PANDA (PROCESSING AND ANALYSIS OF NETWORK DATA) PARA LA DETECCIÓN DE ATAQUES SYN-FLOOD Y SATAN SOBRE FLUJOS CONTINUOS DE PAQUETES DE RED MEDIANTE EL ALGORITMO DBSCAN

LUZ ADRIANA PINTO DÍAZ

RICARDO SANTOS DÍAZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS

BUCARAMANGA

2013

PANDA (PROCESSING AND ANALYSIS OF NETWORK DATA) PARA LA DETECCIÓN DE ATAQUES SYN-FLOOD Y SATAN SOBRE FLUJOS CONTINUOS DE PAQUETES DE RED MEDIANTE EL ALGORITMO DBSCAN

LUZ ADRIANA PINTO DÍAZ

RICARDO SANTOS DÍAZ

Trabajo de grado para optar al título de

Ingeniero(a) de Sistemas

Director

MSc. LOLA X. BAUTISTA ROZO

Codirector

MSc. PEDRO J. TRUJILLO

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS

BUCARAMANGA

2013

AGRADECIMIENTOS

A Dios, porque después de todo estamos aquí escribiendo esto.

A nuestros padres por no perder la fé.

A nuestros hermanos por fregarnos la vida.

A nuestros profesores por la paciencia y la voluntad de ayudarnos a aprender.

A nuestros amigos por hacer de este paso emocionante.

A la universidad, nuestro segundo hogar.

A la vida, que nos puso juntos aquí y nos puso juntos lejos de aquí.

A ustedes que leen esto, por ser parte del último capítulo de esta etapa.

Gracias

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

ANOMALÍA DE RED: comportamiento atípico del tráfico de red.

ATAQUE DE DENEGACIÓN DE SERVICIOS (DoS): es un ataque que causa que un recurso o servicio de red sea inaccesible a los usuarios legítimos.

INTRUSION: violación de la política de seguridad de un sistema.

SISTEMA DE DETECCIÓN DE INTRUSIONES (IDS): herramienta software para el monitoreo de eventos que suceden en una red en busca de signos de intrusiones.

DARPA'98: conjunto de datos de tráfico de red simulado por DARPA (Defense Advanced Research Projects Agency) en el año 1998.

ENTROPÍA: medida del desorden de un sistema.

SNIFFER: es un programa de captura de tramas de una red de computadores.

CLUSTERING: es una técnica de minería de datos descriptiva que busca identificar un *set* finito de categorías o clústeres para describir un conjunto de datos.

DBSCAN: es un algoritmo de *clustering* basado en densidad.

SYN-FLOOD: es el ataque de denegación de servicio más frecuente.

SATAN: es un ataque de escaneo de vulnerabilidades en máquinas remotas.

COMPONENTE: que compone o entra en la composición de un todo.

RESUMEN

Título: PANDA (PROCESSING AND ANALYSIS OF NETWORK DATA) PARA LA DETECCIÓN DE ATAQUES SYN-FLOOD Y SATAN SOBRE FLUJOS CONTINUOS DE PAQUETES DE RED MEDIANTE EL ALGORITMO DBSCAN.*

Autores: Luz Adriana Pinto Díaz, Ricardo Santos Díaz.**

Palabras Clave: DoS, SYN-FLOOD, Probing, Intrusiones de red, Minería de Datos.

Descripción: Este trabajo presenta un modelo para la detección de anomalías de red enfocándose en los ataques de denegación de servicios SYN-FLOOD y SATAN de escaneo de vulnerabilidades por medio del análisis y procesamiento de flujos continuos de paquetes de red. La arquitectura del modelo contempla los componentes de: captura del tráfico de red, pre-procesamiento de las características más relevantes para la detección de los ataques SYN-FLOOD y SATAN de acuerdo a diversos trabajos para el dominio mencionado, caracterización de las conexiones anómalas por medio del algoritmo de agrupamiento o clustering de minería de datos DBSCAN basado en densidad y finalmente visualización de los resultados en una interfaz web, la cual genera una respuesta pasiva mediante una notificación en el escritorio para advertir al administrador de la red la ocurrencia de un ataque.

Para la validación del modelo se simularon flujos continuos de paquetes de red con los datos de DARPA'98 los cuales presentan diversos tipos de ataques entre ellos los de interés de este trabajo.

El modelo presentado es una base para un sistema de detección de anomalías capaz de detectar otro tipo de ataques de red mediante la extracción y análisis de otras características presentes en el tráfico de red.

* Proyecto de grado.

** Facultad de ingenierías Físico-mecánicas. Escuela de Ingeniería de Sistemas. Director: Lola Xiomara Bautista. Codirector: Pedro Javier Trujillo.

ABSTRACT

Title: PANDA (PROCESSING AND ANALYSIS OF NETWORK DATA) TO DETECT SYN-FLOOD AND SATAN ATTACKS OVER CONTINUOUS STREAMS OF NETWORK PACKETS THROUGH DBSCAN ALGORITHM.*

Authors: Luz Adriana Pinto Díaz, Ricardo Santos Díaz.**

Keywords: DoS, SYN-FLOOD, Probing, Network Intrusion Detection, Data Mining.

Description: This work presents a model for anomaly detection focusing on denial of service attack SYN-FLOOD and vulnerability scanning attack SATAN through analysis and processing over continuous streams of network packets. The architecture of the model leads to the development of the next components: the network traffic capture, pre-process the most relevant features for detecting SYN-FLOOD and SATAN attacks according to various works for the mentioned domain, the characterization of anomalous connections through the clustering data mining density based algorithm DBSCAN and finally display the results in a web interface, which generates an passive response via desktop notification to alert the network administrator the occurrence of an attack.

To validate the model, were simulated continuous streams of network packets with DARPA'98 dataset which presents various kinds attacks including those of interest for this work.

The model presented is a basis for an intrusion detection system, capable of detecting other types of network attacks by the extraction and analysis of other features in network traffic.

* Grade Work.

** Faculty of Engineering Physique-Mechanics School of Systems Engineering and Informatics. Director: Lola Xiomara Bautista Roso. Co-director: Pedro Javier Trujillo.

INTRODUCCIÓN

El estado actual de las redes computacionales, la multiplicidad y métodos de ataque representan un problema inminente en la seguridad de la comunicación y los servicios a través de la red. Ataques de denegación de servicio DoS buscan impedir que los usuarios legítimos de un sistema puedan acceder a servicios de red agotando los recursos computacionales de la víctima, siendo SYN-FLOOD uno de los ataques más frecuentes y devastadores, el cual mediante el envío de múltiples peticiones de conexión SYN en cortos períodos de tiempo consumen los recursos del servidor de tal forma que este no pueda responder adecuadamente al tráfico legítimo. Otra de las amenazas que con frecuencia se presenta en las redes computacionales es el escaneo de puertos y vulnerabilidades mediante probing, el cual hace uso de peticiones a través de diversos protocolos a nivel de aplicación, para buscar vulnerabilidades en la víctima.

Los precedentes expuestos crean la necesidad de herramientas capaces de detectar intrusiones y escaneos a nivel de transporte de forma ágil sin afectar la operación normal del sistema. En este trabajo se propone un modelo para la detección de anomalías de red enfocándose en los ataques de denegación de servicios SYN-FLOOD y SATAN de escaneo de vulnerabilidades, basado en el análisis de las características más relevantes expuestas en diversos trabajos para el dominio citado. El enfoque que se propone incluye desde la captura de los datos hasta la visualización de los resultados en una interfaz web y usa una implementación del algoritmo basado en densidad DBSCAN para el agrupamiento y caracterización de las conexiones anómalas.

Este documento está organizado de la siguiente manera: en la primera sección la introducción al trabajo realizado, en la segunda sección se resumen los conceptos estudiados en este trabajo, posteriormente en la tercera sección se sintetizan métodos de detección de intrusiones en red que hacen uso del algoritmo

DBSCAN, luego, en la cuarta sección se presenta la propuesta para la detección de intrusiones el cual hace énfasis en la detección de los ataques SYN-FLOOD y SATAN y la metodología utilizada en la elaboración de este trabajo. Los resultados se muestran en la quinta sección y finalmente en la sexta sección se exponen las conclusiones y recomendaciones para el desarrollo de futuras investigaciones.

OBJETIVOS

Objetivo general:

Crear un modelo que analice y procese flujos continuos de paquetes de red mediante el algoritmo de minería de datos DBSCAN para la detección de los ataques SYN-FLOOD y SATAN.

Objetivos específicos:

- Realizar la captura del tráfico de red mediante un *sniffer*.
- Establecer los criterios cualificables y cuantificables necesarios para la identificación de los ataques SYN-FLOOD y SATAN.
- Caracterizar las conexiones anómalas mediante el algoritmo de agrupamiento basado en densidad DBSCAN.
- Desarrollar una interfaz usable y portable para la visualización y abstracción de los resultados
- Integrar mediante el desarrollo de un software las tareas de captura, pre-procesamiento, análisis y visualización de las conexiones identificadas como anómalas.

JUSTIFICACIÓN

Definición del problema: Actualmente la comunicación ininterrumpida a través de internet mediante el uso de dispositivos informáticos se han convertido en una necesidad para las personas y las organizaciones. Con la creciente importancia de la conexión a internet, los sistemas basados en red se han convertido en un objetivo frecuente para grupos que buscan generar pérdidas en información o servicios de red. Tradicionalmente esta problemática es atacada mediante herramientas software denominadas IDS o sistemas de detección de intrusiones, los cuales funcionan mediante métodos basados en firmas de ataques conocidos, las cuales son proveídas por expertos humanos. Una importante limitación en este tipo de método es la dificultad para detectar nuevos ataques, ya que, cada vez que un nuevo ataque es descubierto se deben implementar y distribuir nuevas firmas prolongando la vulnerabilidad de la red y la información mientras se espera la actualización. Estos precedentes crean la necesidad de modelos con la capacidad de detectar ataques automáticamente durante el transcurso del flujo de red, puesto que, almacenar los datos de las conexiones representa un reto debido a la limitación de memoria en disco.

En diversos estudios a nivel global se reporta que el ataque DoS SYN-FLOOD es el ataque con mayor incidencia y efectos devastadores sobre la víctima. En el año 2012 24.54% de los ataques de DDoS a la red a nivel mundial fueron ataques de tipo SYN-FLOOD, presentándose un incremento del 1.75% comparado con el año 2011.

Varios sistemas de detección de intrusiones de código abierto y comercial basados en firmas, han sido comparados para evaluar la capacidad de detectar ataques tipo DoS y probing encontrándose que la mitad de los sistemas evaluados detectaban ataques DoS con una tasa del 50% y la otra mitad de los sistemas evaluados detectaban menos del 50% de los ataques DoS. En cuanto a los

ataques tipo probing sólo uno de los sistemas evaluados detectó un 60% de los ataques y los demás detectaron un 40% de los ataques presentes en los datos. Es necesario entonces un modelo que permita la detección automática de intrusiones en cortos períodos de tiempo, con bajas tasas de falsos positivos para atacar eficazmente las problemáticas de seguridad de red.

Impacto esperado: se espera que la herramienta PANDA sea utilizada por administradores de red para la detección temprana de los ataques SYN-FLOOD y SATAN de modo que se puedan tomar las medidas necesarias para evitar la ejecución del ataque, consiguiendo que la seguridad de la red no se vea afectada.

En cuanto a investigación el modelo presenta una base extensible y escalable para futuros proyectos en sistemas de detección de ataques de red, que permitan la detección de otro tipo de ataques, con el fin de evitar la pérdida de información valiosa y la interrupción de servicios basados en red.

1. MARCO TEÓRICO

En este capítulo se abarca la base teórica necesaria para comprender el desarrollo del modelo de detección de intrusiones de red planteado en este trabajo. Se inicia con el concepto de minería de datos y la técnica utilizada para el agrupamiento, luego se exploran las capas del modelo TCP/IP con énfasis en la capa de transporte para el entendimiento del funcionamiento de la comunicación en redes de computadores. Posteriormente se detalla el comportamiento de los ataques SYN-FLOOD y SATAN los cuales son los identificados por la herramienta desarrollada y finalmente se explica el concepto de sistema de detección de intrusiones y se muestran algunas herramientas a modo de ejemplo.

1.1. MINERÍA DE DATOS

La proliferación de las tecnologías de la información como soporte operativo en organizaciones, comunidades, estudios científicos, etc. Y como medio de interacción social entre individuos, permite el almacenamiento histórico de todos los datos involucrados en investigaciones, operaciones de mercado, transacciones financieras, debates, intereses, opiniones, entre otros. Todos estos datos reciben el nombre de universo digital [1].

En 2010 el tamaño del universo digital excede el zettabyte (equivalente a un trillón de gigabytes) y en 2011 llega a un valor de 1.8 zettabytes, que desde una perspectiva histórica ha aumentado su tamaño nueve veces desde 2006 [1]. Existe entonces la necesidad de teorías y herramientas computacionales que puedan asistir a los humanos en la tarea de extracción información útil con un nivel de abstracción lo suficientemente alta en los inconmensurables volúmenes de datos digitales [2] que al cabo de cinco años crecerán en un factor de ocho con respecto al volumen actual [1].

Tradicionalmente la transformación de datos en conocimiento depende del trabajo y análisis manual de personas familiarizadas con la organización, siendo los productos de este proceso de análisis lento, costoso y altamente subjetivos generando ruido en la argumentación de la toma de decisiones y la planeación. Adicionalmente con el crecimiento desproporcionado de volúmenes de datos este tipo de procedimiento manual se vuelve totalmente impráctico desde diferentes dominios [2].

La minería de datos surge a partir de dicha necesidad de extracción de conocimiento (patrones e información útil caracterizada por tener un alto nivel de abstracción) de los altos volúmenes de datos digitales [2] almacenados en bases de datos, almacenes de datos, OLAP u otros repositorios de información [3] consecuencia de los recientes avances técnicos en poder de procesamiento, capacidad de almacenamiento, y la interconectividad de las tecnologías computacionales [4].). Sus orígenes se remontan a la década de los 80's cuando el termino empezó a ser utilizado en la comunidad investigativa, especialmente interesada en el análisis y extracción de conocimientos en las crecientes bases de datos, de diversos orígenes [5]. Esta disciplina pertenece al campo del Descubrimiento de Conocimiento en Bases de datos (Knowledge Discovery Databases) cuyo enfoque es el mapeo de datos de bajo nivel, el cual debido a su alto y creciente volumen dificulta la comprensión y entendimiento de comportamientos y patrones ocultos en ellos, para su transformación en formas más compactas, más abstractas o más útiles [2].

Formalmente se define a la minería de datos como una ciencia derivada de las ciencias de la computación, especializada en la extracción de conocimiento útil y oculto [5] en grandes repositorios de datos [4]. Se compone de diversas disciplinas para el desarrollo de sus conceptos, herramientas y algoritmos usando una mezcla de la ciencia estadística, la inteligencia artificial y la investigación en bases de datos [6]. El objetivo de los métodos estadísticos es promover prácticas óptimas para el análisis de los conjuntos de datos masivos, permitiendo a los

investigadores usar herramientas, teorías y prácticas de inferencia estadística con alta pericia [7].

1.1.1. Estilos de minería de datos.

Existen dos estilos de la minería de datos, que no son mutuamente excluyentes y que en muchas ocasiones son necesarias para el desarrollo exitoso de proyectos de minería de datos [8]:

- **Minería de datos supervisada:** Este enfoque es utilizado cuando el analista o investigador conoce el propósito de la minería de datos y desea el resultado más preciso posible, sin importar lo que el modelo realice. Este es un enfoque predictivo representado usualmente bajo un modelo de caja negra, que recibe una o más entradas y arroja una salida que corresponda a la mejor predicción posible, ver Figura 1.

Figura 1 - Modelo de caja negra

Este estilo de minería usualmente corresponde a la búsqueda de datos o modelos predictivos, que se basan en datos históricos y entrenamientos continuos y nunca son 100% precisos.

- **Minería de datos no supervisada:** en algunas ocasiones el enfoque predictivo no corresponde al objetivo de la aplicación de la minería. Esto se debe a la necesidad de descubrir nuevos patrones ocultos en los datos. Estos patrones corresponden a nuevos conocimientos, que en el contexto son muy informativos, siendo este enfoque descriptivo y predictivo. Este estilo de minería se suele representar con una caja semitransparente, ver Figura 2.

Figura 2 - Modelo de caja semitransparente

Este estilo de minería permite la comprensión del modelo, pues su objetivo es buscar entendimiento y conocimientos ocultos en los datos. A diferencia de la minería de datos supervisada, en este caso deseamos entender como el modelo funciona.

1.1.2. Objetivos de la minería de datos.

En general las tareas de la minería de datos pueden ser clasificadas dentro de dos categorías [9]:

- **Descriptivo:** encontrar patrones interpretables por humanos, asociaciones, correlaciones y descripciones de los datos.
- **Predictivo:** construir uno o más conjuntos de datos, que permitan inferir resultados en los conjuntos de datos disponibles e intenta predecir el comportamiento de nuevos conjuntos de datos.

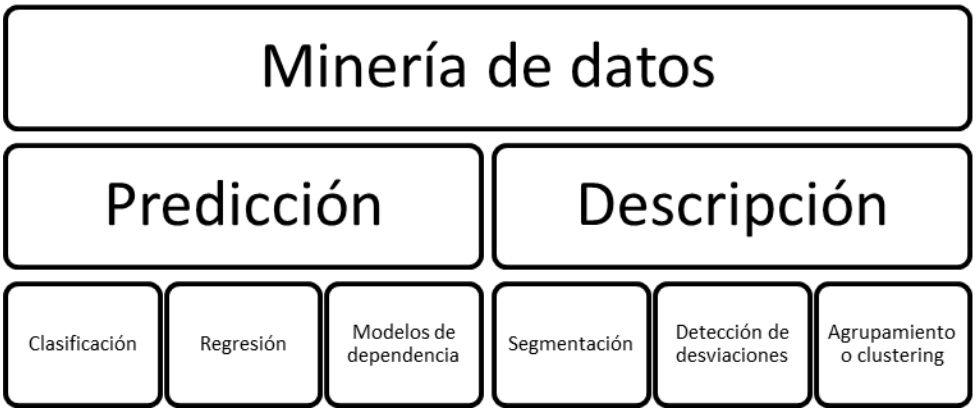
Los límites entre la descripción y la predicción no siempre son claros, pues algunos modelos pueden ser descriptivos y predictivos al mismo tiempo. La importancia relativa de la predicción y la descripción para aplicaciones de minería de datos particulares pueden variar considerablemente. Las tareas de predicción y descripción pueden ser alcanzadas utilizando métodos particulares de minería de datos [2].

1.1.3. Técnicas de minería de datos.

Dependiendo del objetivo que se desee abarcar mediante minería de datos, existen un conjunto de técnicas o métodos (Figura 3), que mediante el análisis y procesamiento del conjunto de datos obtiene resultados representativos ya sea para tareas descriptivas o predictivas. Para el desarrollo de este proyecto se utilizó

específicamente la técnica de agrupamiento o clustering por lo que profundizaremos en este tema.

Figura 3 - Objetivos de la minería de datos



1.1.3.1. Agrupamiento o clustering.

Corresponde a una tarea descriptiva donde se busca identificar un set finito de categorías o clústeres para describir un conjunto de datos [10]. Es el proceso de clasificar un conjunto de datos u observaciones en subconjuntos [11]. Dichas categorías o clústeres pueden ser mutuamente excluyentes y exhaustivas o ser altamente representativas con orden jerárquico o categorías sobrepuestas [2].

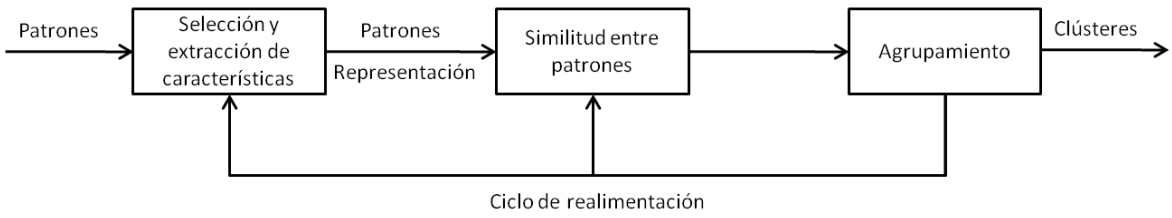
Puesto que el particionamiento del subconjunto es realizado por procedimientos o algoritmos sin intervención humana, puede llevar al descubrimiento de categorías o grupos previamente desconocidos [11].

Esta técnica de minería de datos es utilizada usualmente en tareas de conocimiento y segmentación de mercado, detección de intrusiones de red, estudios poblacionales, entre muchas otras aplicaciones.

Una tarea típica de clustering requiere de las siguientes actividades [10], ver Figura 4:

- Representación de patrones y extracción de características.
- Definición de una medida de proximidad apropiada para el dominio de datos.
- Clustering o agrupamiento.
- Abstracción de datos de ser necesaria.
- Caracterización de la salida.

Figura 4 - Actividades típicas de una tarea de clustering



Fuente: Jain, et al. 1988 [10].

