mean rel. region size (0.95 level)      50      %
Total Number of Instances             1618

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,999    0,001    0,999      0,999    0,999      0,998    0,999     0,998     normal
                 0,999    0,001    0,999      0,999    0,999      0,998    0,999     0,998     anomaly
Weighted Avg.    0,999    0,001    0,999      0,999    0,999      0,998    0,999     0,998

=== Confusion Matrix ===

   a   b   <-- classified as
 808   1 |   a = normal
   1 808 |   b = anomaly
```

Figura 40: Entrenamiento de la información normalizada en WEKA

La interpretación de estos resultados permitirá determinar si el modelo entrenado tiene los valores mínimos requeridos para la detección de los ataques. A continuación se explican los valores obtenidos en el entrenamiento (Tabla 8):

Métricas de desempeño en fase de Entrenamiento WEKA			
	Anómalo	Normal	
Anómalo	TVP: 99.9%	TFP: 0.1%	
Normal	TFN	TVN	
Clasificación: 99.87%		Precisión: 99.9%	Sensibilidad: 99.9%

Tabla 8: Resultado de las métricas de desempeño en entrenamiento WEKA

Los resultados del proceso de entrenamiento a través de la herramienta muestra como el porcentaje de clasificación correcto tiene un valor bastante alto (99.8764%), el número de instancias clasificadas incorrectamente bastante bajo (0.1236%), la cantidad de falsos positivos es bajo (0.1%), la precisión y sensibilidad bastante altas (99%). Estos resultados nos indican que la información de 1618 instancias clasificadas como normales y anómalas permiten tener un excelente nivel de confiabilidad en la detección para la fase de evaluación, teniendo como base los porcentajes de clasificación correcto, la precisión, sensibilidad y el porcentaje de clasificación incorrecto del modelo(0.1236%). En la siguiente Figura (41) se puede observar la visualización basada en Curvas ROC de la tasa de Falsos Positivos y Verdaderos Positivos (Tasa de Detección):

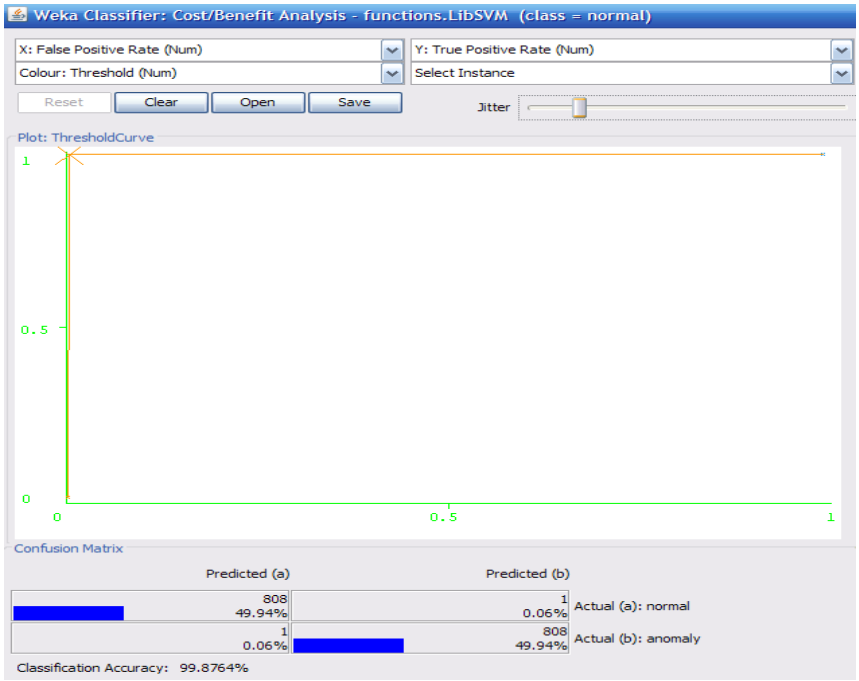


Figura 41: Curva ROC en entrenamiento WEKA

En el gráfico anterior podemos observar que en el eje X están los falsos positivos y en el eje Y los verdaderos positivos, si el valor en el eje Y se acerca a 1 y el valor en el eje X se acerca a 0, significa que los eventos tendrán mayor capacidad de detección y por ende, mayor capacidad de discriminación entre un comportamiento normal o anómalo.

3.4.2.2. VALIDACIÓN DE LOS RESULTADOS CON EL PROTOTIPO:

Para la realización de las pruebas del prototipo en la fase de entrenamiento, primero capturamos la información normalizada y la guardamos en una carpeta configurada en el prototipo (Figura 42):

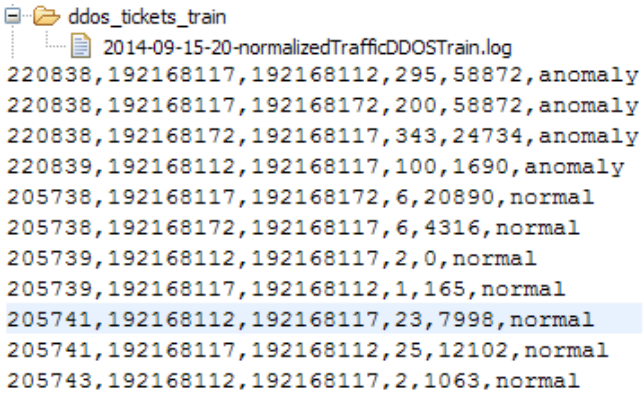


Figura 42: Información normalizada en entrenamiento del prototipo

Al tener la información normalizada, procedemos a ejecutar el proceso de entrenamiento, cuyos pasos se puede observar en la siguiente traza de ejecución (Figura 43):

