aFacultad de Ingeniería

Escuela de Ingeniería Informática

Sistema para la asistencia de configuración de reglas del IDS Snort basado en técnicas heurísticas.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma Firma

Wilmer Pereira Rafael Lara

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma Firma

Andrés Rubio Raúl Tuozzo

# Prologo

# Agradecimiento

# Dedicatoria

Estas 3 son opcional

# Índice

[Prologo 2](#_Toc13048680)

[Agradecimiento 2](#_Toc13048681)

[Dedicatoria 2](#_Toc13048682)

[Índice 3](#_Toc13048683)

[Índice de Tablas y Gráficas 4](#_Toc13048684)

[Sinopsis 5](#_Toc13048685)

[Capítulo I. Introducción 6](#_Toc13048686)

[Capítulo II. Planteamiento del Problema 7](#_Toc13048687)

[Capítulo III. Objetivos 10](#_Toc13048688)

[III.1. Objetivo General 10](#_Toc13048689)

[III.2. Objetivos Específicos 10](#_Toc13048690)

[III.2.1. Diseñar un algoritmo de aprendizaje basado en técnicas heurísticas capaz de reconocer el tráfico propio de una red. 10](#_Toc13048691)

[III.2.2. Diseñar e implantar una base de datos positiva que represente el comportamiento propio de la red. 10](#_Toc13048692)

[III.2.3. Desarrollar el módulo de aprendizaje. 10](#_Toc13048693)

[III.2.4. Desarrollar el módulo de retroalimentación. 11](#_Toc13048694)

[III.2.5. Desarrollar el módulo de traducción. 11](#_Toc13048695)

[III.2.6. Realizar pruebas de ataques en el Sistema de Detección de Intrusos configurado con las reglas producidas por el sistema luego del proceso de aprendizaje. 11](#_Toc13048696)

[Capítulo IV. Alcance y Limitaciones 11](#_Toc13048697)

[IV.1. Diseñar un algoritmo de aprendizaje basado en técnicas heurísticas capaz de reconocer el tráfico propio de una red. 11](#_Toc13048698)

[IV.2. Diseñar e implantar una base de datos positiva que represente el comportamiento propio de la red. 12](#_Toc13048699)

[IV.3. Desarrollar el módulo de aprendizaje. 12](#_Toc13048700)

[IV.4. Desarrollar el módulo de retroalimentación. 13](#_Toc13048701)

[IV.5. Desarrollar el módulo de traducción. 13](#_Toc13048702)

[IV.6. Realizar pruebas de ataques en el Sistema de Detección de Intrusos configurado. 14](#_Toc13048703)

[Capítulo V. Justificación 15](#_Toc13048704)

[Capítulo VI. Marco Referencial 16](#_Toc13048705)

[Capítulo VII. Marco metodológico. 24](#_Toc13048706)

[Capítulo VIII. Desarrollo 26](#_Toc13048707)

[VIII.1. Iteración 1: Captación de paquetes. 26](#_Toc13048708)

[VIII.1.1. Determinación de objetivos. 26](#_Toc13048709)

[VIII.1.2. Investigación. 26](#_Toc13048710)

[VIII.1.3. Análisis de riesgo. 26](#_Toc13048711)

[VIII.1.4. Desarrollo. 26](#_Toc13048712)

[VIII.1.5. Pruebas. 26](#_Toc13048713)

[VIII.1.6. Conclusión. 26](#_Toc13048714)

[VIII.1.7. Planificación. 26](#_Toc13048715)

[VIII.2. Algoritmo de aprendizaje 33](#_Toc13048716)

[Capítulo IX. Resultados 37](#_Toc13048717)

[Capítulo X. Conclusiones 37](#_Toc13048718)

[Capítulo XI. Recomendaciones 37](#_Toc13048719)

[Referencias Bibliográficas 37](#_Toc13048720)

# Índice de Tablas y Gráficas

[Ilustración 1. Muestra de archivo .pcap 31](#_Toc14797591)

[Ilustración 2. Árbol de Telnet. 33](#_Toc14797592)

[Ilustración 3. Árbol de SSH. 34](#_Toc14797593)

[Ilustración 4. Árbol de FTP. 34](#_Toc14797594)

[Ilustración 5. Árbol de HTTP. 35](#_Toc14797595)

[Ilustración 6. Muestra de paquetes PING. 36](#_Toc14797596)

[Ilustración 7. Diagrama de clase de entidades: Módulo de captación de paquetes. 38](#_Toc14797597)

[Ilustración 8. Diagrama de clase de comandos: Módulo de captación de paquetes. 38](#_Toc14797598)

[Ilustración 9. Diagrama Entidad-Relación de la base de datos positiva. 47](#_Toc14797599)

[Ilustración 10. Diagrama de clases DAO. 49](#_Toc14797600)

[Ilustración 11. Diagrama de clase de módulo de traducción. 52](#_Toc14797601)

# Sinopsis

Resumen de 1 pagina de todo el peo.

# Introducción

# Planteamiento del Problema

La resolución de problemas siempre ha sido una de las grandes ambiciones para el ser humano, siempre estamos en búsqueda de técnicas que nos ayuden a mejorar nuestros procesos para obtener mejores soluciones de forma más eficiente, eficaz y efectiva. Pero, en algunos casos, no siempre se puede obtener la mejor solución posible o es muy costoso y se gastarán muchos recursos en conseguirla. Esto ocurre con frecuencia en escenarios de la vida real, especialmente en el área de las matemáticas y en la computación. Es aquí donde entran en juego las técnicas heurísticas, con las cuales es posible conseguir soluciones que se saben que no son óptimas, pero son lo suficientemente buenas para solucionar el problema bajo las restricciones dadas y utilizando una cantidad razonable de recursos.

Existen diversas técnicas heurísticas y toda una clasificación de las mismas. A pesar de esto, se utilizará en primera instancia, los Algoritmos Genéticos como técnica para hallar una solución para el problema que se describe más adelante en esta sección.

“Un algoritmo genético es un método de búsqueda que imita la teoría de la evolución biológica de Darwin para la resolución de problemas. Para ello, se parte de una población inicial de la cual se seleccionan los individuos más capacitados para luego reproducirlos y mutarlos para finalmente obtener la siguiente generación de individuos que estarán más adaptados que la anterior generación”. Extraído de (http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf).

Desde el año 1954 se han estado utilizando algoritmos de programación genética, lo cual es una metodología de la inteligencia artificial que se basa en algoritmos evolutivos para desarrollar automáticamente programas que realicen una tarea definida, todo esto inspirado en la evolución biológica.

Gracias a la necesidad de mantener un sistema o red protegidos contra intrusos, desde el año 1980 se ha estado hablando e implementando los llamados Sistemas de Detección de Intrusos (IDS), los cuales permiten detectar cuando algún intruso quiere violar la seguridad de un sistema.

“Llamaremos intrusión a un conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso”.

Extraído de (https://www.rediris.es/cert/doc/unixsec/node26.html).