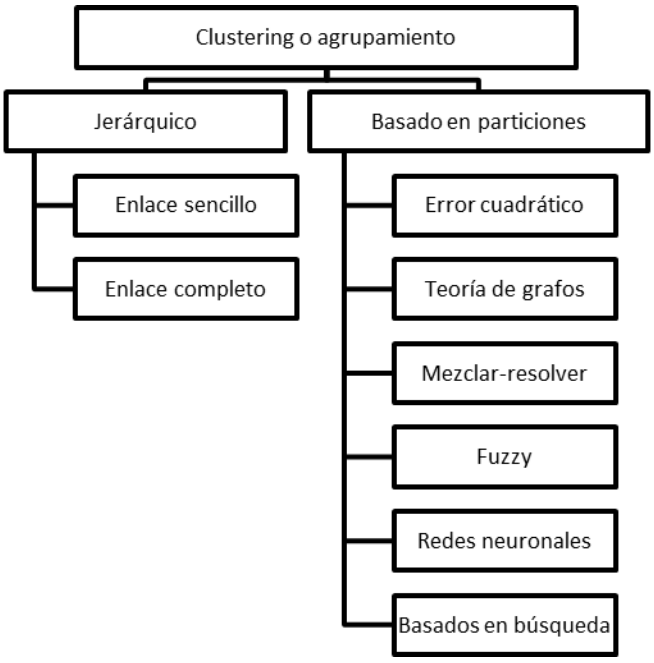
Al terminar el proceso se pueden identificar varios tipos de clúster [12]:

- Clústeres bien definidos: cada del clúster es un conjunto de objetos en los cuales todos los objetos son similares siempre y cuando pertenezcan al mismo clúster.
- Clústeres basados en el centroide o prototipo: un clúster es un conjunto de objetos los cuales cada objeto es cercano o similar al prototipo que define ese clúster que a cualquier otro prototipo de otro clúster. Para datos con atributos continuos el prototipo es usualmente un centroide, la media de todos los puntos del clúster.
- Clústeres contiguos o basados en grafos: si los datos son representados como grafos, donde los nodos son objetos y los vínculos representan conexiones entre objetos, entonces un clúster es un componente conectado, es decir, un grupo de objetos que están conectados entre sí pero no tienen conexiones afuera del grupo. Esta definición de clúster es útil cuando los clúster son irregulares y pueden presentar ruido.

- Clústeres basados en densidad: un clúster es una región densa de puntos, en las que se encuentran regiones de baja densidad y regiones de alta densidad que están separados entre sí. Este tipo de clústeres se usa cuando son irregulares debido a la presencia de ruido y valores extremos.
- Clúster por propiedades compartidas o clúster conceptuales: generalmente, se define un clúster como un conjunto de objetos que comparten una misma propiedad. Este tipo de clúster contiene objetos que comparten la representación de un concepto particular que no se encuentra en otros clúster.
- Clúster descrito por una función objetivo: encontrar clúster que maximizan o minimizan la función objetivo.

Existen varias técnicas y métodos para realizar actividades de clustering o agrupamiento, ver Figura 5:

Figura 5 - Técnicas de clustering



- **Clustering o agrupamiento jerárquico**: es un método de análisis de clúster que busca la construcción de una jerarquía de clústeres. Los

algoritmos de clustering jerárquicos construyen los clústeres gradualmente y su enfoque se subdivide en dos subcategorías.

- Enlace simple: la distancia entre dos grupos es la distancia entre los miembros o elementos más cercanos.
- Enlace completo: donde la distancia entre los dos clústeres es el máximo entre todos los pares de distancias entre los dos clústeres. Este método tiende a la creación de clústeres fuertemente unidos o compactos.

- **Clustering por particionamiento**: un algoritmo de clustering por particionamiento obtiene una sola partición de los datos en vez de una estructura jerárquica de los clúster. Estos métodos tienen ventajas sobre enormes conjuntos de datos. Su funcionamiento se basa en técnicas de optimización usando una función criterio definido local o globalmente. Este tipo de clustering se subdivide en las siguientes subcategorías:

  - **Algoritmos con el error cuadrático**: corresponde a la función criterio más intuitiva y usada en las técnicas de particionamiento de clústeres y suele dar mejores resultados con clústeres aislados y compactos.

    El error cuadrado para el proceso de clustering L de un conjunto de patrones χ (que contiene K clústeres) es:

    $$e^2(\chi, \mathcal{L}) = \sum_{j=1}^{K} \sum_{i=1}^{nj} \left\| x_i^{(j)} - c_j \right\|^2$$

    Donde x_i^j es el *i*-ésimo patrón que pertenece al *j*-ésimo clúster y c_j es el centroide del *j*-ésimo clúster.

  - **Clustering por teoría de grafos**: el algoritmo más conocido por clustering de división por grafos es basado en la construcción del árbol de expansión mínima (minimal spanning tree – MST) de los

datos y luego borrando los bordes con las más grandes distancias para generar clústeres.

- **Algoritmos de Mezclar-Resolver y Búsqueda de modo**: se asume que los patrones para ser agrupados son resultado de una de diversas distribuciones y el objetivo es identificar los parámetros de cada una y quizás su número.
- **Algoritmos de Mezclar-Resolver y Búsqueda de modo**: se asume que los patrones para ser agrupados son resultado de una de diversas distribuciones y el objetivo es identificar los parámetros de cada una y quizás su número.
- **Fuzzy clustering:** las técnicas tradicionales de clustering buscan la creación de particiones, en una partición, cada patrón u objeto pertenece a uno y solo un clúster. Por lo tanto los clústeres son mutuamente excluyentes. Este método extiende esta noción y la asocia a cada patrón a todos los clúster usando una función de membrecía.
- **Redes neuronales artificiales para clustering:** uso del algoritmo de redes neuronales (Hertz et al. 1999) para la clasificación de datos basados en los valores nominales de los mismos.
- **Enfoques evolucionarios de clustering:** los enfoques evolucionarios, son motivados por la evolución natural, haciendo uso de operadores evolucionarios y una población de soluciones para obtener la partición global óptima de los datos. La solución candidata es codificada como cromosomas.
- **Enfoques basados en búsqueda:** son dos las técnicas de búsqueda usadas para obtener el valor óptimo de una función criterio.
    - **Determinísticas:** que garantizan la partición optima ejecutando enumeraciones exhaustivas de los datos.

- **Estocástica:** que generan soluciones cercanas a las óptimas razonablemente rápido.

- **Clustering basado en grid:** este tipo de algoritmos esta propuesto para su aplicación sobre minería de datos espacial. Su principal característica es la cuantificación del espacio en un número finito de celdas sobre el cual se ejecuta el análisis y agrupamiento.

El tráfico de red y los datos usados para el desarrollo de este trabajo presentan una alta variabilidad con gran presencia de ruido, por lo cual es necesario trabajar con grupos o clústeres de formas arbitrarias basados en densidad. Por ende las técnicas de agrupamiento por particionamiento para formar clústeres por densidad se consideran las idóneas para encontrar de forma ágil y precisa intrusiones de red.

1.2. PROTOCOLO TCP/IP

En mayo de 1974 se publica [13] en la universidad de Stanford, planteando un método de comunicación entre redes de computadores a través del empaquetamiento de datos. En diciembre del mismo año se conoce la especificación RFC 675 [14] que estandariza el termino TCP (Transmission Control Program) y define las bases para el funcionamiento de dicha comunicación.

Tras años de desarrollo y múltiples pruebas realizadas en distintos centros de investigación en EEUU, bajo el patrocinio de DARPA, se logra el 1 de enero de 1983 la implementación funcional del protocolo TCP/IP para la comunicación entre computadores de dicha organización.

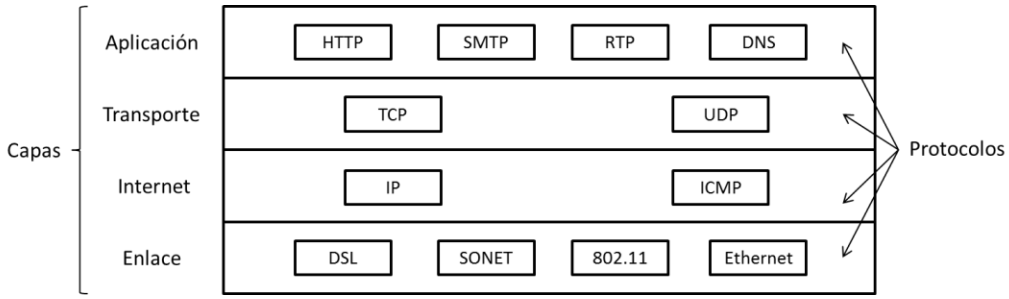
En enero de 1980 bajo la publicación RFC 760 [15] de la IETF es planteado inicialmente el protocolo IPv4 y se detalla una mejora de la especificación en septiembre de 1981 bajo la norma RFC 791 [16]. TCP/IPv4 es actualmente la

versión estándar y de mayor uso a nivel mundial aunque se encuentra en proceso la implementación y despliegue del protocolo TCP/IPv6 definido en la norma RFC 2460 [17].

El protocolo TCP/IPv4 funciona bajo un principio de encapsulamiento y niveles o capas que proveen un nivel de abstracción de protocolos y servicios (RFC 1122) [18]. Estas capas corresponden a, ver Figura 6:

1. Capa o nivel de enlace de datos: esta capa provee una interfaz de servicio bien definida para conexión de hosts y la transmisión de paquetes. Es la encargada de regular el flujo de datos y de lidiar con los problemas de transmisión.
2. Capa o nivel de Internet: su función es permitir a los hosts inyectar paquetes en cualquier red y permitir que viajen independientemente de su destino (potencialmente ubicado en una red diferente). Esta capa define un formato de paquete oficial, un protocolo denominado IP (Internet Protocol) y un protocolo que ayuda a su funcionamiento denominado ICMP (Internet Control Message Protocol). Ruteo de paquetes y congestión son temas fundamentales en la definición de esta capa.
3. Capa o nivel de transporte: está diseñada para permitir entablar una conversación entre pares en la fuente y el destino. Se usan dos protocolos clave a nivel de transporte: TCP (Transmission Control Protocol) y UDP (User Datagram Control).
4. Capa o nivel de aplicación: donde los protocolos de más alto nivel funcionan (HTTP, SMTP, FTP, SSH, etc.) y donde las aplicaciones de interacción directa con el usuario interpretan paquetes enviados a través de la red.

Figura 6 - Capas y protocolos de TCP/IPv4



Fuente: Tanenbaum, et al. 2010 [19].

A continuación se presentan elementos más específicos de cada una de las capas y niveles que pertenecen al protocolo TCP/IP y se profundizara en los datos relevantes para el desarrollo de este proyecto.

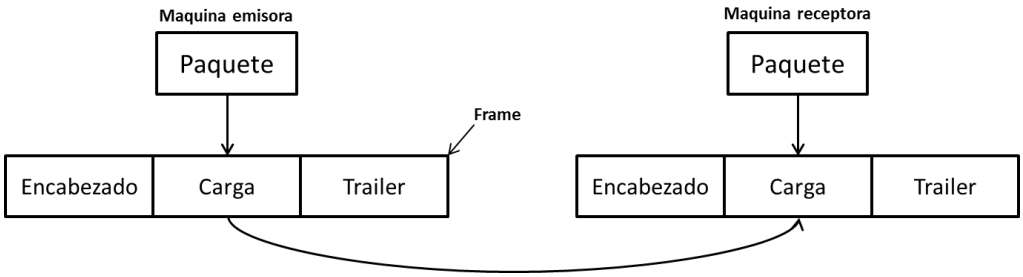
1.2.1. Capa o nivel de enlace de datos

La tarea de la capa o nivel de enlace de datos es convertir el stream de bits puro ofrecido por los dispositivos físicos mediante diversos canales de comunicación en un stream de frames para su uso en la capa de internet [19]. Debe estar en la capacidad de:

- Proveer una interfaz de servicio claramente definida.
- Manejar errores y problemas de transmisión.
- Regular el flujo de datos para mantener un ritmo único de envío y recibo de datos entre máquinas de diversas velocidades y especificaciones.

Para cumplir con estos objetivos la capa o nivel de enlace de datos toma los paquetes que recibe de la capa de internet y los encapsula en frames para su transmisión. Cada frame contiene un encabezado, un campo de carga en donde se almacena el paquete y finalmente un tráiler [19].

Figura 7 - Capa de enlace de datos



Fuente: Tanenbaum, et al. 2010 [19].

En la Figura 7, se observa el funcionamiento de la capa de enlace de datos. El lado del host que envía datos, el controlador (adaptador de red, interfaz de red) toma el datagrama que se crea y se almacena en memoria en los niveles superiores de la pila de protocolos (internet, transporte, aplicación), encapsula dicho datagrama en un frame (llenando los campos adicionales del frame) y posteriormente lo transmite mediante el enlace de comunicación haciendo uso de protocolos a nivel de enlace. En el lado del host receptor de datos, el controlador recibe el frame completo y extrae el datagrama a nivel de internet [20].

1.2.2. Capa o nivel de internet

La capa o nivel de internet es la encargada de llevar los paquetes por una gran ruta desde la fuente hasta el destino. Cumplir con esta tarea requiere de muchos saltos y el paso a través routers intermedios. Esta función se diferencia claramente de la capa o nivel de enlace cuyo objetivo es mover frames desde el final de un cable hasta el otro [19]Este movimiento de paquetes desde un host remitente hasta un host receptor implica dos funciones importantes [20].

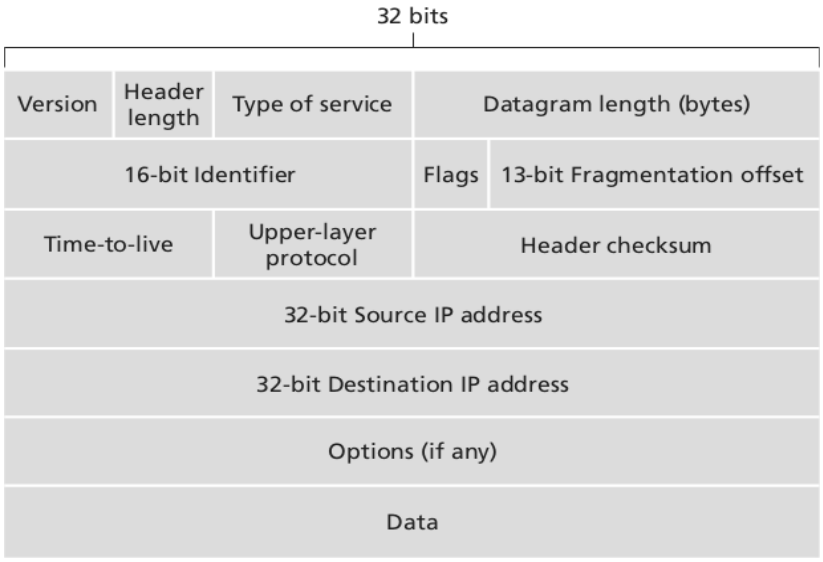
- Reenvío: cuando un paquete llega al canal de entrada de un router, el router debe mover el paquete al canal de salida correcto. Para la realización de esta tarea el router crea una tabla que rige el reenvío de paquetes.

- Ruteo: la capa de internet de determinar la ruta o el camino tomado por los paquetes mientras fluyen desde el host remitente hasta el host destino. Los algoritmos que calculan estos caminos son denominados algoritmos de ruteo.

Para su funcionamiento, el protocolo IP encapsula el paquete de datos en un formato denominado Datagrama. El datagrama de IPv4 juega un papel fundamental en el desarrollo y funcionamiento de la internet, su formato encapsula los campos necesarios para el envío de datos a través de la red, mostrados en la Figura 8.

La capa o nivel de internet provee servicios clave para la capa de transporte. Puede estar basada en cada datagrama o en circuitos virtuales. En cualquier caso el objetivo principal de dicha capa es el ruteo de paquetes desde la fuente hasta el destino [20].

Figura 8 - Datagrama IP



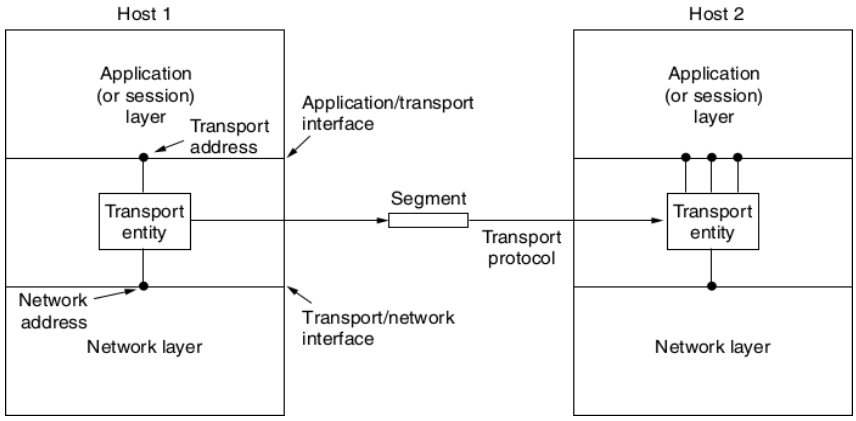
Fuente: Kurkose, et al. 2005 [20].

1.2.3. Capa o nivel de transporte

Junto con la capa de internet, la capa de transporte es el corazón de la jerarquía de protocolos. La capa de internet provee la entrega de paquetes de un host a otro haciendo uso de datagramas. La capa de transporte se construye sobre la capa de internet para proveer datos de transporte de un proceso en una maquina fuente a un proceso en una maquina destino con un nivel de confiabilidad independiente de las redes físicas actualmente en uso. Provee la abstracción que las aplicaciones necesitan para hacer uso de la red [19], ver Figura 9.

El objetivo principal de la capa de transporte es proveer un servicio de transmisión de datos a sus usuarios (procesos de la capa de aplicación) de forma eficiente, confiable y de bajo costo. Para lograr esto la capa de transporte hace uso de servicios proveídos por la capa de internet. El software y el hardware que en la capa de transporte realizan este trabajo se denominan entidades de transporte. Estos pueden ser librerías a nivel del kernel en un sistema operativo o implementaciones directas en la interfaz de red.

Figura 9 - Interacción capa de transporte y capa de internet



Fuente: Tanenbaum, et al. 2010 [19].

Las redes TCP/IP hacen distinción entre dos protocolos disponibles a nivel de transporte para su uso a nivel de aplicación. Uno de estos protocolos es UDP

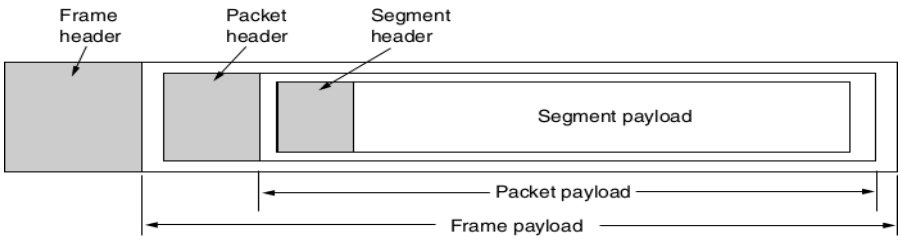
(User Datagram Protocol), que provee un servicio no confiable y sin conexión a la aplicación que invoca la comunicación. El segundo de estos protocolos es TCP (Transmission Control Protocol) que provee una conexión confiable, orientada a la conexión a servicio de la aplicación que la invoca [20].

1.2.3.1. Protocolo TCP

Para el funcionamiento del protocolo TCP se realiza un encapsulamiento del datagrama creado a nivel de internet y al resultado se le denomina segmento.

Los segmentos (usados a nivel de transporte) están contenidos en datagramas (usados a nivel de internet) que a su vez están contenidos en frames (usados a nivel de enlace de datos). Este encapsulamiento se realiza conforme avanzan los datos a través del conjunto de protocolos en el host remitente. Cuando se captura en la red el frame a nivel de enlace de datos, se procesa el encabezado y si la dirección de destino concuerda con alguna para un envío local se envía el campo de carga o datos a la entidad de red. La entidad de red procesa similarmente el encabezado del datagrama y pasa su carga o datos a la entidad de transporte. Este encapsulamiento se encuentra ilustrado en la Figura 10.

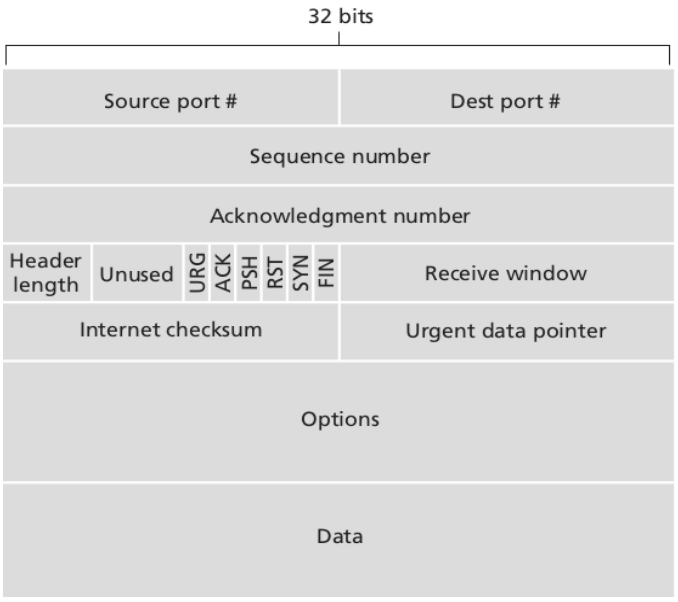
Figura 10 - Encapsulamiento de segmentos, datagramas y frames



Fuente: Tanenbaum, et al. 2010 [19].