```
ServiceDDos] - Servidor SFTP: 192.168.1.17
ServiceDDos] - Usuario SFTP: orionix
ServiceDDos] - Directorio Origen SFTP: /home/orionix/DDos/1_WorkFlow/2_Normalized/OutPut/
ServiceDDos] - Directorio Backup SFTP: /home/orionix/DDos/1_WorkFlow/2_Normalized/OutPut/backup/
ServiceDDos] - Directorio Destino Local Entrenamiento: ddos_tickets_train
ServiceDDos] - Directorio Destino Local Evaluacion: ddos_tickets_evaluation
ServiceDDos] - Directorio Destino Local Backup : ddos_tickets_backup
ServiceDDos] - Directorio Modelo WEKA: \modelSVMDDOS1.model
ServiceDDos] - Acceder por los archivos a través de SFTP
ServiceDDos] - Acceder por los archivos a través de SFTP Terminado
ServiceDDos] - Recuperar la información de la carpeta de Datos Normalizados Entrenamiento: ddos_tickets_train
ServiceDDos] - Recuperar la información de la carpeta de Datos Normalizados Entrenamiento Terminada
ServiceDDos] - Iteramos los archivos de la carpeta de Datos: 2014-09-15-20-normalizedTrafficDDOSTrain.log.11111
ServiceDDos] - Obtenemos las variables operacionales en un objeto tipado 2014-09-15-20-normalizedTrafficDDOSTrain.log.11111
ServiceDDos] - Ejecutamos el entrenamiento de la SVM a partir del modelo de datos obtenido
ServiceDDos] - Entrenamiento de la SVM a partir del modelo de datos obtenido finalizado
```

Figura 43: Traza general de la ejecución de entrenamiento del prototipo

1. Se hace una conexión SFTP y se extrae la información para el entrenamiento o evaluación y se colocan en una ruta local (Figura 44):

```
Servidor SFTP: 192.168.1.17
Usuario SFTP: orionix
Directorio Origen SFTP: /home/orionix/DDos/1_WorkFlow/2_Normalized/OutPut/
Directorio Backup SFTP: /home/orionix/DDos/1_WorkFlow/2_Normalized/OutPut/backup/
Directorio Destino Local Entrenamiento: ddos_tickets_train
Directorio Destino Local Evaluacion: ddos_tickets_evaluation
Directorio Destino Local Backup : ddos_tickets_backup
```

Figura 44: Información normalizada en entrenamiento del prototipo

2. Se obtiene la información del archivo entrenamiento y se carga en memoria (Figura 45):

```
Recuperar la información de la carpeta de Datos Normalizados Entrenamiento: ddos_tickets_train
Recuperar la información de la carpeta de Datos Normalizados Entrenamiento Terminada
Iteramos los archivos de la carpeta de Datos: 2014-09-15-20-normalizedTrafficDDOSTrain.log.11111
Obtenemos las variables operacionales en un objeto tipado 2014-09-15-20-normalizedTrafficDDOSTrain.log.11111
```

Figura 45: Información normalizada cargada en memoria del prototipo

3. Se genera el modelo de entrenamiento y se guarda en la carpeta (Figura 46):

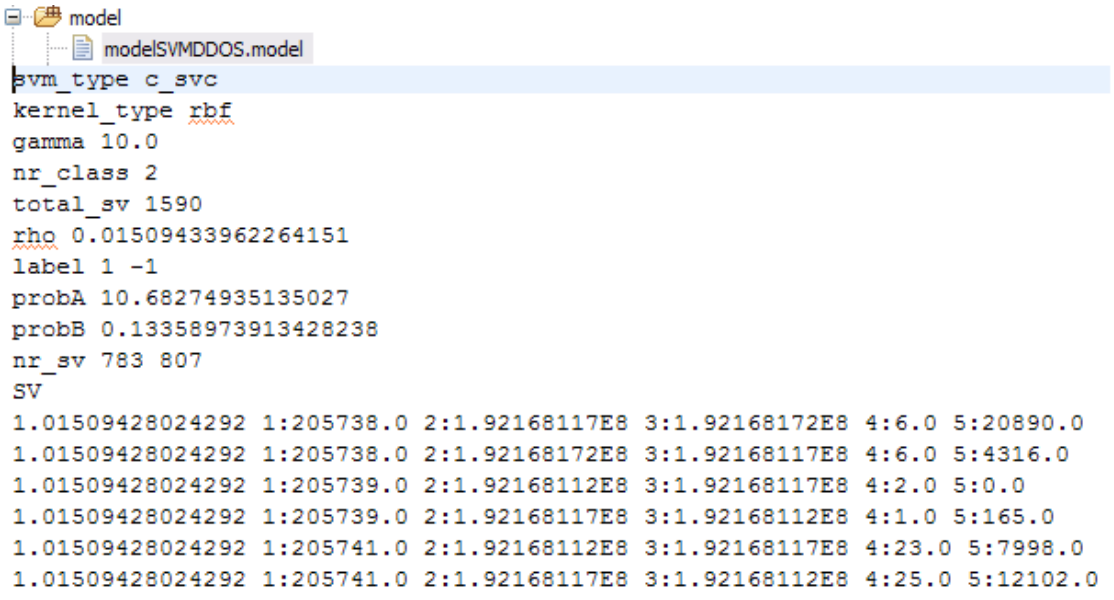


Figura 46: Modelo generado por la SVM en entrenamiento del prototipo

4. Se guardan las estadísticas de desempeño en la base de datos, según la estructura definida en la fase de implementación para la respectiva interpretación por parte del experto (Figura 47):

UUID	TFP	TFN	TTN	TTP	CLASIFICATION	SENSIBILITY	DATEINSERT	UUIDEXECUTION
6ba51774-9316-4...	0.2	0	0	0.98	0.99	0.98	2015-04-04	6ba51774-9316-4103...
6ba51009-9316-1...	0.3	0	0	0.98	0.98	0.98	2015-04-04	087d5974-4175-4c68...

Figura 47: Almacenamiento de las métricas del modelo en entrenamiento del prototipo

La interpretación de estos resultados permitirá determinar si el modelo entrenado tiene los valores mínimos requeridos para la detección de los ataques. A continuación se muestran los valores obtenidos en el entrenamiento del prototipo (Tabla 9):

Métricas de desempeño en fase de Entrenamiento Prototipo		
	Anómalo	Normal
Anómalo	TVP: 98.9%	TFP: 0.2%
Normal	TFN	TVN
Clasificación: 99%	Sensibilidad: 98%	

Tabla 9: Resultado de las métricas de desempeño en entrenamiento del Prototipo

El cuadro anterior nos indica que el porcentaje de clasificación, combinado con la sensibilidad es muy alto, permitiendo tener un excelente nivel de confiabilidad en la detección en la fase de evaluación. Con el modelo generado y almacenado, procedemos a realizar el proceso automático de evaluación, el cual indicará que peticiones se encuentran en un estado normal o anómalo.

3.4.3. EVALUACIÓN DEL PROTOTIPO:

Después del entrenamiento, el 40%(normal 539, anómalo 539) del conjunto de datos de la máquina de aprendizaje se evalúa, tomando como base el modelo generado en la etapa anterior y analizando el porcentaje de clasificación correcta y errada de las instancias. Para realizar la respectiva evaluación, se guarda la información en un archivo y se clasifican los datos automáticamente, colocando como valor por defecto el tráfico normal (Figura 48):