Un sistema de detección de intrusos permite bloquear, alertar o permitir distintos tipos de tráfico de datos, dependiendo de las reglas impuestas por el administrador del sistema, las cuales deberían de ajustarse según la necesidad y comportamiento de la red o host.

Debido al creciente número de ataques al transcurrir los años y la creatividad de los atacantes al momento de inventar nuevos ataques que puedan vulnerar la seguridad de un sistema, ha sido necesario implementar un tipo de sistema de seguridad que pueda modificarse y responder a estos ataques, que sea capaz de evolucionar al pasar del tiempo para mantener un nivel de seguridad confiable, es ahí donde entra un sistema de detección de intrusos, el cual sea capaz de modificar y ajustar sus reglas para mantener este nivel de seguridad alto y confiable durante el tiempo.

Partiendo de esta premisa se desarrollará un sistema de asistencia para la configuración de reglas del IDS Snort el cual utilizará técnicas heurísticas, para que sea capaz de aprender sobre el sistema donde estará instalado. Se utilizará la plataforma del IDS Snort para reducir el tiempo de desarrollo y no mal gastarlo en trabajo ya existente. Este programa será capaz de mapear el sistema donde se instale y podrá aprender sobre su comportamiento para así detectar si una actividad realizada es propia o no y a partir de este conocimiento podrá generar una serie de reglas, utilizando la sintaxis de Snort, que sólo permitirá el acceso al tráfico propio de la red quedando así un sistema protegido contra atacantes.

# Objetivos

## Objetivo General

Desarrollar un sistema de asistencia para la configuración de reglas del Sistema de Detección de Intrusos Snort.

## Objetivos Específicos

### Diseñar un algoritmo de aprendizaje basado en técnicas heurísticas capaz de reconocer el tráfico propio de una red.

### Diseñar e implantar una base de datos positiva que represente el comportamiento propio de la red.

### Desarrollar el módulo de aprendizaje.

### Desarrollar el módulo de retroalimentación.

### Desarrollar el módulo de traducción.

### Realizar pruebas de ataques en el Sistema de Detección de Intrusos configurado con las reglas producidas por el sistema luego del proceso de aprendizaje.

# Alcance y Limitaciones

## Diseñar un algoritmo de aprendizaje basado en técnicas heurísticas capaz de reconocer el tráfico propio de una red.

Se diseñará un algoritmo de aprendizaje que utilizando la información proporcionada por el administrador de la red sobre los paquetes recopilados y su confiabilidad, será capaz de generar una función la cual servirá como filtro para determinar qué paquetes son propios de la red.

Las condiciones iniciales estarán conformadas por las funciones que serán aplicadas a los paquetes y sus atributos. El objetivo del algoritmo es conseguir una configuración de funciones que simulan el criterio del administrador al evaluar la confiabilidad de un paquete.

## Diseñar e implantar una base de datos positiva que represente el comportamiento propio de la red.

Esta base de datos relacional contendrá la información producida por la función del algoritmo de aprendizaje y se estructurará de tal forma que sea posible su traducción a reglas de Snort. Adicionalmente se evaluará la necesidad de generar más información para que la data describa el comportamiento propio de la red de la forma más completa.

## Desarrollar el módulo de aprendizaje.

Este módulo será capaz de recopilar todo el tráfico ocurrido en el período de aprendizaje el cual será evaluado por el administrador de la red y será puntuado según la confiabilidad de cada paquete. Lo cual servirá para alimentar el algoritmo y posteriormente generar una base de datos positiva que represente el comportamiento propio de la red.

La precisión en la puntuación de cada paquete dependerá de la experticia y el conocimiento que tenga el Administrador de la red. Este debe conocer el comportamiento de los servicios y componentes que hay en ella para que su criterio de puntuación sea confiable.

Se asumirá que durante la fase de aprendizaje no ocurrirá ningún ataque ya que es cuando el sistema es más vulnerable. Una falla en esta fase puede significar una mala configuración del sistema y por tanto existiría la posibilidad de una brecha en la seguridad.

Se estudiará qué diagramas UML se realizarán dependiendo de los requisitos del proyecto y su ejecución. Se te utilizará la técnica de *Top-Bottom* como estrategia para la realización de dichos diagramas.

## Desarrollar el módulo de retroalimentación.

Este módulo es de vital importancia para la superación del sistema, será diseñado para que sea posible adicionar información nueva que contribuya con la inteligencia del sistema. Ya sea porque se consiguió una forma de ataque o porque se agregó un nuevo módulo en la infraestructura, este módulo podrá volver a iniciar el proceso de aprendizaje utilizando nuevas (y más completas) condiciones iniciales.

## Desarrollar el módulo de traducción.

El sistema deberá ser capaz de traducir la información que se encuentra almacenada en la base de datos positiva a reglas que serán utilizadas por Snort para determinar cuál será el tráfico aceptado.

La información almacenada en la base de datos no debe estar infectada con información concerniente a ataques de red, ya que esto afectaría de manera negativa las reglas generadas, permitiendo así falsos negativos.

## Realizar pruebas de ataques en el Sistema de Detección de Intrusos configurado.

Al finalizar el desarrollo del módulo de aprendizaje es necesario instalarlo en un ambiente con tráfico real, del cual obtendrá la data necesaria para aprender sobre la red. Es importante recalcar que durante este tiempo es de suma importancia la ausencia de ataques al sistema, ya que afectaría la data recopilada, afectando de manera negativa al resultado final.

Se partirá de la base de datos positiva para generar un archivo de configuración utilizando la sintaxis de Snort, estas reglas bloquearan el acceso a todo el tráfico que no esté contenido en la base de datos positiva. Es importante que todo el tráfico que represente los servicios propios del sistema esté contenido dentro de dicha base de datos, ya que en caso contrario se podrían presentar falsos positivos.

Luego de que se genere el archivo de configuración de Snort, se procederá a instalarlo con dichas reglas en el ambiente. Se someterá al sistema a una serie de ataques conocidos y se documentará el comportamiento del IDS para evaluar el nivel de efectividad con el que se defiende de dichos ataques. Además, se observará el sistema para asegurar que no se vea afectado el tráfico normal del mismo.

# Justificación

La seguridad informática es muy cambiante, los atacantes están constantemente buscando formas creativas e ingeniosas para conseguir vulnerabilidades, por lo tanto, es necesario que los sistemas sean más inteligentes que los atacantes, para ello se utilizará el paradigma de programación genética para que el software sea capaz de aprender en su totalidad el comportamiento del sistema a proteger. Dicho paradigma data del año 1954 pero no está siendo implementado tan frecuentemente en los sistemas de detección de intrusos. Lo cual motiva a combinar estos componentes para obtener un producto innovador.

Para la realización de este trabajo, es necesario dominar a profundidad las siguientes áreas de conocimiento:

* Redes y comunicación computacional en ambientes de alto tráfico.
* Metodologías de trabajo relacionadas con el Software.
* Diseño de documentación de Software usando UML para diferentes diagramas que plasmará gráficamente la estructura, arquitectura y comportamiento del producto.
* Cualidades del software para alcanzar la máxima calidad posible.
* Paradigma de programación orientado a objetos.
* Paradigma de resolución de problemas utilizando técnicas heurísticas.
* Patrones de diseño del software.
* Seguridad computacional.
* Diseño, implementación y gestión de bases de datos relacionales.
* Buen desenvolvimiento en ambientes de trabajo en equipo.
* Manejo de repositorios utilizando herramientas de controlador de versiones.
* Evaluación de sistemas Informáticos.