Un segmento TCP consta de campos de encabezado y un campo de datos como se ilustra en la Figura 11.

Figura 11 – Estructura de un segmento TCP



Fuente: Kurkose, et al. 2005 [20].

Los campos mostrados en la Figura 11 son:

- Puerto fuente y puerto destino: los puertos son una serie de sockets en modo escucha ubicados lógicamente dentro del host. Su función es servir de canales de comunicación para diversas aplicaciones entre las entidades de transporte de dos hosts diferentes. Cada puerto tiene un identificador único de 16 bits que va de 0 a 65535. Los puertos del rango 0 a 1023 se denominan puertos bien conocidos y se encuentran reservados para para protocolos bien conocidos a nivel de aplicación (HTTP usa el puerto 80, FTP usa el puerto 21, etc.). El listado de estos puertos está definido en la norma RFC 3232 [21].

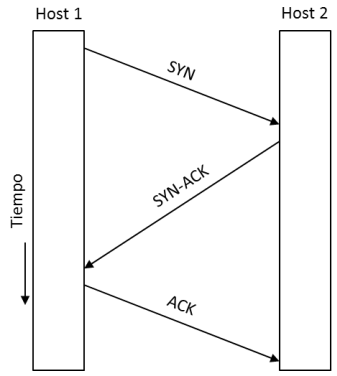
  Cada segmento enviado a través de la red va dirigido a un puerto específico en el host destino y es originario de un puerto específico en el host remitente.
- Numero de secuencia: número de 32 bits usado para secuenciar paquetes de datos que fluyen desde el emisor hasta el receptor. Saltos en los

números de secuencia le permiten al receptor detectar la perdida de algún paquete en la transmisión. Igualmente permite la detección de paquetes duplicados debido a la doble presencia del mismo número de secuencia.

- Numero de reconocimiento: número de 32 bits usado por el receptor para notificarle al emisor que el conjunto de paquetes o el paquete se han recibido correctamente. Este número usualmente tiene el valor del número de secuencia del paquete o de los paquetes reconocidos en el receptor. Los reconocimientos pueden ser individuales o acumulativos dependiendo del protocolo.
- Tamaño del encabezado: este campo especifica el tamaño del encabezado en palabras de 32 bits. Esta información es necesaria puesto que el campo Opciones del encabezado es de tamaño variable e indicar la posición en que se inician los datos en el paquete.
- Banderas o flags: 6 banderas de 1 bit cada una usadas para comunicar estados de la conversación entre dos hosts.
- Tamaño de la ventana: en este campo mediante un número de 16 bits se especifica el tamaño en bytes del buffer para la conexión TCP actual.
- Checksum: campo de 16 bits para el chequeo de errores en el encabezado y los datos durante la transmisión del datagrama.
- Puntero a datos urgentes: cuando se marca la bandera urgente, este campo de 16 bits corresponde a un desplazamiento del número de secuencia indicando el último byte caracterizado como urgente.
- Opciones: este campo provee facilidades extra que no son cubiertas por el encabezado. Son de tamaño variable y pueden llenar un múltiplo de 32 bits utilizando el espaciado con ceros.
- El protocolo TCP tiene un mecanismo para establecer una conexión entre un par de hosts para el envío y la recepción de datos durante un periodo de tiempo. Este mecanismo se denomina negociación en tres pasos y consiste en: primero, el cliente solicita el establecimiento de la conexión a un host remoto a través del envío de un segmento SYN, segundo, en respuesta el

host remoto responde con un segmento SYN-ACK al cliente que solicita la conexión, tercero, el cliente envía un segmento ACK nuevamente al servidor lo que confirma la apertura de la conexión. Este procedimiento se ilustra en la Figura 12.

Figura 12 - Negociación en tres pasos



Fuente: Tanenbaum, et al. 2010 [19].

Este procedimiento es aprovechado por técnicas de ataque a servidores como los ataques de denegación de servicio.

1.2.4. Capa o nivel de aplicación

Finalmente la capa de aplicación ofrece la estructura para que las aplicaciones accedan al conjunto de protocolos inferiores en la red. Es la que realiza la interacción directa con el usuario y la abstracción necesaria para que los datos recibidos y enviados a través de la red sean visualizados, entendidos y leídos por el usuario receptor. Existen múltiples protocolos implementados en esta capa y dependen de la aplicación que se utilice, correo, visualización de video, stream de audio, transferencia de archivos, entre otros, tienen protocolos propios implementados en esta capa.

1.3. ATAQUES DETECTADOS POR LA HERRAMIENTA

1.3.1. Denegación de servicios mediante SYN-FLOOD

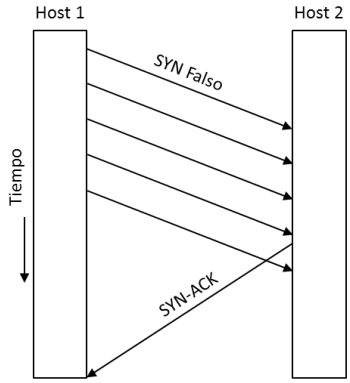
El ataque SYN-FLOOD corresponde a un método de denegación de servicios que afecta a hosts que corren procesos de servidores TCP. Estos ataques toman ventaja del estado de retención que el protocolo TCP realiza tras recibir un segmento SYN en un puerto que se ha puesto en modo escucha. La idea básica es explotar este comportamiento causando que el host víctima retenga un número de semi-conexiones disminuyendo los recursos para la atención de nuevas conexiones legítimas.

Este ataque se da solo a nivel de la capa de Transporte en la arquitectura de la red, más específicamente en el protocolo TCP y se aprovecha del método para establecimiento de una conexión entre dos hosts denominado negociación en tres pasos.

En la Figura 13 se muestra la forma en que opera el ataque SYN-FLOOD enviando múltiples solicitudes de conexión SYN simultáneamente desde diversos hosts cliente o direcciones IP falsas, agotando la capacidad de respuesta del host víctima.

Se estima que aproximadamente en el primer cuarto de 2012 de los ataques de denegación de servicios realizados globalmente a sectores públicos y privados, el 24.66% corresponden a ataques SYN-FLOOD, siendo este el más representativo.

Figura 13 - Forma en que opera un ataque de denegación de servicios SYN-FLOOD



1.3.2. Probing y escaneo de vulnerabilidades mediante SATAN

SATAN (Security Administrator Tool for Analyzing Networks) es un ataque tipo probing para el escaneo de puertos, redes y hosts para encontrar posibles vulnerabilidades que puedan ser explotadas por un atacante. Se evidencia a partir de un número alto de peticiones de conexión a un host víctima, haciendo uso de diversos protocolos a nivel de aplicación y múltiples puertos.

Existen tres niveles de escaneo: escaneo ligero, escaneo mediano y escaneo pesado, y busca desde la disponibilidad de puertos hasta el número de archivos específicos que pueden ser accedidos en el host de la víctima sin autorización.

1.4. SISTEMAS DE DETECCIÓN DE INTRUSIONES

Corresponden a sistemas de monitoreo computacional que buscan señales de la presencia de usuarios no autorizados, o de usuarios abusando de sus privilegios.

Un IDS realiza las siguientes actividades:

- Monitorea múltiples fuentes de información de los sistemas computacionales y los analiza.

- Captura el tráfico de red y lo compara con ciertos patrones específicos de ciertos ataques.
- Identifica problemas causados por usuarios que abusan de los privilegios asignados en el sistema.
- Mediante el análisis estadístico del tráfico capturado busca patrones anormales en las actividades.

El concepto de IDS dio inicio en el año 1980 a través del proyecto de investigación [22] que introdujo la noción de pistas de auditoria o logs de actividad de los usuarios, útiles para rastrear información valiosa del comportamiento del usuario y la detección de intrusiones por mal uso. Su trabajo dio inicio a la investigación de los HIDS o Sistemas de Detección de Intrusiones por Host.

1.4.1. Historia de los sistemas de detección de intrusiones basados en Host (HIDS)

Las autoridades militares norteamericanas notaron que el análisis de eventos de seguridad en sistemas se volvía cada vez más tedioso debido al incremento en el uso de los computadores. James P. Anderson fue la primera persona en documentar la necesidad de un mecanismo que automatizara la revisión de los eventos de seguridad en un estudio encargado por las fuerzas aéreas de EEUU en 1980. Él Ideó un sistema de clasificación que distinguía entre ataques externos e internos, basado en si los usuarios tenían o no acceso a los computadores, para dar solucionar al problema de intrusos que se apoderaron de cuentas de usuarios legitimas. El sistema se basaba en patrones de uso creados a partir de análisis estadísticos de comportamientos de los usuarios para distinguir comportamientos inusuales [23].

- **IDES (Intrusion Detection Expert System)**: Modelo que definía un sistema de detección de intrusiones en tiempo real desarrollado entre 1984 y 1986 por Dorothy Denning y Peter Neumann, proponía una correspondencia

entre actividad anómala y uso indebido. El modelo usaba perfiles para representar el comportamiento de sujetos (principalmente usuarios), y reglas de actividad para definir las acciones que tenían lugar (eventos del sistema o tiempos de CPU). Estos elementos permitían establecer mediante métodos estadísticos las pautas de comportamiento necesarias para detectar posibles anomalías. El sistema IDES era considerado un sistema híbrido ya que añadía un nivel de seguridad adicional mediante el uso de sistema experto, basado en reglas de seguridad, que disminuía los efectos de un intruso que intentara evitar el detector de anomalías [23].

Posteriormente aparecieron numerosos sistemas de detección de intrusiones:

- **Automated Audit Analysis:** Proyecto dirigido por SYTEK un grupo de desarrollo. El sistema recogía información a nivel de interfaz de comando Shell de un sistema UNIX y los comparaba con una base de datos. Estos se analizaban estadísticamente para demostrar que se podían detectar comportamientos anormales [23].
- **Discovery:** Sistema creado por William Tener, utilizaba métodos estadísticos escritos en COBOL para detectar posibles abusos. Discovery era novedoso porque monitorizaba una aplicación y no un sistema operativo [23].
- **Hystack:** desarrollado por Steve Smaha, era un prototipo de sistema de detección de intrusiones escrito en C ANSI y SQL el cual analizaba los logs de su sistema multiusuario con el fin de ayudar a Oficial de seguridad del sistema a detectar intrusiones y actividades anómalas [23].
- **MIDAS (Multics Intrusion Detection and Alerting Systems):** creado por el NCSC (National Computer Security Center) compuesto por cuatro niveles de reglas, además contaba con una base de datos que usaba para determinar comportamientos anormales, al igual que IDES utilizaba un sistema híbrido en el que combinaba tanto la estadística como reglas de seguridad de un sistema experto [23].

- **NADIR (Network Audit Director and Intrusion Reporter):** Desarrollado en el Laboratorio Nacional de los Álamos utilizaba técnicas de detección de comportamientos atípicos al igual que IDES y MIDAS [23].
- Wisdom and Sense (W&S) Genera automáticamente reglas de un historial de datos y basado en esas reglas identifica transacciones del computador cuya varianza genera patrones de uso históricos [24].

Al iniciar la década de los 90, se dio inicio a una rápida evolución de las redes de computadores siendo necesaria la creación de nuevos modelos en detección de intrusiones. El virus del gusano de internet y su aparición en el año 1988 ayudo a que la industria y la academia unieran esfuerzos para dar solución a este problema de seguridad [25].

1.4.2. Historia de los sistemas de detección de intrusiones basados en redes (NIDS)

- **NSM (Network System Monitor):** Fue el primer sistema de detección intrusiones que monitoreaba el tráfico de la red, desarrollado por la Universidad de California. Inicialmente activaba el dispositivo de red en modo promiscuo para monitorear todo el tráfico que recibía, capturaba los paquetes de datos, identificaba el protocolo utilizado para poder extraer los datos necesarios (IP, ICMP, etc.), utilizaba el enfoque basado en matrices para analizar y almacenar los datos en busca de variaciones estadísticas que definieran comportamientos anómalos.
- **DIDS (Sistema de Detección de Intrusiones Distribuido):** corresponde al primer sistema capaz de monitorear y detectar intrusiones a través de redes de computadores, siendo capaz de rastrear y hacer seguimiento del intruso identificado en la red utilizando un modelo de detección de intrusiones estratificado en seis niveles para distinguir los tipos de datos.

- Productos comerciales: a partir de 1990 y hasta la fecha surgen cientos de productos comerciales que buscan atacar el problema de las intrusiones a través de redes TCP/IP.

1.4.2.1. Desventajas de los NIDS

- Latencia alta entre el momento en que sucede el ataque y el momento en que se recibe la identificación y notificación.
- Generan una tasa considerable de falsas alarmas o falsos positivos.
- El tráfico cifrado no es analizado por el sistema.
- No indica si un ataque ha sido exitoso o no.
- Su efectividad depende de la última actualización de firmas o patrones.
- Si existe una alta congestión de red se dificulta el análisis de los paquetes.

1.4.3. Técnicas de detección de intrusiones y ataques de red

Las técnicas de detección de intrusiones en general se dividen en dos categorías: *detección de mal uso y detección de anomalías.*

- **Detección de mal uso** [26]**:** cada instancia en un conjunto de datos es etiquetada como “normal” o “intruso” y el algoritmo de aprendizaje es entrenado de acuerdo a la etiqueta de los datos. Estas técnicas son capaces de entrenar de forma automática los modelos de detección de intrusiones con datos de entrada diferentes, que incluyen nuevos tipos de ataques, siempre y cuando se hallan etiquetado apropiadamente.

  La detección de mal uso se ha enfocado principalmente en el uso de algoritmos de clasificación y reglas de asociación. Una gran ventaja de estas técnicas es la precisión para la detección de ataques conocidos y sus variaciones y su principal desventaja es la inhabilidad para detectar ataques cuyas instancias aún no han sido observadas

- **Detección de anomalías** [26]**:** éste enfoque, se centra en la construcción de modelos de datos normales y la detección de desviaciones en los datos que están siendo observados.

  Los algoritmos de detección de anomalías están capacitados para que puedan detectar nuevos tipos de intrusiones tales como comportamiento inusuales. Dado un conjunto de datos de entrenamiento y un conjunto de datos de prueba, el objetivo del algoritmo de detección de intrusiones es determinar si el conjunto de datos de prueba pertenece a un comportamiento normal o anómalo. Sin embargo éste enfoque sufre de una alta rata de falsas alarmas, esto ocurre porque los comportamientos del sistema nunca antes vistos, son reconocidos también como anomalías y por tanto marcados como intrusiones potenciales.

  Las técnicas de detección de anomalías se clasifican en dos categorías:

  - **Supervisadas:** dado un conjunto de datos normales para entrenamiento y dado un nuevo conjunto de datos de prueba, el objetivo es determinar si los datos de prueba son normales o anómalos, asignando un puntaje a cada conexión de red el cual refleja cuan anómala es ésta.
  - **No supervisadas:** Intenta detectar comportamientos anómalos sin ningún conocimiento de los datos de entrenamiento, está basado en un enfoque estadístico.

2. ESTADO DEL ARTE

Sobre los diversos métodos de clustering mencionados en el capítulo anterior, el algoritmo DBSCAN según la literatura y los trabajos citados corresponde a uno de los algoritmos más utilizados para las tareas de agrupamiento. Su enfoque basado en densidad y su caracterización como algoritmo no supervisado permite realizar tareas descriptivas sobre grandes volúmenes de datos.

En este capítulo se exploran algunos trabajos en torno a sistemas de detección de intrusiones, agrupamiento o clustering en flujos de datos y el algoritmo DBSCAN aplicado a la detección de intrusiones de red. Este resumen es una parte fundamental de los planteamientos teóricos que forman el modelo resultado de este trabajo.

2.1. SISTEMAS DE DETECCIÓN DE INTRUSIONES BASADOS EN RED NIDS RECIENTES

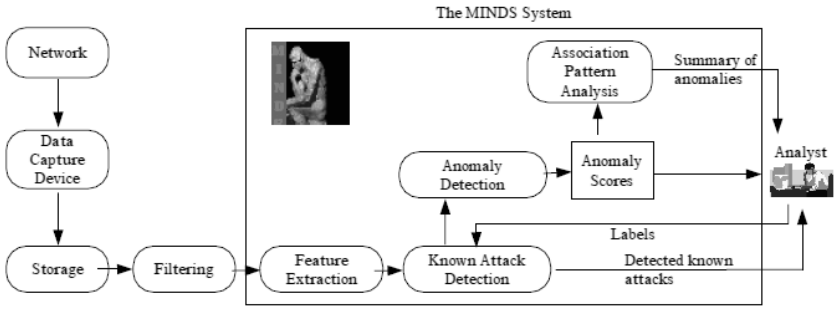
Actualmente el software utilizado por grandes compañías para la detección de intrusiones en red está construido sobre la tradicional metodología basada en firmas. La base de datos del software contiene grandes cantidades de información de ataques que son proveídos por expertos humanos, a medida que se descubre un nuevo ataque éste es desarrollado y revisado manualmente, dicho panorama representa una limitación ya que no puede detectar nuevos tipos de ataques hasta la siguiente actualización. Ésta limitación ha despertado el interés de crear modelos automáticos de detección de intrusiones de red, que a diferencia de los sistemas de detección basados en firmas pueden ser más precisos.

A continuación se enumeran diferentes aportes en el tema de detección de ataques automáticamente mediante técnicas supervisadas y no supervisadas de minería de datos:

- **MINDS (Minnesota Intrusion Detection System)** [26]**:** usa una suite de técnicas de minería de datos para la detección automática de ataques, sus principales contribuciones son: técnica de detección de anomalías que asigna un puntaje a cada conexión de red que refleje cuan anómala es la conexión y un análisis de asociación de patrones basado en el módulo, que resume esas conexiones que fueron clasificadas como altamente anómalas por el módulo de detección de anomalías.

  En la Figura 14. Se muestra el proceso del análisis de tráfico de datos de una red usando MINDS.

Figura 14 - Sistema MINDS



Fuente: Ertöz, et al. 2003 [26].

- **SNORT** [27]**:** es una herramienta de detección de intrusiones de red o NIDS (Network Intrusion Detection System) de bajo consumo de recursos, multiplataforma y de fácil implementación capaz de monitorear redes TCP/IP y detectar gran variedad de ataques.

  Está basada en la herramienta libpcap de captura de paquetes que funciona como logger y sniffer, se caracteriza por la detección de gran variedad de ataques en tiempo real usando un sistema sencillo de reglas. Tiene la capacidad de enviar alertas al syslog, a archivos específicos de "alerta" y otros medios de interacción con el administrador. Su facilidad de

uso permite el desarrollo de nuevas reglas detección combinando el método de firma, protocolo e inspección de anomalías.

Dentro de los componentes que hacen parte de SNORT están:

- **Clasificador de paquetes:** es el encargado de decidir que paquetes serán inspeccionados.
- **Desfragmentador de IP y un re ensamblador TCP:** encargado de asegurarse que SNORT inspeccione los fragmentos IP y los segmentos TCP en el orden correcto.
- **Procesador de escaneo de puertos:** como su nombre lo indica, se encarga de escanear los puertos.
- **Motor de detección y preprocesadores:** realizan la normalización de protocolos, coincidencia de reglas y otras funciones de detección.

- **SPADE (Statistical Packet Anomaly Detection Engine)** [28]**:** SPADE es un plug-in del sistema de detección de intrusiones SNORT que automáticamente detecta escaneos de puerto silenciosos, a diferencia de los detectores tradicionales que buscan ciertos eventos en determinado tiempo SPADE se enfoca en buscar en información obtenida por sondeo.

  SPADE tiene cuatro métodos de calcular la similitud de paquetes. El método más efectivo mide la probabilidad conjunta P (IP destino, Puerto destino).

  SPADE examina paquetes TCP-SYN y mantiene la cuenta de paquetes observados en tuplas (IPdestino, PuertoDestino). Cuando un nuevo paquete es observado, SPADE comprueba la probabilidad de que el paquete se encuentre en la tupla (IPdestino, PuertoDestino), entre más baja sea la probabilidad más alto será el puntaje de anomalía.

2.2. ALGORITMOS DE CLUSTERING EN FLUJOS DE DATOS

- En [29] se Propone una versión más rápida del algoritmo K-means Clustering llamada VFKM, el método muestrea los datos para reducir la cantidad de datos analizados, preservando una similitud entre el análisis del conjunto infinito y el análisis de la muestra. Además propone un método de minimización del tiempo de ejecución del aprendizaje garantizando que dado un conjunto de datos infinitos no se produzcan cambios significativos de un conjuntos de datos finitos.

  El método está basado en limitar la pérdida de aprendizaje como una función del número de ejemplos en cada paso.
- En [30] se describen dos métodos de clustering aplicados a datos de la red, éstos permiten la agrupación de máquinas en "Grupos de Actividad", que consisten en máquinas que tienden a tener perfiles de actividad similares. Además estos métodos permiten al usuario determinar si la actividad actual coincide con esos perfiles y por lo tanto determinar cuando está ocurriendo una actividad "Anormal" en la red.
- En [31] se estudia el algoritmo k-medians en el contexto de flujos y proporciona un algoritmo de flujos de datos, el cual requiere de poca memoria y direcciona el tema de clustering en una sola pasada. El algoritmo se basa en la premisa de divide y vencerás el cual logra un factor constante de aproximación en poco espacio.