```
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
220838,192168172,192168117,343,24734,normal
220839,192168112,192168117,100,1690,normal
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
210441,192168111,192168113,3,1024,normal
210441,192168111,192168114,2,1168,normal
220838,192168172,192168117,343,24734,normal
220839,192168112,192168117,100,1690,normal
```

Figura 48: Información del archivo de evaluación en pruebas

El parámetro *"normal"* es colocado automáticamente en el archivo normalizado para la fase de evaluación, con el propósito que el prototipo de detección cargue el modelo generado en la etapa de entrenamiento y lo compare con cada una de las peticiones existentes.

3.4.3.1. VALIDACIÓN DE LOS RESULTADOS CON WEKA:

El proceso de validación en WEKA requiere como punto inicial que se cargue el modelo generado en la etapa anterior, como se muestra la siguiente figura (49):

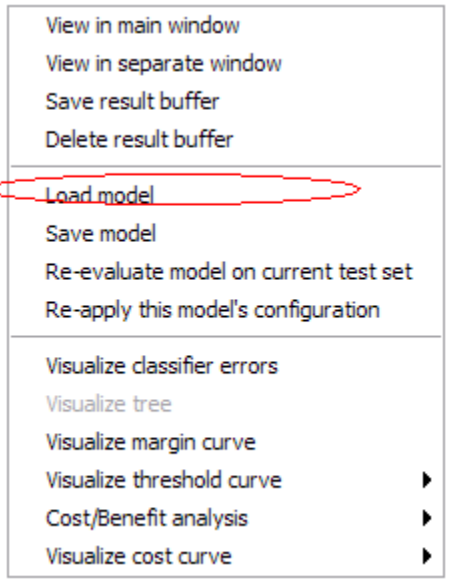


Figura 49: Cargar modelo generado en pruebas

Se carga el modelo y se sube el conjunto de datos de prueba, utilizando la siguiente opción (Figura 50):

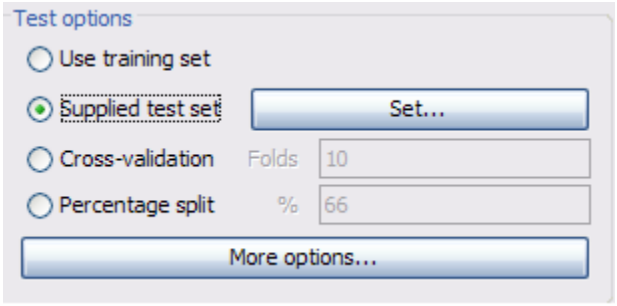


Figura 50: Cargar conjunto de datos de pruebas

Al cargar la información en la herramienta, procedemos a ejecutar la evaluación teniendo en cuenta los parámetros cargados propios del modelo. A continuación se presentan los resultados de las pruebas de evaluación tomando el 40% de la información normalizada con la herramienta WEKA (Figura 51):

```
=== Summary ===

Correctly Classified Instances        1068               99.8131 %
Incorrectly Classified Instances         2                0.1869 %
Kappa statistic                          0.9963
Mean absolute error                      0.0019
Root mean squared error                  0.0432
Relative absolute error                  0.3738 %
Root relative squared error              8.6468 %
Coverage of cases (0.95 level)          99.8131 %
Mean rel. region size (0.95 level)      50      %
Total Number of Instances             1070

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,998    0,002    0,998      0,998    0,998      0,996    0,998     0,997     normal
                 0,998    0,002    0,998      0,998    0,998      0,996    0,998     0,997     anomaly
Weighted Avg.    0,998    0,002    0,998      0,998    0,998      0,996    0,998     0,997