Dichas competencias sólo pueden ser adquiridas a lo largo de la carrera de Ingeniería Informática por lo tanto es imperativo que las personas involucradas en la realización de este Trabajo Especial de Grado hayan cursado la mayoría de las materias del pensum de dicha carrera.

# Marco Referencial

A continuación, se explicarán los conceptos claves que necesitará el lector como conocimiento previo para la comprensión de este trabajo de investigación.

## Seguridad Informática.

La seguridad informática consiste en conjunto de técnicas las cuales son aplicadas en sistemas para protegerlos con el objetivo de prevenir la interrupción del flujo de información, el registro, modificación, destrucción o inspección de datos sensibles, la prevención del acceso y uso no autorizado de un sistema o componente, entre otros ataques que pueden ser de naturaleza más creativa y no estructurada.

Ésta disciplina se basa en los siguientes principios:

* Confidencialidad: proteger la información de receptores no deseados.
* Integridad: mantener la consistencia, integridad y precisión de la información.
* Disponibilidad: asegurar el acceso a los datos y servicios siempre que sea requerido.

El uso de las herramientas informáticas está muy presente en el día a día de las personas y las empresas, por lo tanto, se ha incrementado el valor que en ellas reside. Esto despierta cada vez más el interés de las personas con intenciones maliciosas y trae como consecuencia un número creciente de ataques exitosos, por lo tanto, es importante invertir esfuerzo en proteger los sistemas.

Este esfuerzo mencionado anteriormente debe materializarse en mecanismos que obliguen al atacante a gastar más recursos de los que obtendría de ser exitoso su ataque. Ya que es muy difícil o quizás imposible diseñar e implementar un sistema que sea completamente invulnerable.

Cabe destacar que todo sistema de seguridad informática pierde efectividad al no contar con seguridad física en el lugar geográfico donde se encuentre el hardware correspondiente a la información o servicio que se esté resguardando. Por lo tanto, la seguridad física es indispensable e igual de importante que la seguridad informática.

Existen diferentes tipos de herramientas que nos ayudan a protegernos de actividades maliciosas como, por ejemplo: software de criptografía, firewalls, Los antivirus, *Scanners* de Red, *Sniffers*, VPN, IPS, IDS, entre muchas otras. A continuación, se profundizará sobre IDS ya que es la herramienta en la que se enfoca este trabajo de investigación.

## Sistema de detección de intrusos.

Un sistema de detección de intrusos (IDS) consiste en un sistema capaz de reconocer y alertar sobre los ataques realizados a una red, mediante una serie de reglas impuestas por el administrador de la misma, dependiendo de la configuración del IDS este puede presentar falsos positivos, ocurre cuando un paquete que es propio de la red es identificado erróneamente como un ataque, afectan negativamente al sistema ya que entorpecen su comportamiento normal, y falsos negativos, ocurre cuando un paquete con intenciones maliciosas no es detectado, estos son más graves ya que solo se puede saber de su existencia luego de que el daño ya está hecho.

Existen varios tipos de sistemas de detección de intrusos, los cuales se clasifican según:

### Situación

* IDS de máquina: Verifica si archivos vitales son modificados y revisa históricos del sistema generados por los demonios.
* IDS de red: Detecta intentos de intrusión a nivel de red, realiza un reporte de sus actividades y bloquea intentos de intrusión.

### Estrategia

* Detección de anomalías: Solo conoce el comportamiento “normal” de la red o sistema (todo lo que no se identifique como comportamiento normal será bloqueado o alertado), genera falsos positivos.
* Detección de uso indebido: Solo conoce el comportamiento “indebido” de la red (todo lo que no se identifique como comportamiento indebido será permitido), genera falsos negativos.

### Actividad

* Activos: Capacidad de tomar acciones al momento de estar ocurriendo el ataque.
* Pasivos: Solo informa de un ataque que probablemente ya ocurrió.

Clasificación extraída de

(https://www.rediris.es/cert/doc/unixsec/node26.html).

## Heurística.

La heurística se define como un conjunto de técnicas y algoritmos que se aplican para solucionar un problema que presenta dificultad para hallar una solución óptima. Los algoritmos heurísticos no presentan mayor dificultad al momento de su aplicación y permiten optimizar soluciones con esfuerzos computacionales relativamente pequeños.

Estos algoritmos heurísticos se pueden clasificar en diferentes tipos.

### Algoritmos constructivos: consiste en elegir la opción óptima en cada paso local, para así intentar de lograr una solución óptima general, normalmente se aplica a problemas de optimización.

### Algoritmos de descomposición y división: consiste en descomponer el problema en subproblemas más simples, hasta llegar al punto en que la solución a estos subproblemas es trivial, luego se combinan las soluciones obtenidas hasta llegar a la solución del problema original.

### Algoritmos de reducción: consiste en identificar las soluciones buenas conocidas y suponer que la solución óptima también las tendrá, para así reducir el espacio de búsqueda de la solución.

### Algoritmos de manipulación del modelo: consiste en modificar el modelo del problema original y para así poder obtener una solución de un problema simplificado.

### Algoritmos de búsqueda usando vecindad: consiste en algoritmos que tratan de escapar de un máximo local para encontrar la solución a un problema, estos también son denominados algoritmos metaheurísticos.

En esta última categoría se encuentran los denominados Algoritmos Genéticos y Las redes neuronales entre otros, los cuales son un tipo de inteligencia artificial.

## Inteligencia Artificial.

La Inteligencia Artificial (IA) según John McCarthy (padre de la IA) es "la ciencia y la ingeniería de crear máquinas inteligentes, especialmente programas de computación inteligentes. Está relacionada con la tarea similar de utilizar ordenadores para comprender la inteligencia humana, pero la IA no se limita a métodos que sean observables biológicamente".

Extraído de

(https://elpais.com/diario/2011/10/27/necrologicas/1319666402\_850215.html)

Se entiende por esto que el objetivo principal de la IA consiste en crear un sistema capaz de imitar los procesos de inteligencia humana, estos procesos son el aprendizaje, el razonamiento y la autocorrección.

La IA se puede clasificar en dos grupos, la IA débil o estrecha y la IA fuerte. La débil son los sistemas que están diseñados y entrenados para una tarea en específico, la IA fuerte consiste en los sistemas diseñados para encontrar solución a una tarea desconocida.

Según Arend Hintze (profesor asistente de biología integradora e ingeniería y ciencias de computación en la Universidad Estatal de Michigan) existen 4 tipos de IA:

### Máquinas reactivas: Aquellos sistemas que toman las decisiones según las posibilidades que se les presentan al momento, es capaz de hacer predicciones pero no tiene memoria que lo ayude al momento de la toma de decisiones.