  Éste algoritmo se ejecuta en tiempo $O(n^{1+\varepsilon})$, usa memoria $O(n^{\varepsilon})$ y hace una sola pasada sobre los datos, además usa aleatoriedad para mostrar como reduce el tiempo de ejecución a $\tilde{O}(nk)$ sin requerir más que una sola pasada, el algoritmo, produce más de *k* centros, además muestra como la ubicación del algoritmo puede ser modificada para producir exactamente *k clústeres,* y por lo tanto resolver el problema del K-Median.

- En [32] se usa un algoritmo de clustering de un solo enlace. Diseña un sistema basado en vectores de características de la red. El sistema está habilitado para detectar varios tipos de intrusiones con una tasa de falsos positivos baja.
- En [33] se propone un nuevo algoritmo STREAMS el cual se enfoca en el problema de clustering de la siguiente forma, dado un entero k y una colección N de n puntos en un espacio métrico, encontrar k centros del clúster en el espacio métrico de tal forma que cada punto en N es asignado al clúster definido por el punto medio más cercano a éste. La calidad del clúster se mide por la suma de los cuadrados de las distancias (SSQ) de cada punto medio asignado. El objetivo es encontrar un conjunto de k centros de clúster los cuales minimice la medida SSQ.

  El algoritmo STREAM empieza determinando el tamaño de la muestra y luego aplica el algoritmo LOCALSEARCH si el tamaño de la muestra es mayor que el resultado previamente especificado. El anterior proceso se repite para cada fragmento, y finalmente el algoritmo LOCALSEARCH se aplica a los centros de clúster generados en al anteriores iteraciones.
- En [34] se presenta una mejora del algoritmo K-means (Incremental K-means algorithm) para flujos de datos binarios. Entre las mejoras implementadas están: la distancia computacional eficiente para vectores binarios dispersos, operaciones de matrices dispersas y un resumen de los resultados de clustering mostrando valores frecuente binarios y outliers.
- En [35] se estudia el problema de la técnica de Clustering en las aplicaciones de flujos de datos continuos, inicialmente, dado el conjunto de datos, éstos se particionan en grupos cuyos elementos tengan una similitud la cuál es definida con una medida de distancia o función objetivo. El algoritmo está dividido en dos fases las cuales separan el proceso en el componente de micro-clustering online que requiere de un eficiente proceso de almacenamiento de estadísticas de un flujo de datos y el componente macro-clustering offline que usa las anteriores estadísticas juntos con otras

entradas del usuario para dar un rápido entendimiento de los clúster en diferentes períodos de tiempo.

- En [36] se propone un nuevo algoritmo de alta dimensión llamado HPStream (High Dimensional, Projected Data Stream Clustering), el método incorpora una estructura de desvanecimiento del clúster y la proyección basada en la metodología de clustering. El algoritmo se enfoca en un proceso iterativo el cual determina de forma continua nuevas estructuras de clúster mientras redefine el conjunto de dimensiones para cada clúster. Inicia ejecutando un proceso de normalización con el fin de ponderar las diferentes dimensiones correctamente. El conjunto de dimensiones está representado como un vector de bits d-dimensional.
- En [37] se presenta un algoritmo de clustering basado en densidad y basado en grid. Desarrolla una red (grid) y un algoritmo fpMAFIA^2 el cual es una versión mejorada del algoritmo fpMAFIA, es capaz de ejecutar un gran conjunto de datos de 1 millón de registros en un solo computador en menos de 11 minutos. La meta del algoritmo es dividir cada dimensión en particiones desiguales de acuerdo a la distribución de los datos, las cuales son las cestas frecuentes que determinan la formación y límite de los clústeres.

  Básicamente la estructura del algoritmo es la siguiente:

  - Buscar todas las cestas frecuentes haciendo uso del algoritmo modificado de red adaptativa.
  - Transformar las instancias de los datos en cestas frecuentes y usarlas para construir el FP-Tree.
  - Recuperar el conjunto de clústeres candidatos utilizando el método de recuento de vuelta.
  - Examinar todos los conjuntos de clústeres y eliminar los clústeres duplicados.

Una vez se obtienen los clústeres cada punto dentro del clúster es etiquetado como dato normal y los puntos que están fuera de los clústeres se etiquetan cono datos anormales.

- En [38] sus investigaciones producen un resultado sorprendente en sub-secuencias de clustering y además dan la introducción a un nuevo método basado en el concepto de series de tiempo motifs, Figura 15.

  Los resultados de la ejecución de un algoritmo k-means por STS clustering (Subsequence Time Series) para un conjunto de datos del mercado de valores, encontró que los centros de clustering no son significativamente similares entre sí, de tal forma que si se eligiera un dato aleatorio del conjunto de datos como centro del clúster, nadie podría notar la diferencia. El algoritmo considera el concepto de series de tiempo motif, concepto que está altamente relacionado con clustering.

  Motif: puede ser considerado similar a un clúster pero tiene grandes diferencias importantes:

  - Cuando se minan motifs, se debe especificar un parámetro adicional R.
  - La distancia R es definida como distancia Euclidiana, los motifs siempre definen regiones circulares en el espacio, a diferencia de los clústeres que pueden tener formas arbitrarias.
  - Los motifs generalmente definen pequeños subconjuntos de datos y no un conjunto de datos completo.
  - La definición de motif explícitamente elimina las parejas de datos triviales.

Figura 15 – Algoritmo de clustering basado en Motif

Algorithm *motif-based-clustering*

1. Decide on a value for k.
2. Discover the K-motifs in the data, for $K = k \times c$
   (c is some constant, in the region of about 2 to 30)
3. Run k-means, or k partitional hierarchical clustering, or any other clustering algorithm on the subsequences covered by K-motifs

Fuente: Keogh, et al. 2005 [38].

- En [39] se propone un método de clustering DUCstream (Dense Units Clustering for data stream) incremental basado en la detección de unidades de densidad, los clústeres evolutivos son identificados con base en las unidades de densidad, las cuales contienen relativamente una gran cantidad de puntos. El algoritmo de una sola pasada por los datos y basado en la densidad encuentra clústeres de alta calidad en poco tiempo y memoria considerablemente, éste descarta ruido y unidades obsoletas a través de las unidades de densidad. El resultado se actualiza mediante cambios de las unidades de densidad.
- En [40] se presenta el algoritmo DenStream un nuevo enfoque para descubrir clústeres en flujos de datos evolutivos. La densidad del micro-clúster llamada core-micro-clúster encargada de resumir clústeres con formas arbitrarias y las estructuras potencial core-micro-clúster y outliers core-micro-clúster son propuestas para mantener y distinguir el potencial y los outliers del clúster. El algoritmo se divide en dos fases: la parte en línea de mantenimiento del micro-clúster y la parte fuera de línea outlier, la cual debe ser borrada para liberar espacio de memoria para nuevos p-micro-clusters. Por lo tanto es necesario observar el peso de cada p-micro-clúster periódicamente.
- En [41] se propone un algoritmo UMicro (Uncertain MICROClustering) para agrupar flujos de datos inciertos, basado en el modelo de micro-clustering. El algoritmo trabaja usando un enfoque iterativo el cual mantiene un número de centroides de micro-clusters alrededor de los cuales es

construido el clúster. Para cualquier dato entrante se calcula el centroide de clúster más cercano, el cual es determinado por la “Distancia esperada”. Una vez se ha determinado dicha distancia se observa si se encuentra en un límite de incertidumbre crítico, si éste se encuentra dentro del límite de incertidumbre crítico se agrega al micro-clúster.

- En [42] se presenta el algoritmo RepStream basado en un grafo despoblado que emplea puntos representativos de un clúster para procesar de forma incremental los datos entrantes. La descripción basada en el grafo permite modelar las relaciones espacio-temporales en un flujo de datos con más precisión que un resumen de estadísticas. Cada clúster es definido en tipos de puntos representativos: puntos ejemplares que son usados para capturar propiedades estables del clúster puntos predictivos los cuales son usados para capturar las propiedades evolutivas del clúster.
- En [43] se propone el algoritmo D-Stream (Density-Based Clustering framework for Data Stream) el cual usa dos componentes: online y offline. El componente online se usa para mapear todos los datos de entrada en la red y el componente offline que calcula la densidad y clústeres de la red basados en la densidad. El algoritmo adopta una técnica decreciente para capturar los cambios dinámicos del flujo de datos y un mecanismo basado en atracción para generar con precisión los límites del clúster.

  El factor de decrecimiento es utilizado para hacer los clústeres de manera que se vayan agrupando por peso iniciando por el más reciente sin descartar el historial de información a diferencia de la arquitectura del CluStream el cual pregunta al usuario la duración para el agrupamiento, además el D-Stream no requiere que el usuario especifique el número de clústeres k.

  Debido a los grandes volúmenes de datos de un flujo de datos es imposible retener la información de la densidad de cada registro de dato, para esto se propone dividir el espacio en redes finas y discretas así el nuevo registro de datos se hará en la correspondiente red.

Finalmente propone un nuevo concepto la atracción de redes que caracteriza la información de la posición del dato en cada red, lo cual evita que cuando el dato se asigne a la red pierda información de posición.

- En [44] se presenta el sistema ODAC (Online Divisive – Agglomerative Clustering) el cual construye incrementalmente una jerarquía de clústeres basado en un árbol binario usando la estrategia top-down. Es capaz de adaptarse a los cambios en series de tiempo por medio de aglomeración posterior. Una importante característica del sistema es que la matriz de diferencias es actualizada solo para el conjunto de clústeres que actualmente está siendo probado, disminuyendo el número global de diferencias necesarias para ser calculado en cada paso.
- En [45] se propone un algoritmo StreamPredeCon el cual aplica técnicas de clustering a sub-espacios con pesos y referencias. Para esto se hace uso de un factor de decadencia de la Distancia Euclidiana y de la medida de similitud de los pesos y preferencias, de tal modo que el algoritmo pueda capturar los conceptos de “shifting y drifting” cambio y desvío, natural del flujo de datos.

  Inicialmente se calcula el factor de decaimiento y una vez está determinado dicho factor se calcula la distancia entre puntos. Para mantener la información actualizada el modelo realiza un borrado de datos antiguos StreamPredeCon realiza un chequeo de la distancia de decaimiento entre los puntos nuevos y el núcleo de puntos del modelo, si el punto sobrepasa el parámetro del umbral y determina y en ese momento corresponde a un punto con ruido, el punto es eliminado. El paso anterior no solo permite reducir el tamaño y mantener actualizado el modelo sino, además mejorar la eficiencia y precisión en la remoción de puntos antiguos que están lejos de los clústeres en el flujo de datos.

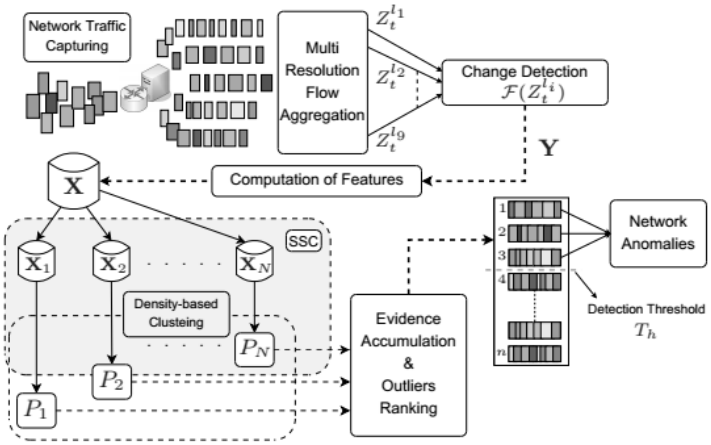
  El modelo de clustering generado por el StreamPredeCon sirve para hacer la clasificación de los puntos de tal forma que cuando un nuevo punto llega el StreamPredeCon calcula el vector de preferencia y observa donde este

nuevo dato puede ser agrupado. Una vez los puntos se han clasificado como datos de ruido o se han asignado a un clúster, el StreamPredeCon clasifica los puntos como núcleos o no-núcleos de puntos. Debido a la anterior clasificación un punto puede pertenecer al conjunto de puntos núcleos o puntos no núcleos, lo cual permite determinar mediante la clasificación binaria si es un problema tal como la detección de un paquete anormal en el flujo de la red. La clasificación de los paquetes de red es evaluada a través de métricas de rendimiento de sensibilidad y una tasa de falsos positivos.

El estudio concluye que los IDS basados en la detección de anomalías permiten descubrir nuevos ataques pero se presenta una alta tasa de falsos positivos y dificulta la detección de ataques polimórficos.

- En [46] se propone el algoritmo UNADA Unsupervised Network Anomaly Detection Algorithm (Algoritmo de Detección de anomalías No supervisado). El algoritmo toma todo el flujo de las IP de entrada en un intervalo de tiempo y clasifica el grado de anormalidad de dichos flujos. Para hacer esto captura los paquetes y los agrega en un flujo de tráfico multi.-resolución que a diferencia de las series de tiempo son construidas en la cima de los flujos para luego con un algoritmo genérico de detección de cambio basado en el análisis de series de tiempo etiquetar los cambios anómalos, después toma como entrada todos los flujos que fueron marcados como anómalos, en éste punto se identifican los flujos de datos atípicos del flujo usando un algoritmo robusto multi-clustering basado en una combinación de Sub-espacios Clustering (SSC), Clustering basados en densidad y la técnica de Evidencia de acumulación de Clústeres (EAC). Con la evidencia proveída por estos algoritmos se hace una clasificación del grado de anormalidad de los flujos atípicos. Finalmente los flujos con el más alto puntaje se marcan como anómalos utilizando un enfoque de detección de umbral, Figura 16.

Figura 16 - Descripción de alto nivel de UNADA



Fuente: Casas, et al. 2011 [46].

2.3. APLICACIÓN DEL ALGORITMO DBSCAN EN LA DETECCIÓN DE ANOMALÍAS DE RED

- **The Anomaly Detection by Using DBSCAN Clustering with Multiple parameters** [47]**:** Proponen un Nuevo algoritmo DBSCAN-MP con múltiples valores para los parámetros Epsilon y MinPts. basados en el algoritmo DBSCAN. Para evaluar el algoritmo utilizan el conjunto de datos KDD Cup 99.

  Se asumen los siguientes puntos para el conjunto de datos:

  - La mayoría de las conexiones de red son de tráfico normal.
  - El tráfico de ataques es estadísticamente diferente del tráfico normal.

  El algoritmo tiene dos etapas, una etapa de entrenamiento y una etapa de detección.

  - **Etapa de entrenamiento:** se crean clústeres con datos de entrenamiento para encontrar un valor de Epsilon y MinPts. para cada clúster respectivamente. Inicialmente usan el algoritmo DBSCAN para crear los primeros clústeres y hallar la distancia entre cada punto, en este momento el valor de Epsilon es el mismo para

todos los clústeres creados. Luego se va haciendo un incremento en el valor del Epsilon y MinPts. para cada clúster y los nuevos puntos encontrados se van añadiendo a los clústeres, si comprueba que estos pueden ser agregados.

Para hallar el valor de MinPts de cada clúster encuentran el número de vecinos para cada punto y el número más pequeño de vecinos es el valor determinado para el parámetro MinPts.

- **Etapa de detección:** asignan nuevos puntos a los clústeres y reportan una alerta cuando hay nuevos puntos en el clúster de puntos anómalos. Un clúster es considerado como anómalo si el tamaño es pequeño y tiene muy pocos puntos, en el caso de los puntos que no han sido asignados a ningún clúster son considerados como puntos inseguros.

  El algoritmo DBSCAN-MP es adecuado para la detección de anomalías de red que contiene muchos tipos de tráfico.

- **A New Semi-unsepervised Intrusion Detection Method Based on Improved DBSCAN** [48]**:** Proponen una mejora del algoritmo DBSCAN llamado algoritmo IIDBG, para la construcción de un modelo de detección de intrusiones de red. El IIDBG es aplicado como motor de detección.

  El sistema de detección de intrusiones basado en minería de datos usa etiquetas del conjunto de datos de entrenamiento para entrenar el motor de detecciones. Para extraer las características ocultas de la actividad de la red, el motor de detección entrenado puede ser usado directamente.

  El IIDBG tiene dos etapas: etapa de entrenamiento y etapa de prueba.

  - Etapa de entrenamiento: Se entrena un conjunto de datos compuesto por un gran número de paquetes de datos de red, el algoritmo IIDBG toma análisis de clústeres para pre-procesarlos y los coloca en el repositorio de conocimientos del IIDBG.

- Etapa de pruebas: El algoritmo toma captura de datos y realiza pre-procesamiento, luego obtiene la etiqueta de los datos utilizando el repositorio de conocimiento IIDBG y genera alarmas para el comportamiento anormal.

IIDBG algoritmo de detección: Para una conexión se pre-procesan los atributos y determina si es el ataque de acuerdo a la definición de accesible por densidad del algoritmo DBSCAN.

- **Anomaly Detection in Data Mining. Hybrid Approach between Filtering and Refinement and DBSCAN** [49]**:** hacen un estudio para comparar los resultados de la combinación de un enfoque híbrido con el algoritmo DBSCAN y los resultados del tradicional algoritmo DBSCAN, con el fin de obtener beneficios de las dos técnicas y minimizar las desventajas de cada una.

  Este método se enfoca en las anomalías cuando intenta detectar anomalías a diferencia de otras técnicas que se enfocan en las instancias normales del conjunto de datos. La técnica se enfoca en dos fases filtrado y refinamiento.

  - **Fase de filtrado:** excluye las instancias normales del conjunto de datos.
  - **Fase de refinamiento**: procesa los datos restantes y analiza el conjunto de datos con medidas basadas en densidad.

  El estudio aplica el algoritmo DBSCAN para el conjunto de datos normal y para el conjunto de datos resultante de la etapa de filtrado se aplica el algoritmo DBSCAN como método de refinamiento.

- **Relative Network Entropy based Clustering Algorithm for Intrusion Detection** [50]**:** Propone un nuevo algoritmo EB-DBSCAN basado en el algoritmo de minería de datos DBSCAN. EB-DBSCAN usa el método de procesamiento de datos por lotes el cual se puede agrupar rápidamente con precisión y sin supervisión para flujos de datos de altas velocidades.

El algoritmo inicialmente lee los datos desde el conjunto de datos agrupados en una ventana de datos, manteniendo la distribución de cada dimensión y continúa buscando los puntos directos accesibles por densidad utilizando la entropía relativa de red como medida de distancia entre dos puntos de datos. Finalmente en cuanto a los puntos de ruido que tiene entropía de red relativa más baja serán los puntos que tienen mayor similitud, por lo que se compara cada punto de ruido con puntos normales y se clasifica cada punto de ruido dentro del clúster al que pertenece el punto normal más cercano. Se establece cada dimensión con atributos dimensión con diferente peso para obtener resultados de agrupamiento óptimos, basados en que diferentes atributos contribuyen a una entropía relativa diferente.

3. METODOLOGÍA

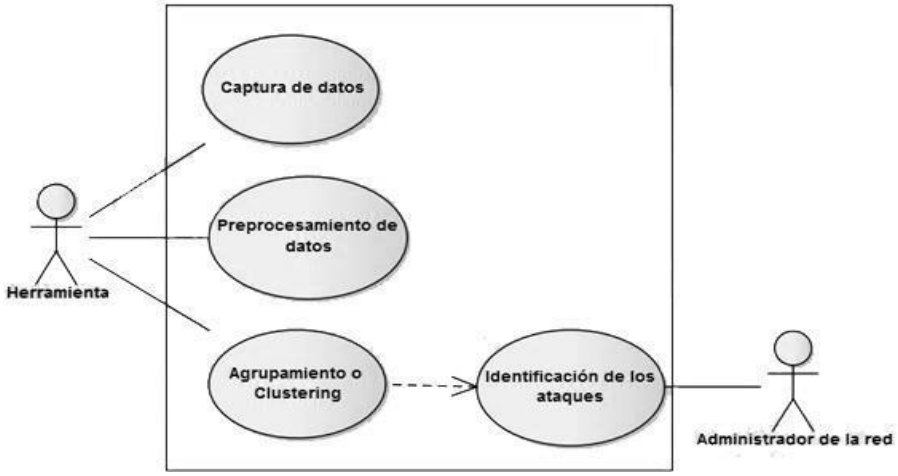
En este capítulo se muestra el modelo planteado para el desarrollo e implementación del sistema de detección de intrusiones de red. Se definen los componentes y la interacción entre ellos y se expone la metodología de desarrollo de la herramienta.

Con base en los conceptos mencionados y los trabajos estudiados en los capítulos anteriores, se identifican las actividades requeridas para la ejecución de una tarea de agrupamiento o clustering aplicado en el modelo del sistema de detección de intrusiones sobre los cuales se basa la herramienta software, Figura 17.

Figura 17 - Actividades para realizar agrupamiento o clustering

Los pasos ilustrados en la Figura 17 y los criterios del administrador de la red permitieron identificar los casos de uso necesarios para el desarrollo de la identificación de intrusiones mediante el modelo y la forma en que este interactúa con el usuario. Los casos de uso se ilustran en la Figura 18.