=== Confusion Matrix ===

   a   b   <-- classified as
 536   1 |   a = normal
   1 532 |   b = anomaly
```

Figura 51: Evaluación de la información normalizada en WEKA

A continuación se muestran los valores obtenidos en la evaluación con WEKA (Tabla 10):

Métricas de desempeño en fase de Evaluación WEKA			
	Anómalo	Normal	
Anómalo	TVP: 99.8%	TFP: 0.2%	
Normal	TFN	TVN	
Clasificación: 99.81%	Precisión: 99.8%	Sensibilidad: 99.8%	

Tabla 10: Resultado de las métricas de desempeño en evaluación WEKA

Las métricas de desempeño mostradas en la fase de evaluación, tienen la misma relevancia que en la fase de entrenamiento, ya que nos permiten observar si efectivamente la SVM está clasificando correctamente las peticiones. En este caso la evaluación tiene un buen nivel de clasificación. En la siguiente figura (52) se puede observar la visualización basada en Curvas ROC de la tasa de Falsos Positivos y Verdaderos Positivos (Tasa de Detección):

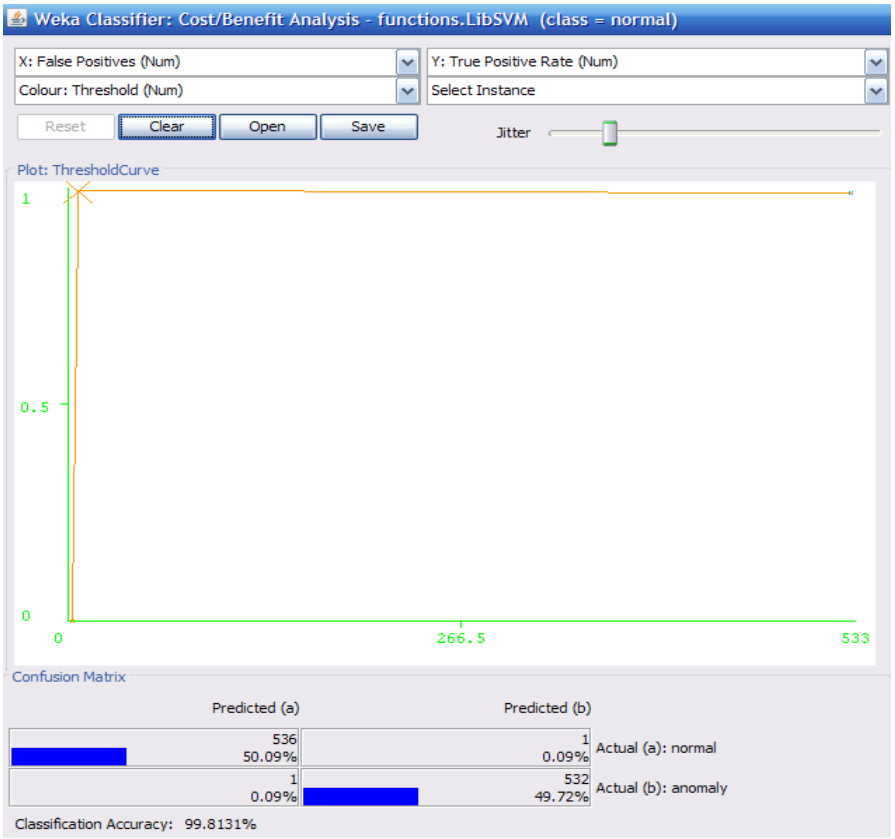


Figura 52: Curva ROC en evaluación WEKA

De la misma manera que en la fase de entrenamiento, podemos observar que en el eje X están los falsos positivos y en el eje Y los verdaderos positivos, si el valor en el eje Y se acerca a 1 y el valor en el eje X se acerca a 0, significa que los eventos tendrán mayor capacidad de detección y por ende, mayor capacidad de discriminación entre un comportamiento normal o anómalo.

3.4.3.2. VALIDACIÓN DE LOS RESULTADOS CON EL PROTOTIPO:

Para la realización de las pruebas del prototipo en la fase de evaluación, primero capturamos la información normalizada y la guardamos en una carpeta configurada en el prototipo (Figura 53):

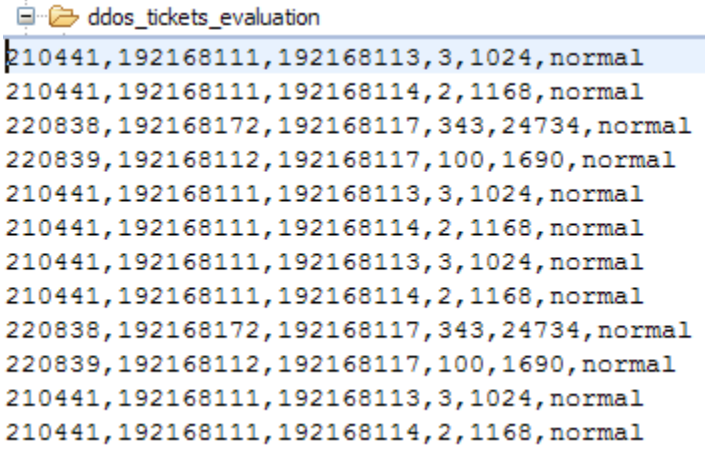


Figura 53: Información normalizada en evaluación del prototipo

Ejecutamos el proceso de evaluación y automáticamente va tomando cada petición de manera individual y comparando contra el modelo generado, mostrando una traza donde se resalta si la solicitud es normal o anómala (Figura 54):

```
INFO: 2014-09-25 22:02:33.453 [com.orionix.ddos.svm.services.impl.ServiceDDos] -
 UUID: d0cb7c4d-d5cc-48a6-aee2-b8ed54dca2aa
Duración: 210441
Dirección Origen: 192168111
Dirección Destino: 192168114
Cantidad: 2
Bytes: 1168
Clase Actual: 1
Actual:NORMAL
Prediction:NORMAL

INFO: 2014-09-25 22:02:33.500 [com.orionix.ddos.svm.services.impl.ServiceDDos] -
 UUID: 830a5737-3d05-4748-8753-7c60406e76c8
Duración: 220838
Dirección Origen: 192168172
Dirección Destino: 192168117
Cantidad: 343
Bytes: 24734
Clase Actual: 1
Actual:NORMAL
Prediction:ANOMALY
```

Figura 54: Información de la traza de la detección en evaluación del prototipo

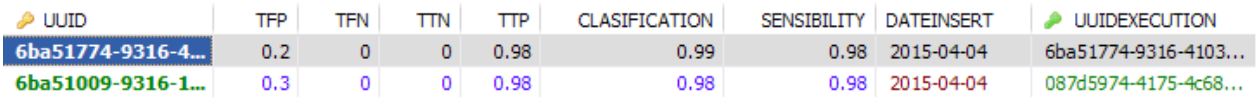
Analicemos la traza anterior, la primera petición fue marcada por el proceso de detección como *normal*, teniendo en cuenta que la cantidad de peticiones en un segundo (2) y la cantidad de información recibida (1168) Bytes. En el segundo caso el proceso de detección marco la petición como *anómala*, basado en la cantidad de peticiones en un segundo (343) y la cantidad de información recibida (24734) Bytes. Cada una de las peticiones anteriormente mostradas es almacenada en una base de datos, teniendo en cuenta la estructura mostrada en la fase de implementación (Figura 55):

DURATION	IP_SRC	IP_DEST	IP_SRC_COUNT	BYTES	CLASS1	ACTUAL	PREDICTION
210441	192168111	192168114	2	1168	1	NORMAL	ANOMALY
220839	192168112	192168117	100	1690	1	NORMAL	ANOMALY

Figura 55: Almacenamiento de las variables operacionales de detección en pruebas

La información anterior puede ser utilizada en futuras versiones del prototipo para que pueda tener una participación activa al bloquear las direcciones IP que presentan este tipo de comportamientos anómalos. El prototipo en evaluación también guarda las estadísticas de

desempeño en la base de datos, según la estructura definida en la fase de implementación para la respectiva interpretación por parte del experto (Figura 56):



UUID	TFP	TFN	TTN	TTP	CLASIFICATION	SENSIBILITY	DATEINSERT	UUIDEXECUTION
6ba51774-9316-4...	0.2	0	0	0.98	0.99	0.98	2015-04-04	6ba51774-9316-4103...
6ba51009-9316-1...	0.3	0	0	0.98	0.98	0.98	2015-04-04	087d5974-4175-4c68...

Figura 56: Almacenamiento de las métricas del modelo en evaluación del prototipo

A continuación se muestran los valores obtenidos en la evaluación del prototipo (Tabla 11):

Métricas de desempeño en fase de Evaluación Prototipo		
	Anómalo	Normal
Anómalo	TVP: 98.9%	TFP: 0.2%
Normal	TFN	TVN
Clasificación: 99%	Sensibilidad: 98%	

Tabla 11: Resultado de las métricas de desempeño en evaluación del prototipo

Las métricas de desempeño mostradas en la tabla anterior (11), nos permiten validar la tasa de clasificación del prototipo e integración de la librería *libsvm* comparada con la herramienta WEKA.

3.4.4. EVALUACIÓN DE LA HERRAMIENTA SNORT:

Con el objetivo de realizar la comparación entre una técnica que utiliza un motor de reglas como SNORT, contra otra técnica que utiliza máquinas de aprendizaje SVM, se tomó como base la arquitectura física planteada en la Figura (29) y los datos recopilados en la sección (*3.4.1 Recopilación de la Información*) para la evaluación (normal 539, anómalo 539) .

Al tener instalada la herramienta SNORT en la máquina donde vamos a realizar el experimento, ejecutamos el comando que nos permite enlazar la interfaz de red con el aplicativo (Figura 57):