### Memoria limitada: Aquellos sistemas que utilizan experiencias pasadas para asistirse al momento de tomar decisiones, estas experiencias pasadas no se utilizan a largo plazo por lo tanto no son almacenadas de forma permanente.

### Teoría de la mente: Aquellos sistemas que son capaces de entender que los demás tienen creencias, deseos e intenciones las cuales juegan un papel fundamental para su toma de decisiones, es un tipo de IA que aún no existe.

### Autoconocimiento: Aquellos sistemas que tienen conciencia, son capaces de comprender su estado actual y utilizar información para influenciar en los pensamientos ajenos, es un tipo de IA que aún no existe.

En la IA existen muchos campos de estudio donde se especializa en distintas áreas, entre ellas tenemos las redes neuronales, algoritmos genéticos, lógica difusa, programación genética. A continuación, se profundizará sobre los algoritmos genéticos ya que será implementado en el desarrollo de este trabajo de investigación.

## Algoritmos genéticos.

Los algoritmos genéticos son métodos que se adaptan al entorno, utilizan modelos inspirados en los principios evolutivos de la naturaleza, la selección natural y el principio de supervivencia del más apto.

Los elementos básicos de los algoritmos genéticos son:

* Individuo: Representación de una posible solución.
* Población: Conjunto de individuos que serán evaluados y competirán entre ellos.
* Gen: representación de una característica del objeto real.
* Alelo: Los posibles valores de un gen.
* Cromosoma: Conjunto de genes que definen un individuo.
* Generación: Es el producto de un proceso evolutivo aplicado a una población anterior.
* Genotipo: Codificación de cómo está compuesto cada individuo.
* Fenotipo: El individuo resultante de operaciones propias del problema aplicadas al genotipo.
* Función objetivo: Artefacto con el cual se medirá el desempeño de cada individuo.
* Selección: Proceso mediante el cual se escogen los individuos más aptos de una generación.
* Cruce: Proceso mediante el cual se utiliza el genotipo de los individuos más aptos para hacer combinaciones y generar una generación más apta.
* Mutación: Modificaciones aleatorias que se le aplican a los genes de los individuos después del cruce para agregar diversidad a la generación.

Un algoritmo genético básico puede ser explicado de la siguiente forma:

1. Se genera aleatoriamente una población inicial la cual representa la primera generación.
2. Cada individuo de la generación se evalúa utilizando la función objetivo para determinar qué tan apto es.
3. Se seleccionan los individuos más aptos de la generación.
4. Se aplican los cruces y mutaciones necesarios para generar la siguiente generación.
5. Se itera hasta que la condición de parada se satisfaga.

Existen muchas combinaciones y diseños ingeniosos de los algoritmos genéticos los cuales son utilizados para resolver problemas de diferente naturaleza en múltiples áreas de la ciencia. En general se considera su implementación cuando los problemas tienen soluciones complicadas.

Los algoritmos genéticos son muy sencillos de implementar, su complejidad está en cómo se diseñan sus componentes y funciones. Ya que no sólo deben representar las entidades del mundo real, sino que se deben crear las condiciones ideales para que el algoritmo genético sea capaz de conseguir una buena solución que se ajuste a las necesidades del problema.

## Redes Neuronales.

Las Redes Neuronales son un modelo computacional que consiste en un grupo de unidades llamadas neuronas, las cuales están conectadas entre sí a través de unos enlaces, estos enlaces son utilizados por las neuronas para transmitir señales y así se capaces de comunicarse entre ellas.

Cada vez que un valor pasa de una neurona a otra a través de estos enlaces, es multiplicado por un valor de peso, estos pesos pueden variar por cada enlace, permitiendo así darles prioridad a unos enlaces sobre otros. Además de estos pesos puede existir una función de activación, esta modifica el resultado obtenido de una neurona o le impone un limite necesario para propagar el valor a la siguiente neurona.

Las Redes Neuronales son algoritmos capaces de aprender por si mismo, esto permite que sean utilizadas para detección de soluciones o características que sean más complejas utilizando programación convencional.

En las Redes Neuronales sobresalen las siguientes partes:

### Neurona: es un elemento que recibe múltiples valores de entrada, los suma e introduce el resultado en una función de activación que genera el resultado final, el cual es pasado a los siguientes enlaces si existen.

* Enlace: es la conexión entre las neuronas, aquí el valor de salida de la neurona anterior es multiplicado por el valor del peso perteneciente al enlace, para luego pasar el resultado de esta multiplicación a la siguiente neurona.
* Capas: representa cada nivel de neuronas que contiene la red neuronal, dividiéndose en 3 tipos:
  + Entrada: compuesta por las neuronas que reciben los valores de entrada de la Red Neuronal.
  + Escondida: compuesta por las neuronas pertenecientes a las capas entre la capa de entrada y la capa de salida, estas pueden variar en tamaño y cantidad dependiendo del problema a resolver y el tipo de Red Neuronal que se desea implementar.
  + Salida: compuesta por la o las neuronas encargadas de dar un resultado final, la cantidad de neuronas contenidas en esta capa dependerá del diseño de la Red Neuronal.
* Peso: es el valor por el cual son multiplicados los valores al pasar de una neurona a otra, a cada enlace se le asigna un peso, el cual puede incrementar o inhibir el estado de activación de las neuronas adyacentes.
* Bias:
* Función de activación: es una función que puede ser aplicada en cada neurona, con la cual se evalúa el valor de entrada y se define si la neurona debe ser activada o no.
* Tasa de aprendizaje:
* Función de perdida:

## Wireshark.

## Transmission Control Protocol.

## Internet Control Message Protocol.

## Three Way Handshake.

## Socket.

## IntelliJ IDEA.

## Encapsulamiento.

## Herencia.

## Polimorfismo.

## Patrones de diseño.

### Comando.

### Fábrica.

### Data Access Object: DAO.

### Bean.

## JUnit

## MySQL WorkBench.

## PL/SQL

## Stored Procedures

## Operaciones CRUD.

# Marco metodológico.

Como metodología para el desarrollo del Software se utilizará como base el Modelo Espiral, utilizando la mayoría de su estructura y recomendaciones. En este caso se utilizarán técnicas heurísticas, el cual es un paradigma en el que el conocimiento inicial del equipo de desarrollo es poco, por lo tanto, se debe utilizar una metodología que permita y facilite a los desarrolladores cometer errores y volver a porciones de código ya culminadas, para así mejorarlas o cambiarlas a medida que se adquiera más conocimiento y experiencia en este paradigma. Es de suma importancia la característica de esta metodología de ser iterativa e incremental, ya que, por la naturaleza del proyecto, una vez que se haya concluido una iteración, será necesario volver a la fase de diseño y hacer repetidas veces esfuerzos a nivel de investigación para acercarse cada vez más a los resultados aceptables. Esta metodología implica el desarrollo de prototipos, lo cual facilita al equipo evaluar el trabajo realizado y además calcular riesgos. De esta forma se podrá analizar la factibilidad de lograr los objetivos mediante el uso de las técnicas aplicadas y así determinar si se debe proseguir con otra técnica heurística.