Figura 18 - Diagrama de casos de uso de la herramienta



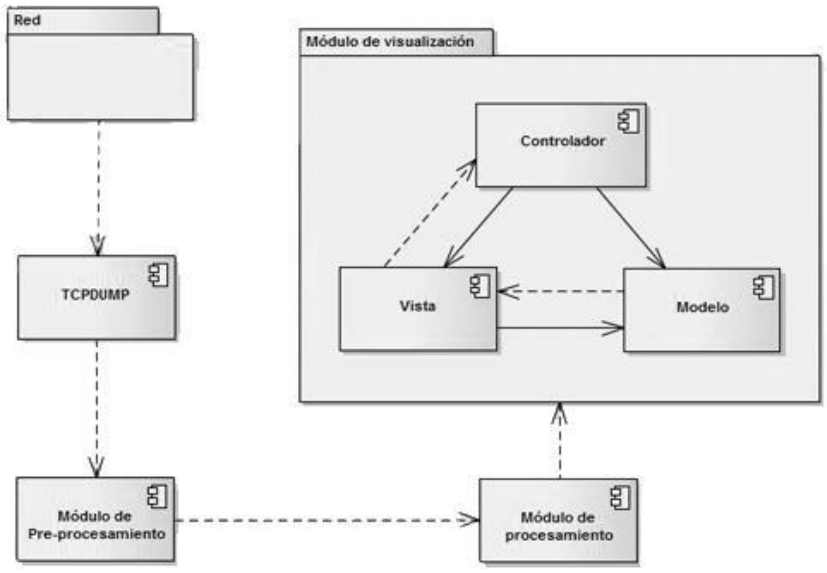
Con base en los requerimientos se identifican 4 componentes básicos para el funcionamiento de la herramienta:

- Componente de captura de paquetes.
- Componente de pre-procesamiento.
- Componente de procesamiento o clustering.
- Componente de visualización de resultados.

Cada uno de estos componentes fue desarrollado de forma separada e interactúa con otro mediante una interfaz para la recepción del flujo de información.

Se usaron dos estilos arquitectónicos para el desarrollo de la aplicación ilustrados en la Figura 19, tubos y filtros para la comunicación entre componentes y modelo-vista-controlador para el desarrollo de la interfaz de usuario y la plataforma de visualización.

Figura 19 - Diagrama de componentes de la herramienta



El estilo arquitectónico de la aplicación permite una alta modularidad de la herramienta y se buscan como requerimientos no funcionales la trazabilidad para llevar un seguimiento continuo y permanente del proceso, la usabilidad para la visualización de los resultados, la eficiencia para el bajo consumo de recursos computacionales y la portabilidad para la verificación del estado de la red en diversos medios o momentos.

Se usó como metodología general de desarrollo el modelo de construcción de prototipos, que corresponde a un modelo evolutivo útil para la programación y la verificación de cada uno de los componentes del software con ayuda de expertos en el tema o interesados mediante reuniones periódicas de prueba y soporte de requerimientos [51].

El modelo de prototipado rápido consta de 5 fases para el desarrollo ilustradas en la Figura 20.

Figura 20 - Fases del modelo de prototipado rápido

Fuente: Pressman, et al. 2009 [51].

Cada una de las fases se define como:

- **Comunicación:** reunión periódica con el administrador de la red para la definición de requerimientos.
- **Plan rápido:** cronograma de desarrollo y planificación de las estructuras generales de la herramienta entre los integrantes del equipo de desarrollo
- **Modelado y diseño rápido**: creación de diagramas UML de las estructuras específicas de la herramienta.
- **Construcción del prototipo:** desarrollo y codificación de la herramienta.
- **Implementación entrega y retroalimentación:** mostrar la herramienta funcional en un estado de pruebas al administrador de la red para la identificación de falencias o posibles mejoras en el período de tiempo estimado.

4. DESARROLLO DE LA HERRAMIENTA

Este capítulo detalla el desarrollo de cada uno de los componentes que conforman la herramienta. Se inicia con el componente de captura de datos, luego se explica el desarrollo del componente de pre-procesamiento y extracción de las características relevantes para la identificación de los ataques planteados, posteriormente se muestra el componente de procesamiento para el cual se hizo uso del algoritmo DBSCAN y finalmente el desarrollo de la interfaz de visualización y la integración de los componentes de la herramienta.

4.1. CAPTURA Y RECEPCIÓN DE LOS DATOS

4.1.1. Descripción de los datos.

Los datos utilizados para evaluar el modelo son producto de una simulación a lo largo de 7 semanas realizada por DARPA (Defense Advanced Research Projects Agency) en el año 1998, para hacer la simulación se creó una red militar ficticia con tres máquinas objetivo ejecutando varios sistemas operativos y servicios y tres máquinas para suplantar direcciones IP que generaran tráfico entre diferentes IP [52]. El conjunto de datos recolectado contiene tráfico normal y 32 tipos de ataques etiquetados entre ellos los ataques tipo DoS SYN-FLOOD y tipo probing SATAN.

4.1.2. Captura de datos

Se simuló un flujo continuo de paquetes de red en un sistema operativo GNU/Linux, para luego proceder a realizar la captura de los datos a analizar.

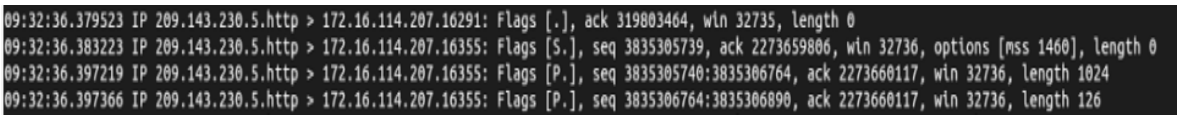
La captura de los paquetes de red se realizó mediante la herramienta de captura de tráfico o sniffer tcpdump. Tcpdump es una herramienta de línea de comandos

para la captura de contenidos de paquetes transmitidos y recibidos en una interfaz de red mediante el uso de una biblioteca llamada libpcap [53].

Para la realización de la captura se probaron otras herramientas de captura de tráfico de red, tales como, Wireshark [54] y Bro IDS [55], sin embargo por facilidad de uso, implementación y compatibilidad con otras herramientas se escogió tcpdump.

Tcpdump realiza la captura de los encabezados del segmento TCP y del datagrama IP, en dichos encabezados se encuentra información relevante de la comunicación, tal como los números de puerto fuente y destino, host o máquina fuente y destino, número de secuencia, número de reconocimiento, tamaño del encabezado, banderas o flags, tamaño de la ventana, etc. En la Figura 21 se muestra el resultado de una captura mediante tcpdump.

Figura 21 - Captura de tráfico de red mediante tcpdump



Muchos investigadores en el modelo de flujos de datos asumen que todos los datos vistos en el flujo son igualmente importantes y todos los modelos, sinopsis o estadísticas deben reflejar el conjunto de datos completo. Sin embargo, esa premisa no es válida para muchas aplicaciones [56]. Para éste trabajo el flujo continuo de datos que se presenta en la red se trata con ventanas de datos deslizantes tomados en períodos de 5 minutos.

El modelo de ventana de datos deslizante consiste en: los elementos llegan a cada instante, cada elemento expira después de exactamente N período de tiempo, y por tanto, la porción de datos que es relevante para realizar las estadísticas y obtener resultados es el conjunto de los elementos que llegaron en el último N período de tiempo [56]. La ventana de datos deslizante hace referencia

a los elementos activos en un instante de tiempo dado y permite considerar los datos recientes y descontar los pasados.

4.2. PRE-PROCESAMIENTO

Cada ventana de datos capturada se almacena temporalmente en un archivo formato tcpdump, la ventana contiene los encabezados del segmento TCP y del datagrama IP.

Es necesario identificar las conexiones realizadas entre pares de host a través de la interfaz de red para obtener las características relevantes de cada una mediante el análisis de los encabezados nombrados anteriormente. Este proceso se logró mediante el uso de la herramienta tcptrace y un script programado en Bash para ser ejecutado en consola GNU/Linux, el cual toma la ventana de datos almacenada anteriormente y realiza un análisis de la conexiones.

Una conexión es definida como una secuencia de paquetes TCP de inicio y finalización en momentos bien definidos, entre los cuales los flujos de datos van a partir de una dirección IP de origen a una dirección IP destino bajo algún protocolo bien definido [57].

Tcptrace es una herramienta de análisis de archivos en formato tcpdump que procesa todos los segmentos TCP presentes en el archivo resultante de la captura de tráfico de red y otorga datos característicos de cada conexión vista Figura 22 tales como, el tiempo, número de bytes enviados y recibidos, retransmisiones, etc. [58]. Otras herramientas alternativas para el análisis de tráfico de red son Wireshark, Bro IDS, SNORT [27], entre otros.

Figura 22 - Resumen de conexión mediante tcptrace

```
TCP connection 2:
        host c:        sloan.lander.edu:1230
        host d:        205.153.63.238:23
        complete conn: yes
        first packet:  Wed Aug 11 11:23:25.151274 1999
        last packet:   Wed Aug 11 11:23:53.638124 1999
        elapsed time:  0:00:28.486850
        total packets: 160
        filename:      telnet.trace
   c->d:                              d->c:
     total packets:            96           total packets:            64
     ack pkts sent:            95           ack pkts sent:            64
     pure acks sent:           39           pure acks sent:           10
     unique bytes sent:       119           unique bytes sent:      1197
     actual data pkts:         55           actual data pkts:         52
     actual data bytes:       119           actual data bytes:      1197
     rexmt data pkts:           0           rexmt data pkts:           0
     rexmt data bytes:          0           rexmt data bytes:          0
     outoforder pkts:           0           outoforder pkts:           0
     pushed data pkts:         55           pushed data pkts:         52
     SYN/FIN pkts sent:       1/1           SYN/FIN pkts sent:       1/1
     mss requested:         1460 bytes      mss requested:         1460 bytes
     max segm size:           15 bytes      max segm size:          959 bytes
     min segm size:            1 bytes      min segm size:            1 bytes
     avg segm size:            2 bytes      avg segm size:           23 bytes
     max win adv:           8760 bytes      max win adv:          17520 bytes
     min win adv:           7563 bytes      min win adv:          17505 bytes
     zero win adv:             0 times      zero win adv:             0 times
     avg win adv:           7953 bytes      avg win adv:          17519 bytes
     initial window:          15 bytes      initial window:           3 bytes
     initial window:           1 pkts       initial window:           1 pkts
     ttl stream length:      119 bytes      ttl stream length:     1197 bytes
     missed data:              0 bytes      missed data:              0 bytes
     truncated data:           1 bytes      truncated data:        1013 bytes
     truncated packets:        1 pkts       truncated packets:        7 pkts
     data xmit time:      28.479 secs       data xmit time:      27.446 secs
     idletime max:        6508.6 ms         idletime max:        6709.0 ms
     throughput:               4 Bps        throughput:              42 Bps
```

Una vez se obtuvo el resumen de las conexiones se seleccionaron las características importantes para la detección de los ataques SYN-FLOOD y SATAN, en este caso, se seleccionaron 18 características para la identificación de dichos ataques mostrados en la Tabla 1, 4 características son de tipo cadena de caracteres o string, utilizadas para la identificación de la máquina víctima y la máquina atacante y las 14 características restantes son de tipo continuo las cuales se utilizaron para determinar el tipo de ataque y el comportamiento de la conexión.

Tabla 1 – Atributos de conexión seleccionados

Atributo	Descripción	Tipo
IpFuente	Dirección IP del host fuente.	String
puertoFuente	Número del puerto del host Fuente	String
ipDestino	Dirección IP del host Destino	String
puertoDestino	Número del puerto del host Destino	String
paq_ft_dest	Cantidad de paquetes enviados de Fuente a Destino	Continuo
paq_dest_ft	Cantidad de paquetes enviados de Destino a Fuente	Continuo
ack_ft_dest	Cantidad de paquetes de confirmación de Fuente a Destino	Continuo
ack_dest_ft	Cantidad de paquetes de confirmación de Destino a Fuente	Continuo
bytes_ft_dest	Cantidad de bytes que fluyen de Fuente a Destino	Continuo
bytes_dest_ft	Cantidad de bytes que fluyen de Destino a Fuente	Continuo
retrans_ft_dest	Cantidad de paquetes retransmitidos de Fuente a Destino	Continuo
retrans_dest_ft	Cantidad de paquetes retransmitidos de Destino a Fuente	Continuo
push_ft_dest	Cantidad de paquetes con el flag PUSH de Fuente a Destino	Continuo
push_dest_ft	Cantidad de paquetes con el flag PUSH de Destino a Fuente	Continuo
syn_ft_dest	Cantidad de paquetes con el flag SYN de Fuente a Destino	Continuo
syn_dest_ft	Cantidad de paquetes con el flag SYN de Destino a Fuente	Continuo
Fin_ft_dest	Cantidad de paquetes con el flag FIN de Fuente a Destino	Continuo
fin_ dest_ft	Cantidad de paquetes con el flag FIN de Destino a Fuente	Continuo

Finalmente se generó un archivo en formato ARFF (Atributte-Relation File Format) que presenta al usuario una vista consistente de los datos Figura 23. Este formato

es soportado por Weka, MOA y ELKI, los cuales son ambientes para el desarrollo y aplicación de algoritmos de minería de datos.

Figura 23 - Datos en formato ARFF

```
@relation Tcptrace
@attribute ipFuente STRING
@attribute puertoFuente STRING
@attribute IPdestino STRING
@attribute puertoDestino STRING
@attribute cant_paquetes_fuente_destino INTEGER
@attribute cant_paquetes_destino_fuente INTEGER
@attribute cant_ack_fuente_destino INTEGER
@attribute cant_ack_destino_fuente INTEGER
@attribute cant_bytes_fuente_destino INTEGER
@attribute cant_bytes_destino_fuente INTEGER
@attribute cant_paquetes_retrans_fuente_destino INTEGER
@attribute cant_paquetes_retrans_destino_fuente INTEGER
@attribute cant_paquetes_push_fuente_destino INTEGER
@attribute cant_paquetes_push_destino_fuente INTEGER
@attribute cant_syn_paquetes_fuente_destino INTEGER
@attribute cant_syn_paquetes_destino_fuente INTEGER
@attribute cant_fin_paquetes_fuente_destino INTEGER
@attribute cant_fin_paquetes_destino_fuente INTEGER
@attribute land {1,0}
@data
172.16.117.111,24295,host60.overdrive.com,80,3,4,3,4,0,2401,0,0,0,1,0,0,1,1,0
172.16.117.111,24358,host60.overdrive.com,80,5,5,4,5,211,1209,0,0,1,2,1,1,1,1,0
172.16.117.111,24420,host60.overdrive.com,80,5,4,4,4,210,267,0,0,1,1,1,1,1,1,0
172.16.117.111,24421,host60.overdrive.com,80,6,7,5,7,210,3904,0,0,1,1,1,1,1,1,0
172.16.117.111,24422,host60.overdrive.com,80,8,15,7,15,200,16773,0,0,1,1,1,1,1,1,0
172.16.117.111,24485,host60.overdrive.com,80,5,5,4,5,213,1293,0,0,1,2,1,1,1,1,0
172.16.116.44,24549,204.120.165.55,80,6,6,5,6,165,3748,0,0,1,1,1,1,1,1,0
172.16.116.44,24602,204.120.165.55,80,10,24,9,24,216,28808,0,0,1,1,1,1,1,1,0
172.16.116.44,24603,204.120.165.55,80,8,12,7,12,214,12596,0,0,1,1,1,1,1,1,0
172.16.116.44,24604,204.120.165.55,80,7,11,6,11,217,10622,0,0,1,1,1,1,1,1,0
135.13.216.191,4317,172.16.112.50,23,2737,1387,2737,1387,1403,1844,0,0,1358,1379,0,0,0,0,0
```

El formato ARFF define un conjunto de datos en términos de una relación o una tabla formada por atributos o columnas de datos. La información sobre los nombres de la relación y los tipos de datos se almacenan en la cabecera ARFF, con los ejemplos o instancias de datos que están siendo representados como filas de datos en el cuerpo del archivo ARFF [59].

4.3. PROCESAMIENTO

Existe una alta variabilidad en el tráfico que fluye en una red en períodos cortos de tiempo, el envío o solicitud de información, el uso de aplicaciones o servicios que dependan de la red generan tráfico diferente y con un alto nivel de entropía. Estos cambios dan lugar a la presencia de ruido y valores atípicos lo cual en un proceso de agrupamiento o clustering genera grupos de formas irregulares.

La técnica de clustering basado en densidad es idónea para tratar conjuntos de datos que presenten niveles de ruido, teniendo en cuenta esta ventaja se eligió el algoritmo DBSCAN basado en densidad (Density Based Spatial Clustering of Applications with Noise) [60], el cual intenta identificar las regiones densas del espacio multidimensional y separarlas de otras regiones densas de acuerdo a características propias de los datos, ver Anexo 1.

Para el desarrollo del proyecto se utilizó el algoritmo DBSCAN implementado en el ambiente de desarrollo de aplicaciones KDD llamado ELKI [61].

ELKI es un ambiente software para la evaluación de algoritmos Clustering de sub-espacio, cuyo objetivo es el descubrimiento de conocimiento en bases de datos (KDD, minería de datos) que permite el desarrollo y la evaluación de algoritmos avanzados de minería de datos y la interacción con estructuras de índices de bases de datos.

El algoritmo DBSCAN depende de dos parámetros Epsilon que corresponde al radio de búsqueda y MinPts que denota un número mínimo de vecinos en el círculo de búsqueda para la conformación de clústeres, estos parámetros fueron calculados a partir de un proceso iterativo sobre las ventanas de tiempo para distintos valores de dichos parámetros, buscando la mayor precisión en la definición de los clústeres. Adicionalmente para la ejecución correcta del algoritmo se estableció el método ideal para el cálculo de distancias entre los puntos que

pertenecen o no a un clúster. Este procedimiento se hizo mediante el método de la distancia Euclidiana, que corresponde al método más común para la solución de problemas de clustering o agrupamiento [10].

Para la evaluación de la tarea de Clustering ELKI tiene en cuenta las siguientes medidas de evaluación:

- Pair Counting Measures: Evalúan la similitud entre dos resultados de agrupamiento mediante el examen de la probabilidad que existe para agrupar un par de puntos de datos en conjunto o separarlos en diferentes clústeres.
- Set Matching Measures: Buscan una coincidencia entre grupos y clases y calcula la medida basada en la superposición entre grupos y clases.
- Entropy-Based Measures: Comparan dos soluciones de clustering.

Una vez el agrupamiento o clusterización de las conexiones se ha realizado se obtienen los clústeres en archivos con formato de texto (.txt), los cuales contienen las conexiones de red que la herramienta detecta como ataques. Algunas conexiones de tráfico generado por la red interna pueden ser agrupadas debido a comportamientos similares a ataques que para este caso no se tienen en cuenta ya que sólo se analiza el tráfico originado desde direcciones IP externas a la red, por esta razón, se realizó un filtro a los clústeres para excluir este tipo de tráfico.

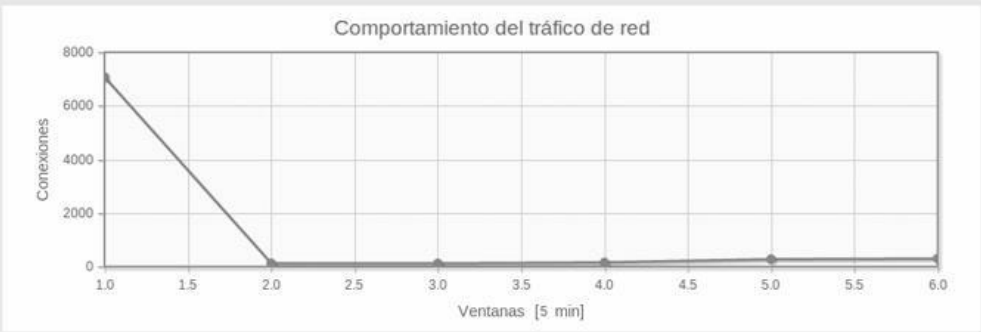
4.4. INTERFAZ DE VISUALIZACIÓN DE LA HERRAMIENTA

Los resultados en el proceso de agrupamiento o clustering son almacenados en disco duro mediante archivos de texto para la ventana actual. Se necesita entonces un componente capaz de abstraer la información importante al administrador de la red y darle una perspectiva global de posibles intrusiones sobre una ventana especifica de tiempo mediante graficas u otras herramientas que otorguen mayor entendimiento de los resultados. Por este motivo se fabrica

una interfaz en lenguaje web en la cual se muestran cuatro elementos para ayudar al administrador de la red en la identificación de intrusiones:

1. Una línea de tiempo que muestra las variaciones en la cantidad de conexiones presentes de una ventana a otra, pues los dos ataques estudiados se caracterizan por generar enormes cantidades de solicitudes y tráfico en cortos periodos de tiempo, Figura 24.