```
orionix@orionix-desktop:~/Escritorio$ sudo snort -i eth0
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.6.0 GRE (Build 47)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.5.3
           Using PCRE version: 8.31 2012-07-06
           Using ZLIB version: 1.2.8
```

Figura 57: Comando de inicio SNORT

En el comando anterior la herramienta enlaza la interfaz de red (eth0) y captura todo el tráfico entrante para realizar el respectivo análisis. Después ingresamos a la configuración de las reglas que tiene SNORT para detectar ataques DDOS (Figura 58), las cuales se encuentran ubicadas en la ruta (*/etc/snort/rules/ddos.rules*):

```
attack-responses.rules          community-nntp.rules           deleted.rules       netbios.rules     sql.rules
backdoor.rules                  community-oracle.rules         dns.rules           nntp.rules        telnet.rules
bad-traffic.rules               community-policy.rules         dos.rules           oracle.rules      tftp.rules
chat.rules                      community-sip.rules            experimental.rules  other-ids.rules   virus.rules
community-bot.rules             community-smtp.rules           exploit.rules       p2p.rules         web-attacks.rules
community-deleted.rules         community-sql-injection.rules  finger.rules        policy.rules      web-cgi.rules
community-dos.rules             community-virus.rules          ftp.rules           pop2.rules        web-client.rules
community-exploit.rules         community-web-attacks.rules    icmp-info.rules     pop3.rules        web-coldfusion.rules
community-ftp.rules             community-web-cgi.rules        icmp.rules          porn.rules        web-frontpage.rules
community-game.rules            community-web-client.rules     imap.rules          rpc.rules         web-iis.rules
community-icmp.rules            community-web-dos.rules        info.rules          rservices.rules   web-misc.rules
community-imap.rules            community-web-iis.rules        local.rules         scan.rules        web-php.rules
community-inappropriate.rules   community-web-misc.rules       misc.rules          shellcode.rules   x11.rules
community-mail-client.rules     community-web-php.rules        multimedia.rules    smtp.rules
community-misc.rules            ddos.rules                     mysql.rules         snmp.rules
orionix@orionix-desktop:/etc/snort/rules$ vi ddos.rules
orionix@orionix-desktop:/etc/snort/rules$ pwd
/etc/snort/rules
orionix@orionix-desktop:/etc/snort/rules$
```

Figura 58: Ruta reglas SNORT

Posteriormente observamos la estructura de las reglas configuradas en el archivo *ddos.rules* para detectar este tipo de comportamientos (Figura 59):

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DDOS TFN Probe"; icmp_id:678; itype:8; content:"1234"; reference:arachnids,443; classtype:a
ttempted-recon; sid:221; rev:4;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DDOS tfn2k icmp possible communication"; icmp_id:0; itype:0; content:"AAAAAAAAAA"; referenc
e:arachnids,425; classtype:attempted-dos; sid:222; rev:2;)
alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"DDOS Trin00 Daemon to Master PONG message detected"; content:"PONG"; reference:arachnids,1
87; classtype:attempted-recon; sid:223; rev:3;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DDOS TFN client command BE"; icmp_id:456; icmp_seq:0; itype:0; reference:arachnids,184; cla
sstype:attempted-dos; sid:228; rev:3;)
```

Figura 59: Reglas configuradas en SNORT

En las reglas anteriores se debe especificar el protocolo de comunicación, la interfaz externa, interfaz remota y el contenido del mensaje que analizará el sistema de detección de intrusos. Es importante aclarar que en el presente experimento las reglas no serán modificadas y se analizará la tasa de detección teniendo como base los valores que trae por defecto la herramienta para un ataque DDOS. Al tener todos los elementos necesarios para realizar las pruebas, lanzamos el generador de ataques (Figura 60) presentados en la sección (*3.3.1 Prototipo generador de ataques DDOS*):

```
INFO: 2015-04-19 19:38:10,906 - [com.orionix.ddos.client.utility.Propiedades - main] - Leyendo propiedades de aplicación...
INFO: 2015-04-19 19:38:10,921 - [com.orionix.ddos.client.utility.Propiedades - main] - Carga exitosa propiedades de aplicación
INFO: 2015-04-19 19:38:10,937 - [com.orionix.ddos.client.Main - main] - Iniciando el aplicativo DDOSCliente...
INFO: 2015-04-19 19:38:10,937 - [com.orionix.ddos.client.Main - main] - Leyendo los parametros de conexion del Cliente DDOS...
INFO: 2015-04-19 19:38:10,937 - [com.orionix.ddos.client.Main - main] - Host: http://192.168.2.2
INFO: 2015-04-19 19:38:10,937 - [com.orionix.ddos.client.Main - main] - Puerto: 80
INFO: 2015-04-19 19:38:10,937 - [com.orionix.ddos.client.Main - main] - Tiempo de Espera: 50000
```

Figura 60: Log generador de ataques SNORT

Al realizar el ataque con la información de evaluación, el log de SNORT muestra las peticiones enviadas desde la IP origen *192.168.2.3* a la IP destino del servidor 192.168.2.2, como se muestra en la figura (61):

```
04/19-19:52:10.489223 192.168.2.2:5901 -> 192.168.2.3:2311
TCP TTL:64 TOS:0x0 ID:16604 IpLen:20 DgmLen:1300 DF
***A**** Seq: 0x3151315E  Ack: 0xE2CEEF19  Win: 0x117  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+

04/19-19:52:10.489237 192.168.2.2:5901 -> 192.168.2.3:2311
TCP TTL:64 TOS:0x0 ID:16605 IpLen:20 DgmLen:1300 DF
***A**** Seq: 0x3151364A  Ack: 0xE2CEEF19  Win: 0x117  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+
```