# Desarrollo

A continuación, se explicará cómo se procedió para la realización de este proyecto. Se explicará paso a paso y en orden cronológico todas las actividades que se realizaron. Debido a que este proyecto está orientado a la investigación, cuenta con un equipo de trabajo y costo reducido, no hay cliente, entre otros aspectos importantes en los que se diferencia este a un proyecto comercial, se utilizó una adaptación de la metodología de espiral para la realización de cada uno de los retos presentados, además, no en todos los retos se pudo realizar todas las actividades planteadas por la metodología debido a sus dificultades propias.

## Iteración 1: Captación de paquetes.

### Determinación de objetivos.

Este módulo debe ser capaz obtener los paquetes que transitan dentro de una red, detectar las diferentes conexiones que existen en dicho tráfico, clasificarlas según su protocolo y acceder a sus atributos para facilitar su posterior análisis.

### Investigación.

Para tener persistencia y facilidad en la visualización de paquetes, se utilizó *Wireshark* para esnifar mientras se realizaron diferentes conexiones siguiendo diferentes criterios y se estudió detalladamente los paquetes que se generan cuando se completa cada conexión. El propósito de este experimento es detectar patrones en las secuencias paquetes que nos permitan identificar cómo una conexión se abre y se cierra limpiamente. Refiriéndose a limpiamente cuando no hay interrupciones o culminaciones repentinas de la conexión.

Los criterios utilizados para los experimentos que se realizaron para este levantamiento de información fueron los siguientes:

* **Protocolos:** Telnet, SSH, FTP, HTTP, HTTPS e ICMP
* **Sistema Operativo:** Windows y Linux
* **Esnifando en:** Cliente y Servidor

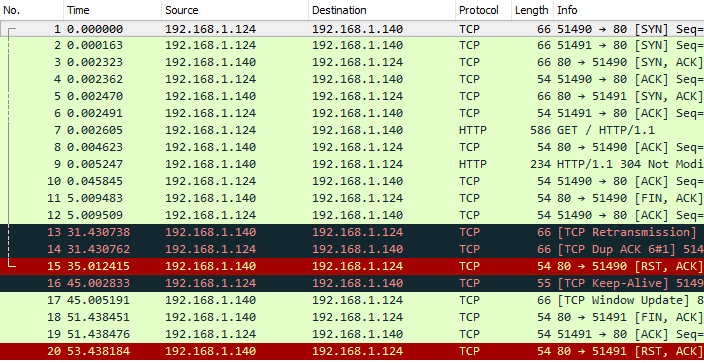
Nótese, la mayoría de los protocolos que se evaluaron son TCP ya que los paquetes UDP son menos comunes de conseguir, son mucho más variantes y más delimitados a tecnologías o servicios específicos por lo que se decidió descartarlos para este trabajo de investigación.

Ademas…………….EXPLICAICON DE POR QUE ESOS PROTOCOLOS Y POR QUE NO ELEGIR APPLE COMO SISTEMA OPERATIVO

Se realizaron experimentos utilizando todas las combinatorias de estos criterios, por ejemplo: En una red local se realizó una conexión HTTP entre dos máquinas locales, una con sistema operativo Linux específicamente Ubuntu la cual en este caso fue donde estaba instalado el servidor Web y otra con sistema operativo Windows la cual en este caso fue el cliente de la conexión. Haciendo la lectura en la maquina cliente.

El **.pcap** generado resultó de la siguiente manera:

Ilustración 1. Muestra de archivo .pcap



Fuente: Captura de pantalla de la lectura de WireShark.

Se pudo notar que para las conexiones TCP, siempre se cumple el *Three Way Handshake* lo cual marca claramente una pauta para poder determinar cómo una conexión TCP abre limpiamente. A diferencia del proceso de cerrar limpiamente una conexión TCP ya que se notó que cada protocolo tiene diferentes formas de cerrar.

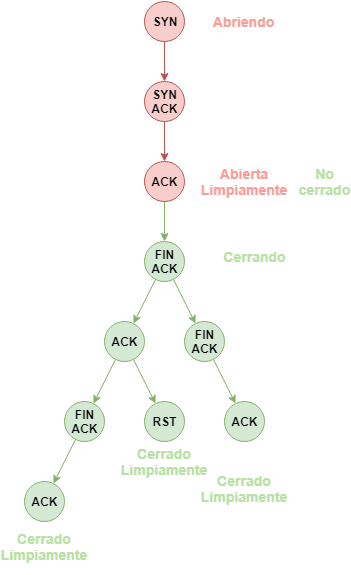
Por lo tanto, se ideó una estructura de árbol por cada protocolo la cual simboliza el ciclo de vida de cada conexión. Esencialmente se modeló el flujo de los paquetes tomando en cuenta sus banderas para que cada conexión siga con los siguientes estados:

1. **Abriendo:** Es el primer estado del ciclo de vida de una conexión. Para las conexiones TCP, esta simbolizado por un SYN y el resto de las banderas apagadas. En una conexión que no exista previamente, significa la creación de una nueva conexión.
2. **Abierto Limpiamente:** Finalización del *Three Way Handshake.*Significa que la conexión fue abierta sin errores y que ya la conexión esta lista para recibir su contenido.
3. **No cerrado:** Significa que en la conexión se está transmitiendo el contenido y que el proceso de cerrado no ha iniciado todavía.
4. **Cerrando:** Significa que la conexión comenzó a cerrarse siguiendo algún modo de cierre propio del protocolo al cual pertenece.
5. **Cerrando Limpiamente:** Significa que la conexión cumplió correctamente con uno de los flujos detectados y que no se esperan más paquetes de la misma.

Cabe estacar que para el análisis sólo se tomará en cuenta aquellas conexiones que fueron abiertas y cerradas limpiamente. Ya que son las que realmente están aportando al tráfico propio de una red.

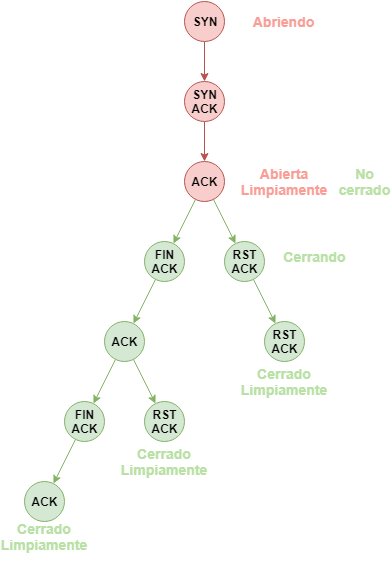
A continuación, las ilustraciones de cada uno de los árboles resultantes del análisis del flujo de las banderas de los paquetes por protocolo:

Ilustración 2. Árbol de Telnet.