Figura 24 – Línea de Tiempo



2. Un histograma que muestra el número de conexiones en cada clúster hallado en la ventana de tiempo, Figura 25.

Figura 25 – Histograma de Clústeres

3. Una ventana emergente que muestra el detalle de cada clúster presente en el histograma, Figura 26.

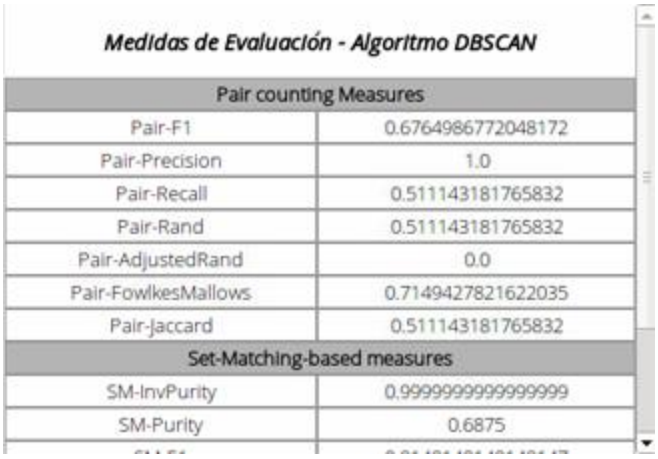
Figura 26 - Ventana emergente con el detalle del clúster

Registros en el Clúster

IP Fuente	Puerto Fuente	IP Destino	Puerto Destino	Paq F->D	Paq D->F	Ack F->D	Ack D->F	Bytes F->D	Bytes D->F	Retrans F->D	Retrans D->F	Push F->D	Push D->F	SYN F->D	SYN D->F	FIN F->D	FIN D->F	Land
172.16.117.132	11514	152.163.210.52	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
172.16.117.111	11519	136.149.142.178	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
172.16.115.234	11515	192.254.26.2	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
172.16.115.234	11526	192.254.26.2	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
172.16.117.132	11529	152.163.210.52	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
172.16.117.132	11716	152.163.212.235	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
172.16.117.132	11779	152.163.210.52	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
172.16.117.132	11841	152.163.210.53	80	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

4. Una sección que muestra valores para la evaluación de la calidad de los clústeres hallados, Figura 27.

Figura 27 – Medidas de evaluación de la calidad de los clústeres



Medidas de Evaluación - Algoritmo DBSCAN

Pair counting Measures	
Pair-F1	0.6764986772048172
Pair-Precision	1.0
Pair-Recall	0.511143181765832
Pair-Rand	0.511143181765832
Pair-AdjustedRand	0.0
Pair-FowlkesMallows	0.7149427821622035
Pair-Jaccard	0.511143181765832
Set-Matching-based measures	
SM-InvPurity	0.9999999999999999
SM-Purity	0.6875

5. Una notificación de escritorio que se activa cada vez que se halla al menos un clúster en una ventana de tiempo determinada, puesto que se asume que el administrador de la red está inmerso en otras tareas diferentes a la supervisión del software, Figura 28.

Figura 28 – Notificación de escritorio

Para el desarrollo de la interfaz se usa la arquitectura MVC (Modelo Vista Controlador) que interactúa a través del sistema de archivos con los componentes de captura, pre-procesamiento y procesamiento de la herramienta. Se usan los lenguajes HTML5, CSS3 y JavaScript para la vista, PHP para el controlador y SQL para el modelo que sirve de almacenamiento temporal para la visualización de resultados.

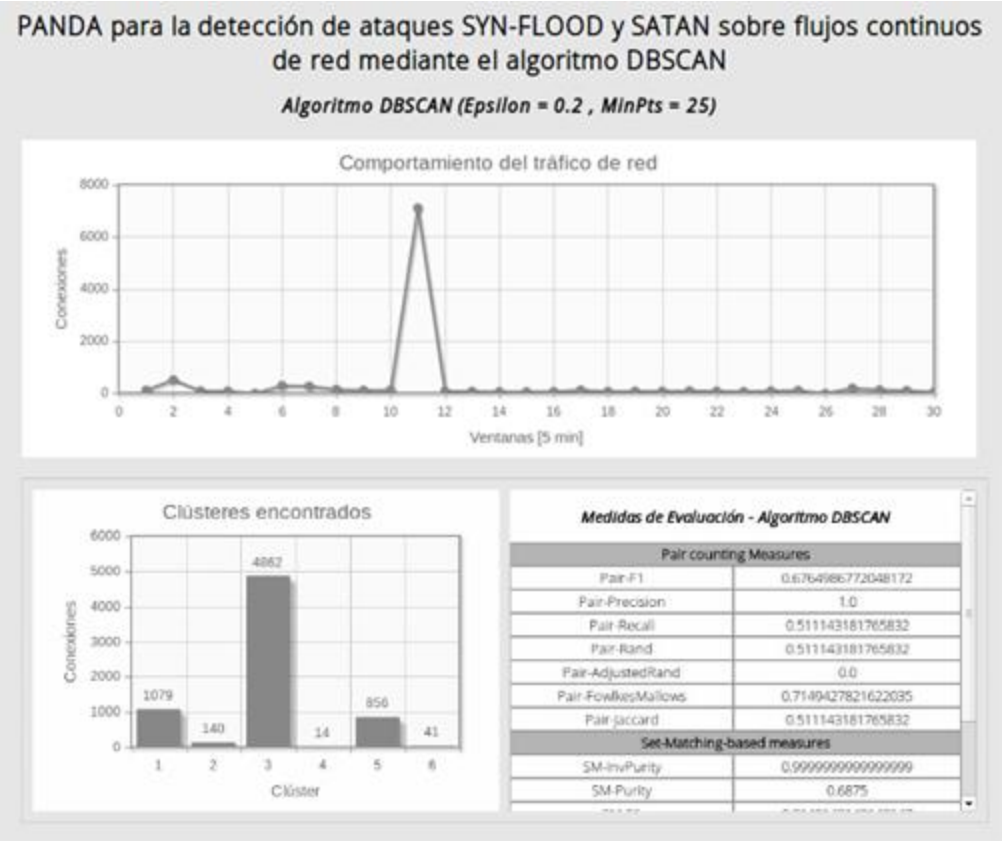
La arquitectura MVC es usada ampliamente en el desarrollo de aplicaciones con interacción cliente-servidor a través de la red mediante diversos dispositivos, consta de tres partes fundamentales [62].

- Modelo: es la representación de la información para el dominio específico de la aplicación.
- Vista: es la representación gráfica del modelo sobre la cual interactúa el usuario.
- Controlador: es la interfaz que interactúa con el modelo, la vista y los dispositivos de entrada.

Se busca cumplir con los requerimientos no funcionales de usabilidad y portabilidad mediante el desarrollo de la interfaz de visualización, asegurando no solo el fácil entendimiento y comprensión de la herramienta por parte de los usuarios, sino también la posibilidad de usarla en diversos dispositivos y computadores sin la necesidad de una previa instalación.

Se visualiza entonces en la Figura 29 el resultado del diseño de la herramienta:

Figura 29 – Interfaz desarrollada para la herramienta PANDA.



Para la presentación de los clústeres y sus contenidos se crearon dos entidades en el motor de base de datos MySQL como medio de almacenamiento temporal durante el tiempo de vida de la ventana, puesto que el almacenamiento permanente de los resultados implican un alto consumo de memoria en disco duro. Las entidades son las siguientes, ver Figura 30:

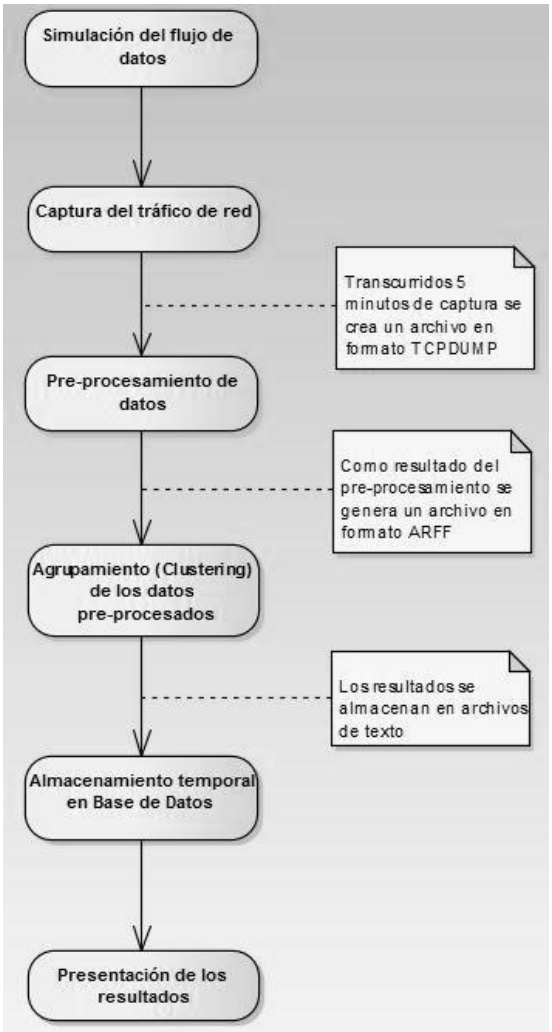
- Entidad “conexiones”: contiene el número de conexiones presente en una ventana de tiempo determinada. Es un único registro por ventana y esta información permanece en memoria para la realización de la línea de tiempo.
- Entidad “registros”: contiene las conexiones pertenecientes a los clústeres hallados en la última ventana de tiempo.

Figura 30 – Entidades "conexiones" y "registros"

4.5. INTEGRACIÓN DE LOS COMPONENTES DE LA HERRAMIENTA PANDA

Cada uno de los componentes del software fue desarrollado y sometido a pruebas independientemente. En esta etapa se realizó la integración de todos componentes en una herramienta única con ayuda de la plataforma web para la sincronización de las tareas. A continuación en la Figura 31 se presenta el diagrama de actividades del modelo.

Figura 31 – Diagrama de actividades

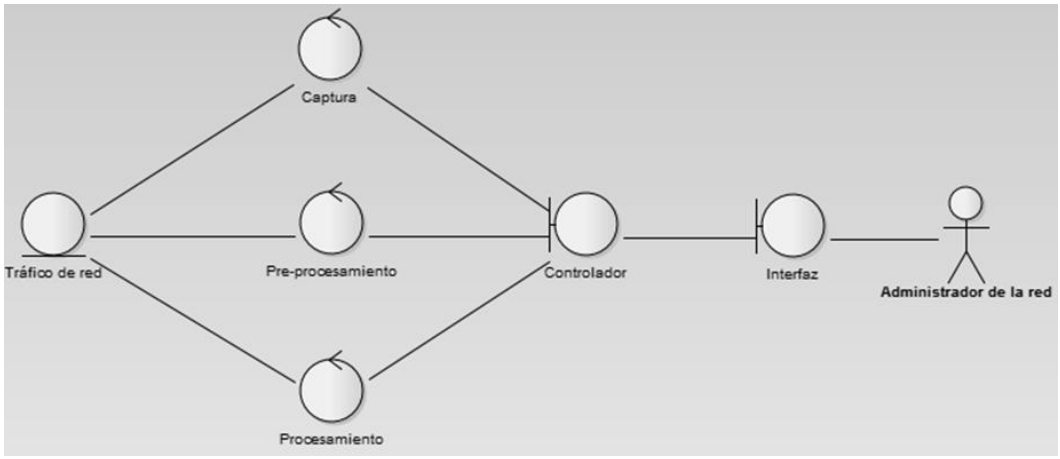


Este procedimiento es iterativo y permanece en ejecución mientras exista un flujo entrante de tráfico de red. La tarea de captura genera un archivo en formato TCPDUMP al cumplirse 5 minutos (ventana), en los siguientes 5 minutos se realizan la demás actividades mencionadas que es el mismo tiempo en que se captura la siguiente ventana, de este modo no hay apilamiento del tráfico capturado.

Las actividades de captura, pre-procesamiento y procesamiento generan archivos en formatos diferentes y almacenados temporalmente en disco, es mediante estos archivos que se comunican los diversos componentes desarrollados, es decir,

estos son el puente de un componente a otro. El diagrama ilustrado en la Figura 32 muestra la forma como se comunican los componentes.

Figura 32 – Diagrama de comunicación



Después de finalizado el desarrollo y la implementación de la herramienta PANDA se procede a la ejecución continua durante 8 días sobre un flujo continuo de paquetes de red simulados mediante el componente de captura de datos, a partir del conjunto DARPA'98. Los resultados se muestran y se analizan en el siguiente capítulo con el fin de medir la efectividad del modelo en la detección de los ataques SYN-FLOOD y SATAN.

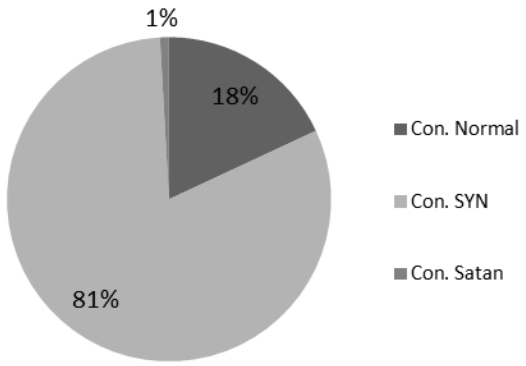
5. ANÁLISIS DE RESULTADOS

Para la evaluación del modelo se capturaron un total de 2277 ventanas de datos cada una de 5 minutos, las cuales contenían 1’639.587 instancias con tráfico de red, de las cuales 296.069 son tráfico normal, 1’328.784 son ataques SYN-FLOOD y 14.734 son ataques SATAN.

La ejecución del componente de procesamiento tarda aproximadamente 0.053 segundos en una ventana de 5 minutos con 175 conexiones en promedio y un máximo de 6 segundos en ventanas de 5 minutos con más de 7000 conexiones.

Una vez realizado el procesamiento con el algoritmo DBSCAN, y el proceso de filtración del tráfico generado por la red interna, se obtuvieron 184 clústeres en 159 ventanas, de los cuales 167 clústeres con 1’334.128 instancias agruparon las conexiones sospechosas de contener el ataque SYN-FLOOD y 17 clústeres con 15.904 instancias sospechas de contener el ataque SATAN, como se ilustra en la Figura 33.

Figura 33 - Total conexiones



A partir de los resultados obtenidos por el componente de procesamiento se observa que los ataques SYN-FLOOD se caracterizan por el alto número de

paquetes SYN enviados a un host en cortos períodos de tiempo sin obtener respuesta ACK. Las solicitudes de conexión SYN son enviadas desde una misma dirección IP remota pero desde diferentes puertos hacia diversos puertos en la máquina objetivo. En la muestra presentada en la Tabla 2 se evidencia la presencia del ataque SYN-FLOOD durante la ventana de tiempo 248.

Tabla 2 - Ataque SYN-FLOOD detectado durante la ventana de tiempo 248

Ip Fuente	Pto Fuente	Ip Destino	Pto Destino	# paq F->D	# ack D->F	# syn F->D
10.20.30.40	48652	172.16.112.50	1	1.0	0.0	1.0
10.20.30.40	53772	172.16.112.50	exec	1.0	0.0	1.0
10.20.30.40	59148	172.16.112.50	3	1.0	0.0	1.0
10.20.30.40	64524	172.16.112.50	4	1.0	0.0	1.0
10.20.30.40	4365	172.16.112.50	5	1.0	0.0	1.0
10.20.30.40	8973	172.16.112.50	6	1.0	0.0	1.0

Este ataque se prolonga durante 10 ventanas de tiempo y presentó el mismo comportamiento ya descrito. El número inusual de conexiones con características similares permitió que el algoritmo DBSCAN realizara el agrupamiento, el cual detectó el ataque como una anomalía que se diferencia enormemente del tráfico normal.

A diferencia del ataque SYN-FLOOD el ataque SATAN obtiene una respuesta ACK del servidor ante las solicitudes de conexión generadas. El atacante establece una conexión desde y hasta múltiples puertos en la máquina mediante diversos protocolos a nivel de aplicación. Su objetivo es buscar la respuesta de cada una de las solicitudes realizadas y así encontrar vulnerabilidades o puertas de acceso a la máquina objetivo. Al igual que SYN-FLOOD existe un alto número de conexiones en un corto período de tiempo, aunque no una cantidad de solicitudes de conexión tan elevada pues no se busca saturar la capacidad de respuesta de la víctima. En la Tabla 3 se muestra el comportamiento de un ataque SATAN durante la ventana de tiempo 933.

Tabla 3 - Ataque SATAN detectado durante la ventana de tiempo 933

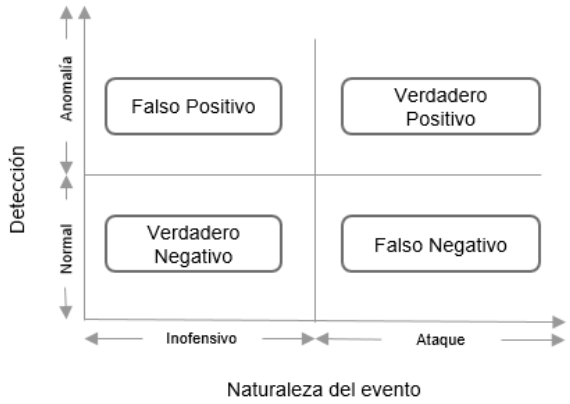
Ip Fuente	Pto Fuente	Ip Destino	Pto Destino	# paq F->D	# ack D->F	# syn F->D
192.168.1.10	4385	172.16.113.50	1	1.0	1.0	1.0
192.168.1.10	4386	172.16.113.50	exec	1.0	1.0	1.0
192.168.1.10	4387	172.16.113.50	3	1.0	1.0	1.0
192.168.1.10	4394	172.16.113.50	4	1.0	1.0	1.0
192.168.1.10	4456	172.16.113.50	5	1.0	1.0	1.0
192.168.1.10	4457	172.16.113.50	6	1.0	1.0	1.0
192.168.1.10	4462	172.16.113.50	8	1.0	1.0	1.0
192.168.1.10	4479	172.16.113.50	10	1.0	1.0	1.0
192.168.1.10	4480	172.16.113.50	11	1.0	1.0	1.0
192.168.1.10	4481	172.16.113.50	12	1.0	1.0	1.0

Este ataque es agrupado por el algoritmo DBSCAN gracias a su comportamiento similar y repetitivo a nivel de la capa de transporte lo cual permite su identificación ya que el tráfico normal se caracteriza por una alta variabilidad presente en las características seleccionadas para la conexión.

Para medir la efectividad de la herramienta se usan 4 indicadores ilustrados en la Figura 34, importantes para medir la efectividad del modelo:

a) *Falso positivo (FP)*, conocido como falsa alarma, corresponde a un evento anómalo que resulta inofensivo desde el punto de vista de la seguridad.
b) *Verdadero positivo (TP)*: que corresponde a la detección satisfactoria de los ataques.
c) *Falso negativo (FN)*: Ataques no detectados ya que muestran un comportamiento suficientemente parecido a los eventos normales.
d) *Verdadero negativo (TN)*: Eventos inofensivos que se han etiquetado como normal ya que se ajustan al modelo de normalidad usado.

Figura 34 – Criterios de evaluación de la herramienta



Con base en estos criterios se obtuvo el siguiente panorama (Tabla 4):

Tabla 4 – Indicadores de evaluación

Indicadores de evaluación						
Ataque	# FP	# FN	# TN	# TP	P	N
SYN-FLOOD	12.263	6.919	298.540	1.321.865	1.328.784	310.803
SATAN	1.739	569	1.623.114	14.165	14.734	1.624.853

A partir de los indicadores se calcularon las siguientes medidas (Tabla 5):

a) Rata de Verdaderos Positivos (TPR): Es el porcentaje de ataques correctamente clasificados.

b) Rata de Falsos Positivos (FPR): Es el porcentaje de ataques incorrectamente clasificados.

c) Rata de Verdaderos Negativos (TNR): Es el porcentaje de tráfico normal correctamente clasificado.

d) Rata de Falsos Descubrimientos (FDR): Es el porcentaje de tráfico normal incorrectamente clasificado.

e) *Valor Predictivo Negativo (NPV):* Es la probabilidad de que el tráfico normal sea correctamente clasificado.

f) *Valor Predictivo Positivo (PPV):* Es la probabilidad de que los ataques sean correctamente clasificados.

g) Exactitud (ACC): Es la medida de clasificar correctamente el tráfico.