Figura 61: Log peticiones capturadas por SNORT

Después de recuperar la información del tráfico a través de SNORT, se calculan las métricas de desempeño manualmente a través de las fórmulas matemáticas presentadas en la sección (*3.4.1 Recopilación de la Información*) y generaron los siguientes datos (Tabla 12):

Métricas de desempeño en pruebas SNORT		
	Anómalo	Normal
Anómalo	TVP: 88.9%	TFP: 12.2%
Normal	TFN	TVN
Clasificación: 89%	Sensibilidad: 88%	

Tabla 12: Resultado de las métricas de desempeño con SNORT

Las métricas de desempeño mostradas en la tabla anterior (12), nos permiten validar la tasa de clasificación en una herramienta como SNORT.

3.5. ANÁLISIS DE RESULTADOS

Los resultados presentados en la etapa de pruebas, permitieron observar y analizar las variables más relevantes en el proceso, tales como: nivel de clasificación correcto, sensibilidad, precisión, falsos positivos, entre otros, tomando como base los 2698 registros entre tráfico normal y anómalo divididos en la etapa de entrenamiento en un 60% del conjunto de datos (normal 809, anómalo 809) y el otro 40% (normal 539, anómalo 539) para la evaluación.

En la etapa de entrenamiento del prototipo (Tabla 13) se pudo observar que el porcentaje de clasificación correcto tiene un valor bastante alto (99%), la cantidad de falsos positivos es bajo (0.2%), la sensibilidad bastante alta (98%). Estos resultados nos indican que la información de 1618 instancias clasificadas como normales y anómalas permiten tener un excelente nivel de confiabilidad en la generación del modelo, teniendo como base los porcentajes de clasificación correcto, la sensibilidad y el bajo porcentaje de falsos positivos del modelo (0.2%).

Métricas de desempeño en fase de Entrenamiento Prototipo		
	Anómalo	Normal
Anómalo	TVP: 98.9%	TFP: 0.2%
Normal	TFN	TVN
Clasificación: 99%	Sensibilidad: 98%	

Tabla 13: Resultado de las métricas de desempeño en entrenamiento resultados

En la etapa de evaluación del prototipo (Tabla 14) se pudo observar que el porcentaje de clasificación correcto tiene un valor bastante alto (98.9%), la cantidad de falsos positivos es bajo (0.2%), la sensibilidad bastante alta (98%). Estos resultados nos indican que la información de 1078 instancias clasificadas como normales y anómalas permiten tener un excelente nivel de confiabilidad en la detección del ataque, teniendo como base los porcentajes de clasificación correcto, la sensibilidad y el bajo porcentaje de falsos positivos del modelo (0.2%).

Métricas de desempeño en fase de Evaluación Prototipo		
	Anómalo	Normal
Anómalo	TVP: 98.9%	TFP: 0.2%
Normal	TFN	TVN
Clasificación: 99%	Sensibilidad: 98%	

Tabla 14: Resultado de las métricas de desempeño en evaluación resultados

Comparación de la máquina de aprendizaje SVM vs IDS CONVENCIONAL

El prototipo implementado con la técnica SVM tuvo una mejoría significativa (Aproximadamente un 10%) en la precisión de la clasificación cuando se comparó con el sistema de detección de intrusos SNORT utilizando las reglas establecidas por los expertos humanos.

El modelo estadístico interno de SVM, permitió identificar los datos anómalos o normales en la fase de evaluación sin necesidad de adicionar o modificar las reglas como sucede con SNORT, haciendo que el proceso de detección de intrusos se realice de manera rápida, automática y sin interacción humana; sin embargo, es importante resaltar que la técnica SVM requiere un nivel de entrenamiento y normalización con un flujo de información alto entre casos (Normales-Anómalos), con el propósito que la evaluación sea efectiva.

El desarrollar cada una de las etapas de la metodología y confirmar los resultados a través de un experimento, permitió a través de este documento plasmar claramente los beneficios de utilizar e integrar las técnicas basadas en SVM a un prototipo computacional que permitiera aportar a la solución definitiva de este comportamiento anómalo.

4. CONCLUSIONES

Al desarrollar una investigación que implique la utilización de máquinas aprendizajes, es necesario definir el alcance y los objetivos específicos del proyecto, ya que son los elementos claves para la selección de la técnica y las variables operacionales, las cuales determinarán la adecuada clasificación del producto desarrollado.

El diseño e implementación del prototipo a un nivel empresarial, requiere un manejo optimizado de los recursos de hardware en las etapas de filtro y normalización de la información, debido al alto volumen de datos capturado por cada una de las peticiones realizadas por los usuarios.

Es realmente sencillo hacer un desarrollo que permita realizar un ataque DDOS a un servidor WEB con certificados de seguridad o sin ellos, lo complejo es detectar qué peticiones son normales o anómalas, teniendo en cuenta el gran flujo de información.

El proceso de entrenamiento y afinamiento de la máquina de aprendizaje a partir del conjunto de datos normalizados, son la base para que el modelo generado tenga un porcentaje de clasificación aceptable en el momento de realizar la evaluación de prototipo en un entorno productivo y con información real. La selección del conjunto de métricas en el problema de detección de intrusos: Tasa de Falsos Positivos, Tasa de Falsos Negativos, Tasa de Clasificación, curvas ROC, permiten tener un estándar de comparación contra otros modelos.

La aplicación de técnicas con un modelo supervisado como SVM, tiene amplias ventajas en comparación con la técnica basada en reglas, ya que la generación del modelo se basa en un modelo estadístico que cambia su comportamiento según los parámetros de entrada definidos en el entrenamiento y el basado en reglas requiere la interacción humana.

En la evaluación del prototipo se pudo constatar que el correcto porcentaje de clasificación de peticiones normales o anómalas en la fase de entrenamiento, está directamente relacionado con la normalización y adecuada selección de los parámetros de entrada, permitiendo que las variables de salida se generen con un porcentaje mínimo de error de clasificación, generando confianza en el modelo generado y en la detección de estos comportamientos.