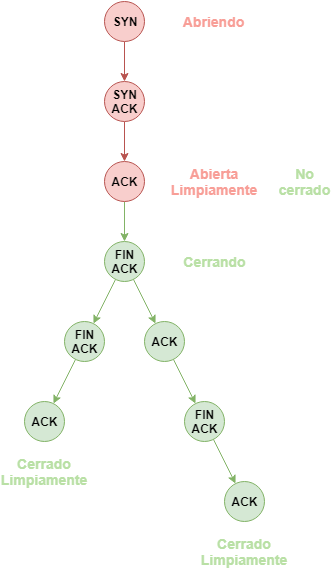
Fuente: Elaboración propia.

Ilustración 3. Árbol de SSH.



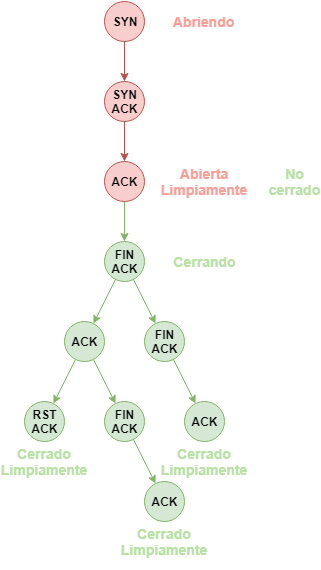
Fuente: Elaboración propia.

Ilustración 4. Árbol de FTP.



Fuente: Elaboración propia.

Ilustración 5. Árbol de HTTP.



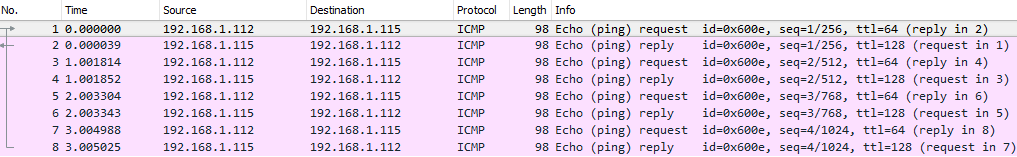
Fuente: Elaboración propia.

Debido a que el protocolo PING pertenece al protocolo ICMP el cual no tiene las mismas banderas que el protocolo TCP, no fue posible realizar un árbol que indique los estados de este tipo de conexión. Sin embargo, se notó que existen dos tipos de paquetes en estas conexiones:

* **Echo Request:** el cual significa la apertura de una conexión y
* **Echo Reply:** el cual significa cerrada de la conexión.

Dichos paquetes siempre vienen en pares, uno de cada uno. Además, los números de secuencia de cada par, indica cuántas peticiones de PING se realizaron en una conexión. A continuación, una foto de la visualización en *Wireshark* de un archivo .pcap de una conexión PING de 4 peticiones.

Ilustración 6. Muestra de paquetes PING.



Fuente: Captura de pantalla de la lectura de WireShark.

### Análisis de riesgo.

Debido a que se consiguió una forma metódica de clasificar las conexiones y determinar sus estados de apertura y cierre, este módulo tiene una alta probabilidad de ser desarrollado con éxito, siempre y cuando se consiga una librería que permita acceder a los atributos de los paquetes requeridos.

### Implementación.

Para desarrollar este módulo se utilizó una librería de Java llamada **Pcap4j**. La cual permite la lectura de un archivo .pcapy acceder a los paquetes que contiene y sus atributos.

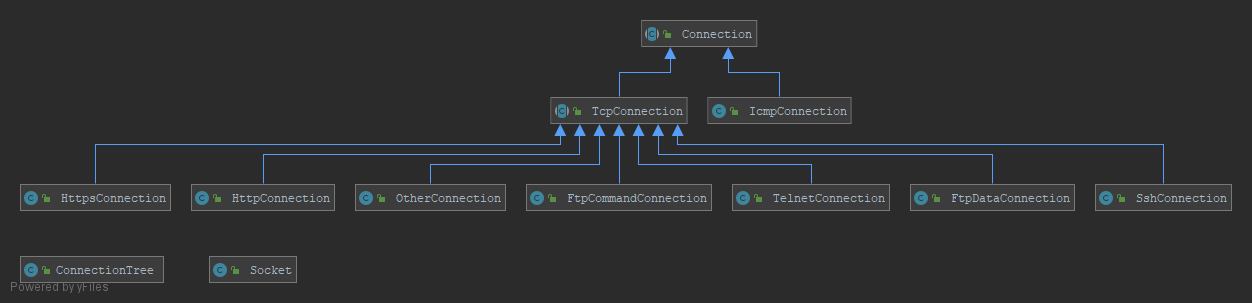
Esta lectura se realiza utilizando los métodos que provee la librería y se encapsuló utilizando el patrón de diseño comando. Se desarrollaron dos comandos llamados **ReadPcap** y **ReadMultiplePcaps** los cuales se en cargan de obtener una lista de paquetes a partir de la dirección donde se encuentra el archivo o los archivos .pcap respectivamente.

Esta lista de paquetes sirve como entrada para otro comando que se desarrolló llamado **Orchestrate**. Este se encarga de recorrer paquete por paquete fijándose en su Socket de origen y Socket destino y los distribuye de la siguiente forma. Comprueba si la combinación de Sockets del paquete que está siendo evaluado, existe en la lista de conexiones abiertas. Si existe, la agrega a la conexión a la que pertenece, de lo contrario, utilizando herencia y polimorfismo, se instancia una conexión nueva dependiendo de su tipo según el protocolo determinado. Esta instanciación se realiza siempre y cuando el paquete que se está evaluando este marcado sólo con la bandera SYN.

Cabe destacar que cada vez que se agrega un paquete a una conexión, se toman en cuenta las banderas que están presentes y se verifican contra el árbol propio de cada conexión según su protocolo para determinar el estado de apertura y cierre en el cual se encuentra la conexión en ese momento.

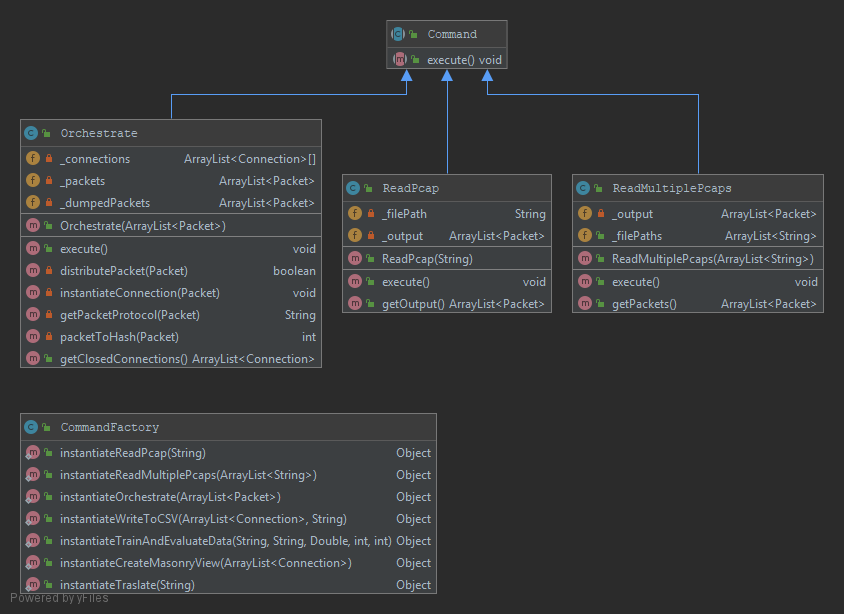
A continuación, el diseño de clases del módulo de captación de paquetes.