Tabla 5 - Medidas de evaluación del modelo

Medidas de efectividad							
Ataque	TPR	FPR	TNR	FDR	NPV	PPV	ACC
SYN-FLOOD	99,48%	3,95%	96,05%	0,92%	97,73%	98,60%	98,83%
SATAN	96,14%	0,11%	99,89%	10,93%	99,96%	85,20%	99,86%

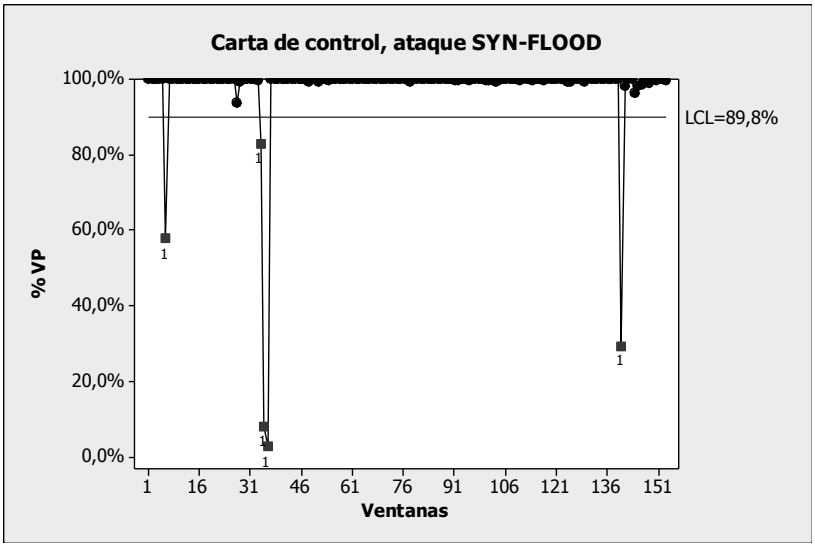
A partir de la Tabla 5 se concluye:

- La herramienta tiene una probabilidad del 98.6% de detectar correctamente un ataque (PPV) SYN-FLOOD y una probabilidad del 85.2% de detectar correctamente un ataque SATAN.
- Se observa una mayor rata de falsos positivos (FDR) encontrados en la clasificación del ataque SATAN comparado con el ataque SYN-FLOOD, reflejado en el FDR con un 10.93% para SATAN y 0.92% para SYN-FLOOD. Esta diferencia con el ataque SATAN puede deberse a la ausencia de características de conexión basadas en el tiempo.
- Se observa una mayor rata de falsos negativos (FPR) en la clasificación del ataque SYN-FLOOD en comparación con el ataque SATAN, con una tasa del 3.95% para SYN-FLOOD y de 0.11% para SATAN.
- Se observa que para SYN-FLOOD existe una tasa de clasificación correcta del ataque del 99.48% y para el ataque SATAN una tasa del 96.14%, es decir la herramienta tiene una alta rata de clasificación correcta de dichos ataques (TPR).
- Finalmente se observa que la herramienta tiene una exactitud (ACC) del 98.83% de clasificar correctamente el tráfico en relación al ataque SYN-FLOOD y un 99.86% en relación al ataque SATAN. SATAN obtuvo un valor más elevado de exactitud comparado con el ataque SYN-FLOOD debido a

la baja proporción de conexiones con ataque SATAN que existen dentro del total de conexiones analizadas.

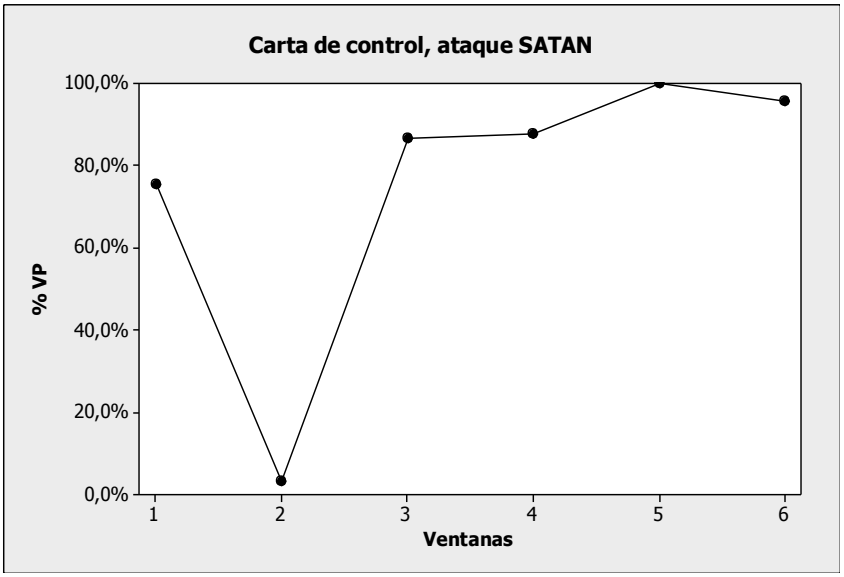
Se realizó un análisis más detallado de las ventanas para el ataque SYN-FLOOD y se hallaron 153 ventanas con ataques, de las cuales solo 5 de ellas tienen una precisión por debajo del 90%. Se observa que la ventana 36 es el caso extremo debido a que solo existen 2.5% de verdaderos positivos con un 97.5% de falsos positivos (Figura 35).

Figura 35 - Carta de control de ataques SYN-FLOOD



La herramienta encontró 6 ventanas con presencia del ataque SATAN agrupados en 17 clústeres. Se observa inicialmente que el volumen de ataques es mínimo en comparación a la proporción de los detectados como ataque SYN-FLOOD. De las 6 ventanas, la ventana 2 presenta un porcentaje extremo de falsos positivos con un 96.9% y un 3.1% de verdaderos positivos (Figura 36).

Figura 36 - Carta de control SATAN



Finalmente se puede concluir del análisis de los resultados que los indicadores de detección de ataques SYN-FLOOD y SATAN son elevados lo cual demuestra la efectividad del modelo planteado.

Cabe resaltar que estos dos ataques identificados son sólo una fracción del alcance del modelo pues la adición de otras características de conexión en la etapa de pre-procesamiento permitiría la identificación de nuevo conocimiento y de una mayor cantidad de comportamientos anómalos y ataques, sin intervención humana.

6. CONCLUSIONES

Gracias a la investigación y el desarrollo de la herramienta se puede concluir que:

- El componente de captura de datos permite la simulación exitosa de un flujo continuo de paquetes de datos y su captura en ventanas de 5 minutos. Este componente logra simular el comportamiento de una red real en un ambiente experimental.
- Las características extraídas en el componente de pre-procesamiento permiten la identificación exitosa de los ataques SYN-FLOOD y SATAN con una alta precisión.
- La extracción y pre-procesamiento de las características de conexión representan el factor determinante para la detección de los ataques con una baja rata de Falsos Positivos y Falsos Negativos. Un mayor número de características extraídas habilita a la herramienta para la identificación de nuevos ataques y de nuevo conocimiento sobre el tráfico de red.
- Las tareas de pre-procesamiento tienen un alto consumo de recursos computacionales, debido a que los datos deben ser tratados como conversaciones o conexiones completas entre pares de hosts, lo cual requiere que el análisis sea realizado sobre flujos de ventanas de tiempo y no sobre registros individuales de envió o recepción de datos o solicitudes.
- El componente de procesamiento permite una ágil tarea de agrupamiento e identificación de los ataques a pesar de la existencia de enormes volúmenes de datos consecuencia de la ocurrencia de los ataques identificados.
- La minería de datos es una técnica idónea para el reconocimiento de patrones en el tráfico de red. Las técnicas de agrupamiento o clustering por densidad no supervisadas son adecuadas para tratar grandes volúmenes

de datos con una alta variabilidad y presencia de ruido como es el caso del tráfico de red.

- La asignación de los valores para los parámetros Epsilon y MinPts representa una tarea delicada ya que mínimas variaciones en su selección pueden producir la obtención de grandes cantidades de Falsos Positivos y Falsos Negativos. A través de la bibliografía se encuentran valores diferentes para estos dos parámetros y su estimación en la mayoría de los casos es empírica.
- Mediante la identificación temprana de ataques de denegación de servicios y de probing se previenen afectaciones a servicios y comunicaciones entre unidades de empresas y organizaciones de sectores públicos y privados. Esta herramienta basada en minería de datos provee al administrador de la red la información esencial para la toma de decisiones y medidas en la detención de ataques en ejecución.
- Los ataques de denegación de servicio de tipo SYN-FLOOD se caracterizan por el alto número de solicitudes de conexión a hosts o redes específicas en un corto período de tiempo, lo que permite su fácil identificación y agrupamiento. Igualmente el ataque de escaneo de vulnerabilidades o probing denominado SATAN se reconoce a partir de su comportamiento altamente repetitivo a nivel de transporte, consecuencia del escaneo realizado a nivel de aplicación.
- El modelo implementado en la herramienta logra la identificación exitosa de los ataques de red SYN-FLOOD y SATAN con una baja tasa de falsos positivos y falsos negativos. Además la herramienta es altamente escalable, portable y modificable, como consecuencia de la alta modularidad y el uso de licencias libres.

7. RECOMENDACIONES Y TRABAJO FUTURO

Se recomienda a futuro la implementación de técnicas o métodos adicionales de pre-procesamiento para el reconocimiento de características de las conexiones a nivel de aplicación aumentando el espectro de ataques reconocidos mediante el algoritmo de agrupamiento. Igualmente se considera que la implementación de métodos de etiquetado de ataques e intrusiones libera cargas en la supervisión del tráfico legítimo para el administrador de la red.

El uso de técnicas adicionales de minería como clasificación y reglas de asociación tendrían un impacto positivo en la toma de decisiones para contrarrestar los ataques. Finalmente se pone a consideración del lector el uso de técnicas que preserven parte de la memoria en puntos relevantes del análisis para mantener un historial del comportamiento de las amenazas a la red y proveer una mejora a la herramienta para la prevención y detección de ataques a futuro.

Referencia bibliográfica

[1] J. Gantz y D. Reinsel, «Extracting value from chaos,» *IDC iView,* pp. 1-12, 2011.

[2] U. Fayyad, G. Piatetsky-Shapiro y P. Smyth, «From data mining to knowledge discovery in databases.,» *AI magazine,* vol. 17, nº 3, p. 37, 1996.

[3] M. Vazirigiannis, M. Halkidi y D. Gunopulos, Uncertainty handling and quality assessment in data mining, Springer, 2003.

[4] S. Chakrabarti, M. Ester, U. Fayyad, J. Gehrke, J. Han, S. Morishita, G. Piatetsky-Shapiro y W. Wang, «Data Mining Curriculum: A Proposal (Version 1.0),» ACM SIGKDD, 2006.

[5] F. Coenen, «Data mining: past, present and future,» *Knowledge Engineering Review,* vol. 26, nº 1, pp. 25-29, 2011.

[6] J. Elder y D. Pregibon, «A statistical perspective on KDD,» *Advances in knowledge discovery and data mining,* pp. 83-116, 1996.

[7] E. Parzen, «Data mining, statistical methods mining, and history of statistics.,» *Computing science and statistics,* pp. 365-374, 1998.

[8] G. Linoff y B. Michael, Data mining techniques: for marketing, sales, and customer relationship management, John Wiley & Sons, 2011.

[9] S. Velickov y D. Solomatine, «Predictive data mining: practical examples,» de *Artificial Intelligence in Civil Engineering. Proceed. 2nd Joint, Workshop*,

Cottbus, Alemania, 2000.

[10] A. Jain y R. Dubes, Algorithms for Clustering Data, Englewood Cliffs: Prentice Hall, 1988.

[11] J. Han, M. Kamber y J. Pei, Data Mining: Concepts and Techniques, Tercera ed., Morgan Kaufmann, 2011.

[12] P.-N. Tan, M. Steinbach y M. Kumar, Introduction to data mining, Pearson Addison-Wesley, 2006.

[13] V. Cerf y R. Kahn, «A Protocol for Packet Network Intercommunication,» *IEEE Trans on Comms,* Vols. %1 de %2Com-22, nº 5, 1974.

[14] V. Cerf, Y. Dalal y C. Sunshine, «Specification of Internet Transmission Control Program,» Diciembre 1974. [En línea]. Available: http://tools.ietf.org/html/rfc675. [Último acceso: 14 05 2013].

[15] J. Postel, «RFC 760,» Enero 1980. [En línea]. Available: http://tools.ietf.org/html/rfc760. [Último acceso: 14 05 2013].

[16] J. Postel, «RFC 791,» Septiembre 1981. [En línea]. Available: http://www.ietf.org/rfc/rfc791.txt. [Último acceso: 14 05 2013].

[17] S. Deering y R. Hinden, «RFC 2460,» Diciembre 1998. [En línea]. Available: http://www.ietf.org/rfc/rfc2460.txt. [Último acceso: 14 05 2013].

[18] R. Braden, «RFC 1122,» Octubre 1989. [En línea]. Available: http://tools.ietf.org/html/rfc1122. [Último acceso: 14 05 2013].

[19] A. Tanenbaum y D. Wetherall, Computer Networks, Pearson Prentice Hall, 2010.

[20] J. Kurkose, Computer Networking: A Top-Down Approach Featuring the Internet, Pearson Education India, 2005.

[21] J. Reynolds, «RFC 3232,» Enero 2002. [En línea]. Available: http://tools.ietf.org/html/rfc3232. [Último acceso: 20 05 2013].

[22] J. Anderson, «Computer security threat monitoring and surveillance.,» James P. Anderson Company, Fort Washington, 1980.

[23] D. Gonzales Gomez, Sistemas de detección de intrusiones, 2003.

[24] H. Vaccaro y G. Liepins, «Detection of anomalous computer session activity,» *Security And Privacy, IEEE,* pp. 280-289, 1989.

[25] E. Spafford, «Crisis and aftermath,» *Communications of the ACM,* vol. 32, nº 6, pp. 678-687, 1989.

[26] L. Ertöz, E. Eilertson, A. Lazarevic, P.-N. Tan, P. Dokas, V. Kumar y J. Srivastava, «Detection and summarization of novel network attacks using data mining,» *Minnesota INtrusion Detection System (MINDS) Technical Report.,* 2003.

[27] M. Roesch, «Snort: Lightweight Intrusion Detection for Networks,» *LISA,* vol. 99, pp. 229-238, 1999.

[28] M. Zaki, «SPADE: An efficient algorithm for mining frequent sequences,» *Machine Learning,* pp. 31-60, 2001.

[29] P. Domingos y G. Hulten, «A general method for scaling up machine learning algorithms and its application to clustering,» *ICML,* pp. 106-113, 2001.

[30] D. Marchette, «A statistical Method for Profiling Network Traffic,» *Workshop on*

Intrusion Detection and Network Monitoring, pp. 119-128, 1999.

[31] S. Guha, N. Mishra, R. Motwani y L. O'Callaghan, «Clustering Data Streams,» *Foundations of computer science, 2000. proceedings. 41st annual symposium on IEEE,* pp. 359-366, 2000.

[32] L. Portnoy, E. Eskin y S. Stolfo, «Intrusion detection with unlabeled data using clustering,» de *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security* , 2001.

[33] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha y R. Motwani, «Streaming-Data Algorithms For High-Quality Clustering,» *Data Engineering, 2002. Proceedings. 18th International Conference,* pp. 685-694, 2002.

[34] C. Ordoñez, «Clustering binary data streams with K-means,» de *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003.

[35] C. Aggarwal, J. Han, J. Wang y P. Yu, «A framework for clustering evolving data streams,» *VLDB '03 Proceedings of the 29th international conference on Very large data bases,* vol. 29, pp. 81-92, 2003.

[36] C. Aggarwal, J. Han, J. Wang y P. Yu, «A framework for projected clustering of high dimensional data streams,» *VLDB '04 Proceedings of the Thirtieth international conference on Very large data bases,* vol. 30, pp. 852-863, 2004.

[37] K. Leung y C. Leckie, «Unsupervised anomaly detection in network intrusion detection using clusters,» *Proceedings of the Twenty-eighth Australasian conference on Computer Science,* vol. 38, pp. 333-342, 2005.

[38] E. Keogh y J. Lin, «Clustering of time-series subsequences is meaningless: implications for previous and future research,» *Knowledge and information*

systems, Vols. %1 de %2154-177, nº 2, p. 8, 2005.

[39] J. Gao, J. Li, Z. Zhang y P.-N. Tan, «An Incremental Data Stream Clustering Algorithm Based on Dense Units Detection,» de *9th Pacific-Asia Conference, PAKDD 2005*, Hanoi, 2005.

[40] F. Cao, M. Ester, W. Quian y A. Zhou, «Density-based clustering over an evolving data stream with noise,» *Proceedings of the 2006 SIAM International Conference on Data Mining,* pp. 328-339.

[41] C. Aggarwal y P. Yu, «A framework for clustering uncertain data streams. En Data Engineering,» *ICDE 2008. IEEE 24th International Conference on IEEE,* pp. 150-159, 2008.

[42] S. Lürh y M. Lazarescu, «Connectivity Based Stream Clustering Using Localised Density Exemplars,» *Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science ,* vol. 5012, pp. 662-672, 2008.

[43] L. Tu y Y. Chen, «Stream data clustering based on grid density and attraction,» *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 3, nº 3, p. 12, 2009.

[44] P. Rodrigues y J. Gama, «Hierarchical clustering of time-series data streams,» *Knowledge and Data Engineering, IEEE Transactions,* vol. 20, nº 5, pp. 615-627, 2008.

[45] Z. Miller y W. Hu, «Data Stream Subspace Clustering for Anomalous Network Packet Detection,» *J. Information Security,* vol. 3, nº 3, pp. 215-223, 2012.

[46] P. Casas, J. Mazel y P. Owezarski, «UNADA: unsupervised network anomaly detection using sub-space outliers ranking,» de *NETWORKING 2011*,

Springer Berlin Heidelberg, 2011, pp. 40-51.

[47] M. Thang y J. Kim, «The Anomaly Detection by Using DBSCAN Clustering with Multiple Parameters,» *Information Science and Applications (ICISA), 2011 International Conference,* 2011.

[48] L. Xue-yong, G. Guo-hong y S. Jia-xia, «A New Intrusion Detection Method Based on Improved DBSCAN,» *2010 WASE International Conference on Information Engineering,* 2010.

[49] S. Handra y H. Ciocârlie , «Anomaly Detection in Data Mining. Hybrid Approach between Filtering and Refinement and DBSCAN,» *Applied Computational Intelligence and Informatics (SACI), 2011 6th IEEE International Symposium,* 2011.

[50] Q. Qian, T. Wang y R. Zhan, «Relative Network Entropy based Clustering Algorithm for Intrusion Detection,» 2013.

[51] R. Pressman y D. Ince, Software engineering: a practitioner's approach, 7 ed., New York: McGraw-hill, 2009.

[52] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham y M. Zissman, «Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation,» *Proceedings DARPA Information Survivability Conference and Exposition (DISCEX) 2000,* vol. 2, pp. 12-26, 2000.

[53] V. Jacobson, C. Leres y S. McCanne, «tcpdump.org,» [En línea]. Available: http://www.tcpdump.org/tcpdump_man.html. [Último acceso: 5 Julio 2013].

[54] G. Combs, «wireshark.org,» [En línea]. Available: http://www.wireshark.org/.

[Último acceso: 5 Julio 2013].

[55] V. Paxson, «bro.org,» [En línea]. Available: http://www.bro.org/. [Último acceso: 5 Julio 2013].

[56] C. Aggarwal, Data Stream Models and Algorithms, Springer, 2007.

[57] G. Kayacık, Z. Heywood y M. Heywood, «Selecting Featuresfor Intrusion Detection:A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets,» *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005),* 2005.

[58] O. Shawn, «tcptrace.org,» [En línea]. Available: http://www.tcptrace.org. [Último acceso: 5 Julio 2013].

[59] S. Garner, «WEKA: The Waikato Environment for Knowledge Analysis,» *Proceedings New Zealand Computer Science Research Students Conference,* 1995.

[60] M. Ester, H.-P. Kriegel, J. Sander y X. Xu, «A density-based algorithm for discovering clusters in large spatial databases with noise,» *KDD,* vol. 96, pp. 226-231, 1996.

[61] H.-P. Kriegel, A. Zimek y E. Achtert, «ELKI: A Software System for Evaluation of Subspace Clustering Algorithms,» *20th International Conference on Scientific and Statistical Database Management (SSDBM 2008),* 2008.

[62] G. Krasner y S. Pope, «A description of the model-view-controller user interface paradigm in the smalltalk-80 system,» *Journal of object oriented programming,* vol. 1, nº 3, 1988.

Bibliografía

Aggarwal, C. (2007). *Data Stream Models and Algorithms.* Springer.

Aggarwal, C., & Yu, P. (2008). A framework for clustering uncertain data streams. En Data Engineering. *ICDE 2008. IEEE 24th International Conference on IEEE*, 150-159.

Aggarwal, C., Han, J., Wang, J., & Yu, P. (2003). A framework for clustering evolving data streams. *VLDB '03 Proceedings of the 29th international conference on Very large data bases, 29*, 81-92.

Aggarwal, C., Han, J., Wang, J., & Yu, P. (2004). A framework for projected clustering of high dimensional data streams. *VLDB '04 Proceedings of the Thirtieth international conference on Very large data bases, 30*, 852-863.

Anderson, J. (1980). *Computer security threat monitoring and surveillance. .* Fort Washington: James P. Anderson Company.

Braden, R. (Octubre de 1989). *RFC 1122*. Recuperado el 14 de 05 de 2013, de http://tools.ietf.org/html/rfc1122

Cao, F., Ester, M., Quian, W., & Zhou, A. (s.f.). Density-based clustering over an evolving data stream with noise. *Proceedings of the 2006 SIAM International Conference on Data Mining*, 328-339.

Casas, P., Mazel, J., & Owezarski, P. (2011). UNADA: unsupervised network anomaly detection using sub-space outliers ranking. En *NETWORKING 2011* (págs. 40-51). Springer Berlin Heidelberg.

Cerf, V., & Kahn, R. (1974). A Protocol for Packet Network Intercommunication. *IEEE Trans on Comms, Com-22*(5).

Cerf, V., Dalal, Y., & Sunshine, C. (Diciembre de 1974). *Specification of Internet Transmission Control Program*. Recuperado el 14 de 05 de 2013, de http://tools.ietf.org/html/rfc675

Chakrabarti, S., Ester, M., Fayyad, U., Gehrke, J., Han, J., Morishita, S., y otros. (2006). *Data Mining Curriculum: A Proposal (Version 1.0).* ACM SIGKDD.