5. RECOMENDACIONES

La cantidad de parámetros a tener en cuenta en la implementación de un prototipo computacional que detecte este tipo de comportamientos anómalos, hace que el alcance se tenga que limitar para poder lograr los objetivos trazados del proyecto, por ese motivo, se realizan las siguientes recomendaciones a los investigadores que desean continuar aportando a la solución de esta problemática:

- Antes de comenzar a diseñar e implementar el prototipo se deben entender y evaluar las diferentes herramientas y técnicas de ataque en el momento de la investigación, debido a que constantemente los atacantes buscan nuevas formas de evadir los controles expuestos por los sistemas de detección.
- El prototipo implementado tienen variables operacionales como la dirección IP Origen y Destino, que permiten detectar la cantidad de peticiones en un período determinado desde esa dirección; sin embargo, es necesario adaptar el prototipo para que detecte las direcciones IP que se enmascaran a través de un servidor PROXY o VPN[21], haciendo que el flujo se incremente drásticamente y se considere como un ataque, cuando realmente son diferentes máquinas utilizando la misma dirección IP.
- El tener un método supervisado permite que el prototipo tenga un entrenamiento previo a la puesta en producción; sin embargo, es importante hibridar la técnica SVM con un método no supervisado en las entradas del prototipo, permitiendo ajustar los valores en ejecución a los diferentes tipos de ataque que cambian en el tiempo.
- Evaluar la posibilidad que el filtro y la normalización se realicen a través de hilos de ejecución con el objetivo de mejorar el rendimiento, teniendo en cuenta que una petición WEB realizada por un cliente a un servidor, son múltiples peticiones de solicitudes que se visualizan cuando se captura el tráfico con el comando TCPDUMP.
- Complementar el prototipo con los requerimientos no funcionales que va a tomar como base para la detección de ataques DDOS en un entorno empresarial, por ejemplo: tiempos de respuesta, disponibilidad, integridad, confidencialidad.

[21] Virtual Private Network

6. REFERENCIAS

Analysis, W. E. (s.f.). *Weka*. Recuperado el 11 de 02 de 2015, de http://www.cs.waikato.ac.nz/ml/weka/

Bellovin, S. (1989). Security problems in the TCP/IP protocol suite. *19*(2), Pág 32-48.

Bonilla, C., Isaza, G., & Duque, N. (2008). Sistema de Detección de Intrusos Neuronal. Tendencias en Ingeniería de Software e Inteligencia Artificial. *Universidad Nacional*.

Brugger, S., & Chow, J. (2007). An assessment of the DARPA IDS Evaluation Dataset using Snort. . U. o. C. Technical Report CSE-2007-1.

Castellano, M., Mastronardi, G., Aprile, A., & Bellone, G. (2007). Applying a Flexible Mining Architecture to Intrusion Detection The Second International Conference on Availability, Reliability and Security (ARES'07). Pág 845,852.

CERP. (11 de 1999). Obtenido de Center Results of the distributed systems intruder tools: http://www.cert.org/reports/dsit-workshop.pdf

Chan, A., Ng, W., Yeung, D., & Tsang, E. C. (2004). "Refinement of rule-based intrusion detection system for denial of service attacks by support vector machine," Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on. *Vol 7*, Pág 4252,4256.

Cio. (15 de 01 de 2015). *Cioal*. Recuperado el 13 de 03 de 2015, de http://www.cioal.com/2015/01/15/ataques-ddos-dejan-corea-del-norte-fuera-de-internet/

Daemon9, I. a.-s.-r. (6 de 1996). Obtenido de http://www.fc.net/phrack/files/p4.8/p48-14.html

DARPA, D. A., & AFRL/SNHS, A. F. (s.f.). Recuperado el 04 de 11 de 2013, de Massachusetts Institute of Tecnology: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/index.html

Debar, H., & Dorizzi, B. (1992). An Application of a Recurrent Network to an Intrusion Detection System. *Proceedings of the International Joint Conference on Neural Networks*, Pág 78,483.

Douligeris, C., & Mitrokotsa, A. (2003). Ddos attacks and defense mechanisms: A classification, in Signal Processing and Information Technology. *ISSPIT*, Pág 190,193.

Echeverri, G. A., O., L. F., Trujillo, M. L., & Marulanda, C. E. (2010). Modelo híbrido de neuroclasificación y clustering en el problema de detección de intrusiones. *Vector*, Pág 69,77.

Feinstein, L., Schnackenberg, D., Balupari, R., & Kindred, D. (2003). Statistical approaches to ddos attack detection and response. *1*, Pág 303,314.

IDWG. (s.f.). *Intrusion Detection Working Group-Intrusion*. Recuperado el 2014, de http://www.ietf.org/rfc/rfc4766.txt

InfoSec. (29 de 10 de 2013). Recuperado el 15 de 01 de 2015, de InfoSec: http://resources.infosecinstitute.com/dos-attacks-free-dos-attacking-tools/

Isaza, G., Perez, C., & Brito, J. (2007). Detección de Ataques en un entorno conectado con Redes Neuronales. Memorias Flisol 2006. Pág 91,102.

Kartalopoulos, S. (1996). *Understanding Neural Networks and Fuzzy Logic:Basic Concepts and Applications* (1 ed.). Wiley-IEEE Press.

Laskov, P., Dussel, P., Schafer, C., & Rieck, K. (2005). Learning intrusion detection: Supervised or unsupervised? In Image Analysis and Processing - ICIAP, 13th International Conference Cagliari. Pág 50,57.

Lau, F., Rubin, S., Smith, M., & Trajkovic, L. (2000). "Distributed denial of service attacks," Systems, Man, and Cybernetics, 2000 IEEE International Conference on. *Vol 3*, Pág 2275-2280.

Lee, W., & Stolfo, S. (1998). Data Mining Approaches for Intrusion Detection. *7th USENIXSecurity Symposium*.

Li, J., Liu, Y., & Gu, L. (1-4 de 11 de 2010). "DDoS attack detection based on neural network," Aware Computing (ISAC), 2010 2nd International Symposium on. Pág 196-199.

Li, K., & Teng, G. (2006). Unsupervised SVM Based on p-kernels for Anomaly Detection. Proceedings of the First International Conference on Innovative Computing, Information and Control - Volume 2. *IEEE Computer Society, 2*.

Li, Q., Chang, E.-C., & Chan, M. C. (13-17 de 3 de 2005). On the effectiveness of DDoS attacks on statistical filtering. *2*, Pág 1373,1383.