Ilustración 7. Diagrama de clase de entidades: Módulo de captación de paquetes.



Fuente: Exportado del código fuente usando IntelliJ IDEA.

Ilustración 8. Diagrama de clase de comandos: Módulo de captación de paquetes.



Fuente: Exportado del código fuente usando IntelliJ IDEA.

### Pruebas.

Las pruebas fueron realizadas utilizando la librería Junit y constan de ejecutar el comando orquestar con cada uno de los .pcap obtenidos durante el proceso de levantamiento de requerimientos y verificando que efectivamente se haya creado las conexiones pertinentes y que además cumplan con los ciclos de vida de estados de apertura y cerrado correctamente.

También, se descargó un archivo .pcap que se llamó “BigFlows.pcap” el cual contiene una buena cantidad de tráfico generado orgánicamente. El objetivo fue verificar que se hayan creado una cantidad coherente de conexiones y que los paquetes sobrantes no pertenezcan a ninguna conexión creada o simbolicen el inicio de una nueva conexión. Lo cual se pudo verificar correctamente.

### Conclusión.

Debido a que las pruebas fueron verificadas, podemos concluir que se desarrolló el módulo de captación de paquetes correctamente. Básicamente se está en presencia de un Software que tiene como entrada un archivo .pcap y como salida una lista de conexiones que se puede asegurar que fueron abiertas y cerradas limpiamente.

A pesar de que se cuenta con una capacidad limitada de protocolos que el módulo acepta, se puede asegurar que, gracias a un buen diseño, éste es escalable y puede aceptar con facilidad y sin necesidad de agregar mucha complejidad, otros protocolos. Siempre y cuando se haga un correcto levantamiento de requerimientos y se pueda construir un árbol que describa correctamente el comportamiento de dicho protocolo.

Por lo tanto, se puede concluir que el desarrollo de este módulo fue exitoso y los objetivos fueron cumplidos.

### Planificación.

Debido a que el desarrollo de este módulo fue exitoso, se procedió a avanzar con el siguiente objetivo.

## Iteración 2: Módulo de aprendizaje utilizando Programación Genética

### Determinación de objetivos

Diseñar un algoritmo genético que sea capaz de detectar tráfico propio partiendo de la data proporcionada por el administrador de la red luego de ser evaluada y filtrada.

### Investigación

Se investigó el funcionamiento básico de los algoritmos genéticos y el tipo de problemas que este tipo de algoritmos soluciona.

Además, se trató de conseguir ejemplos de código funcionales o librerías con los cuales se pueda tener una mejor idea de cómo funcionan. A pesar de que los ejemplos conseguidos solucionan problemas muy diferentes al actual, el código conseguido sirvió para comprender que los algoritmos genéticos realmente son bastante sencillos de programar. El reto verdadero está en saber modelar el problema en base a los datos reales, específicamente en los siguientes aspectos:

* Definir correctamente los cromosomas y genes.
* Debe existir suficiente diversidad en la población inicial.
* Buen diseño de la función objetivo, que realmente defina el desempeño deseado de cada individuo en cada generación.
* Selección de un algoritmo de cruce apropiado.
* Debe existir una probabilidad correcta de mutación.

Estos aspectos son los que se tomarán en cuenta para el diseño del módulo de aprendizaje.

Debido a la sencillez de la forma cómo programar los algoritmos genéticos, la utilización de librerías quedó descartada.

### Análisis de riesgo

Principalmente los algoritmos genéticos están orientados a resolver problemas relacionados con la optimización, por lo tanto, existe la posibilidad de que esta aproximación para resolver el problema planteado no sea viable, ya que se cuestiona la capacidad de un algoritmo genético de reconocer patrones.

### Implementación.

Inicialmente se desarrollaron varios algoritmos genéticos con el objetivo de resolver problemas más simples, para así poder entender un poco más sobre su funcionamiento. A medida que se avanzó con esta aproximación se fue presentando cada vez más palpable el riesgo de factibilidad antes mencionado, la capacidad de reconocimiento de patrones por parte del algoritmo genético para realizar la detección de anomalías, (insertar cita aquí para backear la idea de que los algoritmos genéticos no son buenos reconociendo patrones, aunque ahora no se).

### Pruebas

Debido a que no se prosiguió con el desarrollo de esta aproximación, no es coherente realizar pruebas.

### Conclusión

Se llegó a la conclusión de que los algoritmos genéticos no son capaces de reconocer patrones, lo cual es un elemento crucial para el desarrollo de esta tesis, ya que significa una gran importancia al momento de identificar anomalías, por lo tanto, esta aproximación fue descartada de la posible solución.

Además, por la naturaleza de los algoritmos genéticos y de este problema en específico, se hizo muy difícil conseguir un diseño que realmente se ajuste a las necesidades del problema y que sea planteado de una forma tal que los algoritmos genéticos sean capaces de resolverlo.

### Planificación

Debido al descarte de los algoritmos genéticos para la resolución del problema planteado, se procederá a investigar y evaluar las posibles alternativas para la solución, primeramente, la próxima heurística que será tomada en cuenta son Las Maquinas de Soporte Vectorial.

## Iteración 3: Módulo de aprendizaje utilizando Maquinas de Soporte Vectorial

### Determinación de objetivos

Diseñar un algoritmo utilizando Maquinas de Soporte Vectorial que sea capaz de detectar tráfico propio partiendo de la data proporcionada por el administrador de la red luego de ser evaluada y filtrada.

### Investigación

Luego de realizar la respectiva investigación necesaria para proceder desarrollo de esta iteración, se pudo confirmar que las Maquinas de Soporte Vectorial requieren de data supervisada para su entrenamiento, debido a que solo son capaces de clasificar cuando se han entrenado con data de este tipo.

¿Se investigo sobre las máquinas de soporte vectorial y nos dimos cuenta que no podrán ser utilizadas, aunque representan una excelente manera para la clasificación binaria utilizando diferentes planos en el espacio??, es un algoritmo de clasificación que necesita data supervisada para su entrenamiento (explicar data supervisada? Data supervisada es cuando en un conjunto de datos se puede representar o identificar la diferencia entre data positiva o negativa, o diferentes tipos de data, etiquetas) y debido a las condiciones necesarias para la lectura y captación de paquetes para registrar la data de entrenamiento solo se cuenta con data “positiva” o un solo tipo de data, por lo cual no se puede realizar el entrenamiento ya que seria data no supervisada (explicar data no supervisada?? Data no supervisada es cuando no se puede clasificar la data de entrenamiento en mas de 1 grupo, por lo tanto solo habría un tipo de data para el entrenamiento, positiva o negativa o un solo grupo) debido a que todos los paquetes captados por el modulo de captación de paquetes son paquetes sólo representan la data normal de la red.?

### Análisis de riesgo

No aplica.