Coenen, F. (2011). Data mining: past, present and future. *Knowledge Engineering Review, 26*(1), 25-29.

Combs, G. (s.f.). *wireshark.org*. Recuperado el 5 de Julio de 2013, de http://www.wireshark.org/

Deering, S., & Hinden, R. (Diciembre de 1998). *RFC 2460*. Recuperado el 14 de 05 de 2013, de http://www.ietf.org/rfc/rfc2460.txt

Domingos, P., & Hulten, G. (2001). A general method for scaling up machine learning algorithms and its application to clustering. *ICML*, 106-113.

Elder, J., & Pregibon, D. (1996). A statistical perspective on KDD. *Advances in knowledge discovery and data mining*, 83-116.

Ertöz, L., Eilertson, E., Lazarevic, A., Tan, P.-N., Dokas, P., Kumar, V., y otros. (2003). Detection and summarization of novel network attacks using data mining. *Minnesota INtrusion Detection System (MINDS) Technical Report.*

Eskin, E., Arnold , A., Prerau, M., Portnoy , L., & Stolfo, S. (2002). A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. *Applications of data mining in computer security* .

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD, 96*, 226-231.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine, 17*(3), 37.

Gantz, J., & Reinsel, D. (2011). Extracting value from chaos. *IDC iView*, 1-12.

Gao, J., Li, J., Zhang, Z., & Tan, P.-N. (2005). An Incremental Data Stream Clustering Algorithm Based on Dense Units Detection. *9th Pacific-Asia Conference, PAKDD 2005.* Hanoi.

Garfinkel, S., & Spafford, G. (1996). *Practical Unix & Internet Security* (Segunda ed.). O'Reilly & Associates.

Garner, S. (1995). WEKA: The Waikato Environment for Knowledge Analysis. *Proceedings New Zealand Computer Science Research Students Conference*.

Garner, S. (1995). WEKA: The Waikato Environment for Knowledge Analysis. *Proc New Zealand Computer Science Research Students Conference*.

Gonzales Gomez, D. (2003). *Sistemas de detección de intrusiones.*

Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2000). Clustering Data Streams. *Foundations of computer science, 2000. proceedings. 41st annual symposium on IEEE*, 359-366.

Hall, M., Witten, I., & Frank, E. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (Tercera ed.). San Francisco, USA: Morgan Kaufmann.

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (Tercera ed.). Morgan Kaufmann.

Handra , S., & Ciocârlie , H. (2011). Anomaly Detection in Data Mining. Hybrid Approach between Filtering and Refinement and DBSCAN. *Applied Computational Intelligence and Informatics (SACI), 2011 6th IEEE International Symposium*.

Jacobson, V., Leres, C., & McCanne, S. (s.f.). *tcpdump.org*. Recuperado el 5 de Julio de 2013, de http://www.tcpdump.org/tcpdump_man.html

Jacobson, V., Leres, C., & McCanne, S. (s.f.). *tcpdump.org*. Recuperado el 15 de Abril de 2013, de http://www.tcpdump.org/tcpdump_man.html

Jain, A., & Dubes, R. (1988). *Algorithms for Clustering Data.* Englewood Cliffs: Prentice Hall.

Kayacik, G., Heywood, Z., & Heywood, M. (2005). Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005)*.

Kayacık, G., Heywood, Z., & Heywood, M. (2005). Selecting Featuresfor Intrusion Detection:A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005)*.

Kendall, K. (1999). *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems.* Tesis de Maestría, Massachusetts Institute of Technology.

Keogh, E., & Lin, J. (2005). Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems, 154-177*(2), 8.

Krasner, G., & Pope, S. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming, 1*(3).

Kriegel, H.-P., Zimek, A., & Achtert, E. (2008). ELKI: A Software System for Evaluation of Subspace Clustering Algorithms. *20th International Conference on Scientific and Statistical Database Management (SSDBM 2008)*.

Kurkose, J. (2005). *Computer Networking: A Top-Down Approach Featuring the Internet.* Pearson Education India.

Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., & Srivastava, J. (2003). A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. *Proceedings of Third SIAM Conference on Data Mining*.

Leung, K., & Leckie, C. (2005). Unsupervised anomaly detection in network intrusion detection using clusters. *Proceedings of the Twenty-eighth Australasian conference on Computer Science, 38*, 333-342.

Linoff, G., & Michael, B. (2011). *Data mining techniques: for marketing, sales, and customer relationship management.* John Wiley & Sons.

Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., y otros. (2000). Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. *Proceedings DARPA Information Survivability Conference and Exposition (DISCEX) 2000, 2*, 12-26.

Lürh, S., & Lazarescu, M. (2008). Connectivity Based Stream Clustering Using Localised Density Exemplars. *Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science , 5012*, 662-672.

Marchette, D. (1999). A statistical Method for Profiling Network Traffic. *Workshop on Intrusion Detection and Network Monitoring*, 119-128.

Miller, Z., & Hu, W. (2012). Data Stream Subspace Clustering for Anomalous Network Packet Detection. *J. Information Security, 3*(3), 215-223.

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

O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., & Motwani, R. (2002). Streaming-Data Algorithms For High-Quality Clustering. *Data Engineering, 2002. Proceedings. 18th International Conference*, 685-694.

Ordoñez, C. (2003). Clustering binary data streams with K-means. *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery.*

Ostermann, S. (s.f.). *tcptrace.org*. Recuperado el 20 de Abril de 2013, de http://www.tcptrace.org

Parzen, E. (1998). Data mining, statistical methods mining, and history of statistics. *Computing science and statistics*, 365-374.

Paxson, V. (s.f.). *bro.org*. Recuperado el 5 de Julio de 2013, de http://www.bro.org/

Portnoy, L., Eskin, E., & Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security .*

Postel, J. (Enero de 1980). *RFC 760*. Recuperado el 14 de 05 de 2013, de http://tools.ietf.org/html/rfc760

Postel, J. (Septiembre de 1981). *RFC 791*. Recuperado el 14 de 05 de 2013, de http://www.ietf.org/rfc/rfc791.txt

Pressman, R., & Ince, D. (2009). *Software engineering: a practitioner's approach* (7 ed.). New York, USA: McGraw-hill.

Prolexic Technologies, Inc. (6 de Abril de 2012). *Prolexic Attack Report Q1 2012.*

Qian, Q., Wang, T., & Zhan, R. (2013). Relative Network Entropy based Clustering Algorithm for Intrusion Detection.

Reynolds, J. (Enero de 2002). *RFC 3232*. Recuperado el 20 de 05 de 2013, de http://tools.ietf.org/html/rfc3232

Rodrigues, P., & Gama, J. (2008). Hierarchical clustering of time-series data streams. *Knowledge and Data Engineering, IEEE Transactions, 20*(5), 615-627.

Roesch, M. (1999). Snort: Lightweight Intrusion Detection for Networks. *LISA, 99*, 229-238.

Shawn, O. (s.f.). *tcptrace.org*. Recuperado el 5 de Julio de 2013, de http://www.tcptrace.org

Spafford, E. (1989). Crisis and aftermath. *Communications of the ACM, 32*(6), 678-687.

Tan, P.-N., Steinbach, M., & Kumar, M. (2006). *Introduction to data mining.* Pearson Addison-Wesley.

Tanenbaum, A., & Wetherall, D. (2010). *Computer Networks.* Pearson Prentice Hall.

Thang , M., & Kim, J. (2011). The Anomaly Detection by Using DBSCAN Clustering with Multiple Parameters. *Information Science and Applications (ICISA), 2011 International Conference*.

Thang, T. M., & Kim, J. (2011). The Anomaly Detection by Using DBSCAN Clustering with Multiple Parameters. *Information Science and Applications (ICISA), 2011 International Conference*.

Tu, L., & Chen, Y. (2009). Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery from Data (TKDD), 3*(3), 12.

Vaccaro, H., & Liepins, G. (1989). Detection of anomalous computer session activity. *Security And Privacy, IEEE*, 280-289.

Vazirgiannis, M., Halkidi, M., & Gunopulos, D. (2003). *Uncertainty Handling and Quality Assesment in Data Mining.* New York: Springer-Verlag.

Vazirigiannis, M., Halkidi, M., & Gunopulos, D. (2003). *Uncertainty handling and quality assessment in data mining.* Springer.

Velickov, S., & Solomatine, D. (2000). Predictive data mining: practical examples. *Artificial Intelligence in Civil Engineering. Proceed. 2nd Joint, Workshop.* Cottbus, Alemania.

Velickov, S., & Solomatine, D. (2000). Predictive Data Mining: Practical Examples. *Artificial Intelligence in Civil Engineering. Proceed. 2nd Joint, Workshop.* Cottbus, Alemania.

Wesley, E. (2007). *TCP SYN Flooding Attacks and Common Mitigations.*

Xue-yong, L., Guo-hong, G., & Jia-xia, S. (2010). A New Intrusion Detection Method Based on Improved DBSCAN.

Xue-yong, L., Guo-hong, G., & Jia-xia, S. (2010). A New Intrusion Detection Method Based on Improved DBSCAN . *2010 WASE International Conference on Information Engineering*.

Zaki, M. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 31-60.

Zimek, A., Kriegel, H.-P., & Achtert, E. (2008). ELKI: A Software System for Evaluation of Subspace Clustering Algorithms. *20th International Conference on Scientific and Statistical Database Management (SSDBM 2008)*.

ANEXO A

DBSCAN fue desarrollado por el Instituto de Ciencias de la Computación en la Universidad de Munich. Algoritmo basado en densidad el cual diseña clústeres de forma arbitraria. DBSCAN requiere del ingreso de un parámetro de entrada y soporte, el valor apropiado para dicho parámetro es determinado por el usuario.

La noción de Clústeres basados en densidad indica que dado un clúster los puntos dentro de él tienen una densidad característica la cuál es considerablemente más alta que la densidad de puntos que están fuera del clúster, es decir a aquellos puntos cuya densidad es más baja se le denomina ruido.

Cada punto de una vecindad dado un radio debe contener al menos un mínimo número de puntos. La forma de la vecindad está determinada por la función de distancia que se elija para dos puntos *a* y *b* denotado *dist(a,b).*

Definición 1: (Eps-Neighborhood de un punto), El Eps-Neighborhood de un punto p, denotado por $N_{Eps}(p)$ es definido por $N_{Eps}(p) = \{q \in D \mid dist(p,q) \leq Eps\}$.

Un enfoque de Naive podría requerir que haya al menos un mínimo de puntos vecinos en un Eps-Neighborhood *(MinPts)* para cada punto que pertenece a un clúster, sin embargo este enfoque falla porque hay dos tipos de puntos en un clúster, los puntos dentro del clúster llamados puntos núcleo o (core points) y los puntos en el borde del clúster (border points).

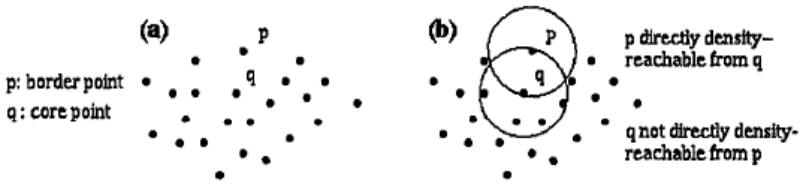
Se requiere que para cada punto p en un cluster C haya un punto q en C tal que p esté dentro de la Eps-neighborhood de q y $N_{Eps}(q)$ contenga al menos un mínimo número de puntos MinPts.

Definición 2: (directamente accesible por densidad) Un punto p es directamente accesible por densidad de un punto $q, Eps, MinPts$ si:

- $p \in N_{Eps}(q)$ y
- $\left|N_{Eps}(q)\right| \geq MinPts\ (Condición\ puntos\ núcleo)$

Directamente accesible por densidad es simétrica para puntos núcleo o core-points. Para el caso en el que sean envueltos un punto borde y un punto núcleo directamente accesible por densidad es asimétrico. Figura 37.

Figura 37 - Puntos núcleo y Puntos borde



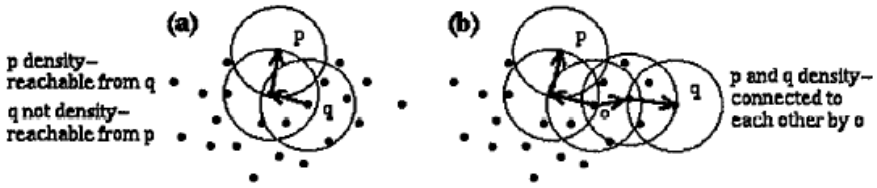
Fuente: Ester, et al. 1996 [63].

Definición 3: (Accesible por densidad) Un punto p es accesible por densidad de un punto $q, Eps\ y\ MinPts$. Si hay una cadena de puntos $p_1, \dots, p_n = q, p_n = p$ tal que p_{i+1} es directamente accesible por densidad de p_i.

Accesible por densidad es una extensión canónica de directamente accesible por densidad, dicha relación es transitiva pero no es simétrica.

Representación de algunas muestra de puntos en particular el caso de la asimetría. Figura 38.

Figura 38 - Densidad accesible y Densidad de conectividad



Fuente: Ester, et al. 1996 [63].

Dos puntos borde de un clúster C posiblemente no es accesible por densidad, porque la condición de los puntos núcleo puede no envolverlos a ambos, sin embargo debe haber un punto núcleo en C en el cuál ambos puntos borde de C son accesibles por densidad.

Definición 4: (Conectado por densidad) Un punto p es conectado por densidad a un punto $q, Eps\ y\ MinPts$ si hay un punto o tal que p y q sean accesibles por densidad de $o, Eps\ y\ MinPts.$

Conectado por densidad es una relación asimétrica. Para puntos accesibles por densidad, la relación de conexión por densidad es también reflexiva.

Definición 5: (Clúster) Sea D una base de datos de puntos. Un clúster $C, Eps\ y\ MinPts.$ es un subconjunto no vacío de D que satisface las siguientes condiciones:

- $\forall\ p, q$: Si $p \in C$ y q es accesible por densidad de $p, Eps\ y\ MinPts$, entonces $q \in C$. (Máximo).
- $\forall\ p, q \in C: p$ es accesible por densidad a $q, Eps\ y\ MinPts.$ (Conectividad)

Definición 6: (Ruido) Sean $C_1, \dots, C_k$ los clústeres de la base de datos D, los parámetros Eps_i y $MinPts.\ i = 1, \dots, k.$ Se define el ruido como el conjunto de puntos en la base de datos D que no pertenecen a ningún cluster C_i, por ejemplo $ruido = \{p \in D \mid \forall i: p \notin C_i\}$.

Un clúster $C, Eps, MinPts.$ contiene un Mínimo número de puntos por las siguientes razones, puesto que C contiene al menos un punto p, p debe ser conectado por densidad a el mismo por algún punto o (el cual debe ser igual a p), por lo tanto o debe satisfacer al menos la condición de punto núcleo y consecuentemente, el Eps-Neighborhood debe contener un mínimo número de puntos.

A continuación dos lemas que se consideran importante para la validación del algoritmo. Dados los parámetros Eps y $MinPts$, se descubre un clúster basado en un enfoque de dos paso. Primero se escoge arbitrariamente un punto de la base de datos que satisfaga la condición de punto núcleo, éste será el punto semilla. Segundo, se recuperan todos los puntos que son accesibles por densidad desde la semilla obteniendo el clúster con la semilla incluida.

Lema 1: Sea p un punto en D y $|N_{Eps}(p)| \geq MinPts.$ entonces el conjunto $O = \{ o|o \in D$ y o es accesible por densidad desde p, Eps y $MinPts.\}$ es un cluster Eps y MinPts.

No es obvio que un clúster C, Eps y $MinPts.$ es únicamente determinado por algún punto núcleo. Sin embargo, cada punto en C es accesible por densidad desde alguno de los puntos núcleo de C y, por lo tanto, un clúster C contiene exactamente los puntos los cuales son accesibles por densidad desde un punto núcleo arbitrario de C.

Lema 2: Sea C un cluster Eps y MinPts y sea p algún punto en C con $|N_{Eps}(p)| \geq MinPts.$ Entonces C igual al conjunto $O = \{o|o$ *es accesible por densidad desde* p, Eps y $MinPts.\}$.

Algoritmo:

1. DBSCAN inicia con la selección arbitraria de un punto p.
2. Recupera todos los puntos accesibles por densidad de p Eps y $MinPts.$
3. Si p es un punto núcleo, se forma un clúster ***(Lema 2)***.

4. Si p es un punto borde, no es un punto accesible por densidad desde p, entonces DBSCAN continua con el siguiente punto en la base de datos.
5. El proceso continúa de igual formas hasta terminar todos los puntos.

DBSCAN usa valores globales para Eps y $MinPts$, basados en lo anterior, DBSCAN puede mezclar dos clústeres en uno solo ***(Definición 5)*** si los dos clústeres de densidad diferente son "cercanos" entre sí.

Sea S_1 y S_2 la distancia entre dos conjuntos de puntos definida como $dist(S_1, S_2) = \min\{dist(p, q) | p \in S_1, q \in S_2\}$, entonces dos conjuntos de puntos que tienen la densidad del clúster más pequeño serán separados el uno del otro solo si la distancia entre los dos conjuntos es más grande que Eps.

A continuación una versión básica del algoritmo DBSCAN. Figura 39.

Figura 39 - Algoritmo DBSCAN

```
DBSCAN (SetOfPoints, Eps, MinPts)

// SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point,
             ClusterId, Eps, MinPts) THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

Fuente: Ester, et al. 1996 [63].

Dónde:

- *SetOfPoints* es la base de datos completa o el descubierto en la anterior ejecución.

- *Eps* y *MinPts.* Son los parámetros de densidad globales determinados manualmente.
- La función *SetOfPoints.get(i)* retorna el i-ésimo elemento de SetOfPoints.

La función más importante usada en el DBSCAN es la función *ExpandCluster:* Figura 40.

Figura 40 - Función ExpandCluster algoritmo DBSCAN

```
ExpandCluster(SetOfPoints, Point, ClId, Eps,
              MinPts) : Boolean;
  seeds:=SetOfPoints.regionQuery(Point,Eps);
  IF seeds.size<MinPts THEN // no core point
    SetOfPoint.changeClId(Point,NOISE);
    RETURN False;
  ELSE   // all points in seeds are density-
         // reachable from Point
    SetOfPoints.changeClIds(seeds,ClId);
    seeds.delete(Point);
    WHILE seeds <> Empty DO
      currentP := seeds.first();
      result := SetOfPoints.regionQuery(currentP,
                                        Eps);
      IF result.size >= MinPts THEN
        FOR i FROM 1 TO result.size DO
          resultP := result.get(i);
          IF resultP.ClId
              IN {UNCLASSIFIED, NOISE} THEN
            IF resultP.ClId = UNCLASSIFIED THEN
              seeds.append(resultP);
            END IF;
            SetOfPoints.changeClId(resultP,ClId);
          END IF; // UNCLASSIFIED or NOISE
        END FOR;
      END IF; // result.size >= MinPts
      seeds.delete(currentP);
    END WHILE; // seeds <> Empty
    RETURN True;
  END IF
END; // ExpandCluster
```

Fuente: Ester, et al. 1996 [63].

Parámetros Eps y MinPts: se presenta una efectiva heurística para determinar los parámetros Eps y $MinPts.$ del cluster mas pequeño en la base de datos.

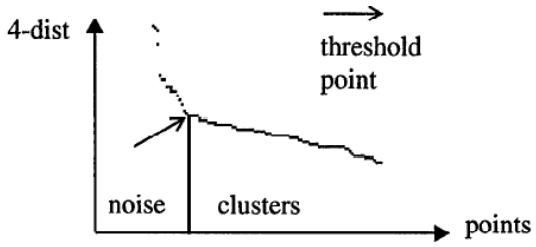
Sea d la distancia de un punto p a su *k-ésimo* vecino más cercano, entonces la *d-vecindad* de p contiene exactamente $k+1$ puntos solo si muchos puntos tienen exactamente la misma distancia d desde p lo cual es muy poco probable. Por lo tanto cambiar k por un punto en un clúster no resulta un gran cambio en d. Eso solo pasa si el *k-ésimo* vecino más cercano de p para $k = 1,2,3, \ldots$ se encuentra sobre una línea recta, lo cual en general no es cierto para un punto en un clúster.

Para un k dado se define una función *k-dist* de la base de datos D a los números reales, posicionando cada punto a la distancia del *k-ésimo* vecino más cercano. Cuando se organizan los elementos de la base de datos en orden descendiente de sus valores *k-dist,* la gráfica de esta función da una idea de la distribución de la densidad de la base de datos, a ésta grafica se le llama *sorted k-dist graph.*

Si se elige un punto p, establecer el parámetro Eps a *k-dist(p)* y establecer el parámetro *MinPts.* Para k, todos los puntos con un valor igual o menor que *k-dist* serán puntos núcleo. Si se pudiera encontrar un *punto límite* con el máximo valor *k-dist* en el clúster más pequeño de D, éste seria el valor del parámtero deseable.

El punto límite es el primer valor en el primer valle de la gráfica sorted k-dist graph. Todos los puntos con el valor más alto del k-dist (a la izquierda del límite) son considerados como ruido, todos los puntos (a la derecha del límite) son asignados a un mismo clúster. Figura 41

Figura 41 - Gráfica sorted 4-dist graph para una muestra de una base de datos



Fuente: Ester, et al. 1996 [63].