Lin, H.-J., Yang, F.-W., & Kao, Y.-T. (2005). An efficient Genetic Algorithm based Clustering. *Tamkang journal of science and engineering, 8*, Pág 113,122.

Liu, W.-T. (12 de 07 de 2008). Research on intrusion detection rules based on XML in distributed IDS," Machine Learning and Cybernetics, 2008 International Conference on. *Vol.3*, Pág 1400-1403.

Liu, Y., Chen, K., Liao, X., & Zhang, W. (2004). A Genetic Clustering Method for Intrusion Detection. *Journal of Pattern Recognition, 37*, Pág 927,942.

Lu, L. F., Huang, M. L., Orgun, M., & Zhang, J. W. (2010). An Improved Wavelet Analysis Method for Detecting DDoS Attacks. *Network and System Security (NSS), 2010 4th International Conference on*, Pág 318,322.

Luo, J., & Bridges, S. M. (2000). Mining Fuzzy Association Rules and Fuzzy Frequency Episodes for Intrusion Detection. *International Journal of Intelligent Systems, 15*(8), Pág 687,704.

Moore, D., Voelker, G., & Savage, S. (2001). Inferring Internet Denial of Service Activity. *Proceedings of the 2001 USENIX Security Symposium*.

Mukkamala, S., & Sung, A. (2003). "Detecting denial of service attacks using support vector machines," Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on. *Vol 2*, Pág 1231-1236.

Muthuregunathan, R., Siddharth, S., Srivathsan, R., & Rajesh, S. (23-25 de Julio de 2009). "Efficient Snort Rule Generation Using Evolutionary Computing for Network Intrusion Detection," Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on. Pág 336-341.

Nachif Fernandes, S., Pilastri, A., Pereira, L., Goncalves Pires, R., & Papa, J. (26-30 de Agosto de 2014). Learning Kernels for Support Vector Machines with Polynomial Powers of Sigmoid. Pág 259-265.

Ng, W., Chang, R., & Yeung, D. (2003). Dimensionality reduction for denial of service detection problems using RBFNN output sensitivity. *Conf. of Machine Leaming and Cybernetics*, Pág 1293,1298.

Perez, C., Isaza, G., & Brito, J. (2005). Aplicación de Redes Neuronales para la detección de intrusos en redes y sistemas de información. *11*, Pág 225,230.

Roy, R. (2001). *Industrial Knowledge Management: A Micro-Level Approach.* Springer.

S. Floyd, V. J. (8 de 1995). Link-sharing and resource management models for packet networks. *3*(4), Pág 365,386.

Sanchez, S., Henao, C. P., & Mauricio, A. (Mayo de 2007). Evaluación de Algoritmos de Detección de Complejos QRS mediante las curvas de funcionamiento ROC, DET y EPC. (34), Pág 43-47.

Security, I. (05 de 11 de 2010). *Info Security*. Recuperado el 10 de 12 de 2013, de http://www.infosecurity-magazine.com/view/13762/massive-ddos-attack-knocks-burma-offline/

Seufert, S., & O' Brien, D. (24-28 de Junio de 2007). "Machine Learning for Automatic Defence Against Distributed Denial of Service Attacks," Communications, 2007. ICC '07. IEEE International Conference on. Pág 1217-1222.

Shawe-Taylor, J., & Cristianini, N. (2000). *Support Vector Machines and other kernel-based learning methods.* (C. U. Press, Ed.) UK.

Shon, T., Kim, Y., Lee, C., & Moon, J. (15-17 de 06 de 2005). A machine learning framework for network anomaly detection using SVM and GA. Pág 176,183.

Snort. (2008). *Snort.* Recuperado el 2014, de http://www.snort.org

Soman, K., Loganathan, R., & Ajay, V. (2009). *Machine Learning with SVM and other Kernel Methods.*India: PHI.

Subbulakshmi, T., Shalinie, S., GanapathiSubramanian, V., BalaKrishnan, K., AnandKumar, D., & Kannathal, K. (14.16 de Diciembre de 2011). Detection of DDoS attacks using Enhanced Support Vector Machines with real time generated dataset," Advanced Computing (ICoAC), 2011 Third International Conference on. Pág 17-22.

Thing, V., Sloman, M., & Dulay, N. (9 de 2009). Locating network domain entry and exit point/path for DDoS attack traffic. *Network and Service Management, IEEE Transactions on, 6*(3), Pág 163,174.

Toure, M. (1994). An interdisciplinary approach for adding knowledge to computer security systems. *Institute of Electrical and Electronics Engineers 28th Annual 1994 International Carnahan Conference on*, Pág 158,168.

Vijaya, M., Jamuna, K., & Karpagavalli, S. (28-29 de Diciembre de 2009). Password Strength Prediction Using Supervised Machine Learning Techniques," Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT '09. International Conference on. Pág 401-405.

Wang, B., & Schulzrinne, H. (2003). Analysis of denial-of-service attacks on denial-of-service defensive, measures. *IEEE Proc. of Global Telecommunications Conference, 3*.

Witten, I. H., Frank, E., & Hall, M. A. (2005). *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.). Elsevier.

Xiao, H., Hong, F., Zhang, Z., & Liao, J. (2007). Intrusion Detection Using Ensemble of SVM Classifiers. Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007) Vol.4 - Volume 04. *IEEE Computer Society, 4*.

Xie, Y., & Yu, S.-z. (2006). A Novel Model for Detecting Application Layer DDoS Attacks. *Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on*, Pág 56,63.

Yu, J., Li, Z., Chen, H., & Chen, X. (19-25 de 6 de 2007). A Detection and Offense Mechanism to Defend Against Application Layer DDoS Attacks. *Networking and Services, 2007. ICNS. Third International Conference on*, Pág 54.

Yu, Y., Wei, Y., Fu-Xiang, G., & Ge, Y. (2006). Anomaly Intrusion Detection Approach Using Hybrid MLP/CNN Neural Network. *IEEE Intelligent Systems Design and Applications, 2*, Pág 1095,1102.

Zhang, L.-h., & Duan, H.-x. (2008). Design and Implementation of DDoS Attack-defense Testbed. *34*(13).

Zhi, P., C. SongCan, G.-B. H., & Zhang, D.-G. (2003). Hybrid neural network and C4.5 for misuse detection. Proceedings of the Second International Conference on Machine Learning and Cybernetics. Pág 2463,2467.