### Desarrollo

No aplica.

### Pruebas

No aplica.

### Conclusión

Debido a la incapacidad de las Maquinas de Soporte Vectorial de ser entrenadas con data no supervisada, se descarta esta aproximación ya que no cumple con los requerimientos del sistema. (hablar mas a fondo del por que se utilizara data no supervisada y la importancia de esto)

### Planificación

Debido al descarte de esta aproximación utilizando Maquinas de Soporte Vectorial para la resolución del problema planteado, se procederá a investigar y evaluar sobre las Redes Neuronales.

## Iteración 4: Módulo de aprendizaje utilizando Redes Neuronales (¿normales?)

### Determinación de objetivos

Diseñar una Red Neuronal (normal) que sea capaz de detectar tráfico propio partiendo de la data proporcionada por el administrador de la red luego de ser evaluada y filtrada.

### Investigación

¿?

### Análisis de riesgo

¿tiene riesgo?

### Desarrollo

Se diseñó una red neuronal de x capas, teniendo 10 nodos en la primera capa, de los cuales 8 nodos representarían cada octeto de bits pertenecientes a un ip, los dos nodos restantes representan los puertos utilizados por el cliente y el servidor, con la excepción del protocolo ICMP, ¿ya que este no utiliza puertos, y estos son representados por el valor numérico “0”?.

Esta Red Neuronal contaba con x capas escondidas, una de tantos nodos y la otra de tantos, meter paja de activaciones y funciones de perdida y lo que se pueda.

La capa de salida estaba compuesta por un solo nodo, el cual es activado cuando el paquete evaluado pertenece al trafico normal de la red usando tal activación (binario).

Al momento de cargar la data se presentó el mismo problema encontrado con las maquinas de soporte vectorial, es necesario entrenar la red neuronal normal con data supervisada, lo cual imposibilitó la continuación del desarrollo de esta aproximación.

### Pruebas

No aplica.

### Conclusión

Debido a la necesidad de las redes neuronales normales de ser entrenadas con data supervisada se descarta esta aproximación, ya la data que será utilizada para el entrenamiento es del tipo no supervisada.

### Planificación

En este caso se decidió mantener la idea de las Redes Neuronales como posible solución para el problema presentado, por lo tanto, se procedió a trabajar con las redes neuronales de tipo variatonal autoencoder.

## Iteración 5: Módulo de aprendizaje utilizando Redes Neuronales de tipo Variational Autoencoders

### Determinación de objetivos

Diseñar una Red Neuronal (de tipo Variational Autoencoder) que sea capaz de detectar tráfico propio partiendo de la data proporcionada por el administrador de la red luego de ser evaluada y filtrada.

### Investigación

Luego de haber determinado la imposibilidad de las redes neuronales (normales) de realizar de manera correcta el entrenamiento utilizando data no supervisada, se investigó sobre las redes neuronales de tipo variational autoencoder, este tipo de redes neuronales hacen blablá (investigar labia)

### Análisis de riesgo

Posibilidad de obtener los valores reales de los paquetes reconstruidos luego de ser evaluados por la red neuronal.

### Desarrollo

Este módulo fue desarrollado en el lenguaje de programación Java, utilizando la librería DeepLearning4J, (buscar en el repo la versión que tiene los codigos de deeplearning4j de variational autoencoder, para agregar anexos de ser necesario y poder explicar el desarrollo del modulo)

### Pruebas

(buscar en el repo la versión con este código, agregar anexos)

### Conclusión

Debido a la limitación al momento de obtener la reconstrucción de los paquetes evaluados, resulta imposible el cálculo del error obtenido por cada paquete, para asi poder determinar si el mismo es propio de la red o representa una anomalía, por lo tanto, esta aproximación para resolver el problema planteado fue descartado.

### Planificación

Se decidió proceder con las redes neuronales de tipo autoencoder, ya que este tipo de red neuronal permite obtener la reconstrucción de cada paquete evaluado. (muy corto?)

## Iteración 6: Módulo de aprendizaje utilizando Redes Neuronales de tipo Autoencoders

### Determinación de objetivos

Diseñar una Red Neuronal (de tipo Autoencoder) que sea capaz de detectar tráfico propio partiendo de la data proporcionada por el administrador de la red luego de ser evaluada y filtrada.

### Investigación

Debido a que la data con la cual será entrenada la red neuronal es de tipo no supervisada, y el descarte de las redes neuronales de tipo variational autoencoder, se decide utilizar una aproximación parecida a la iteración 5, pero utilizando una red neuronal de tipo autoencoder.

Este tipo de red se entrena con el objetivo de que sea capaz de aprender a reconstruir data de cierto tipo, en este caso de paquetes pertenecientes a la red, se utiliza esta aproximación ya que la data no es supervisada, por lo tanto, la red neuronal no tiene la posibilidad de ser entrenada utilizando “ejemplos” de anomalías y de paquetes normales.

La librería previamente utilizada (DeepLearning4J) no ofrece este tipo de solución, por lo tanto se decidió realizar este módulo en el lenguaje de programación Python, utilizando la librería llamada Tensorflow, la cual si permite el desarrollo de redes neuronales de tipo autoencoder.

### Análisis de riesgo

Tiempo de desarrollo debido de a la poca experiencia con el lenguaje de programación Python. ¿?

### Desarrollo

### Pruebas

### Conclusión

### Planificación

## Iteración 7: Base de datos positiva.

### Determinación de objetivos.

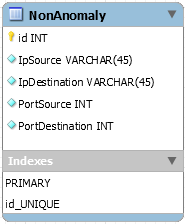
Desarrollar una base de datos que almacene la información obtenida del módulo de aprendizaje. Debido a que dicho módulo se encarga de filtrar las conexiones propias de un archivo .pcap, se puede decir que esta base de datos sólo almacenará el tráfico positivo de la red de la cual se obtuvo el .pcap.

### Investigación.

Como sabemos, Snort tiene una cantidad limitada de atributos mediante los cuales se permite la escritura y configuración de reglas. Por esta razón el diseño de esta base de datos es bastante sencillo, ya que no existe razón para almacenar una variedad de datos con los que no se podrá extraer información útil que aporte al objetivo principal que es filtrar paquetes usando Snort.

Luego, el diseño de esta base de datos es el siguiente:

Ilustración 9. Diagrama Entidad-Relación de la base de datos positiva.



Fuente: Exportado de MySQL WorkBench.

Se aplicó una regla en la que se indica que la combinación de IpOrigen, IpDestino, PuertoOrigen y PuertoDestino debe ser única con el motivo de que nunca se repita una combinación de los valores de estos atributos para que no exista ninguna regla duplicada.

### Análisis de riesgo.

Debido a la sencillez de los datos que maneja el módulo de aprendizaje, por razones ya explicadas, no existe riesgo alguno para el desarrollo correcto de esta iteración.

Además, el equipo de trabajo tiene experiencia desarrollando este tipo de funcionalidades e incluso se tiene un marco de trabajo estandarizado para acceder bases de datos MySQL utilizando el lenguaje de programación Java.

### Implementación.

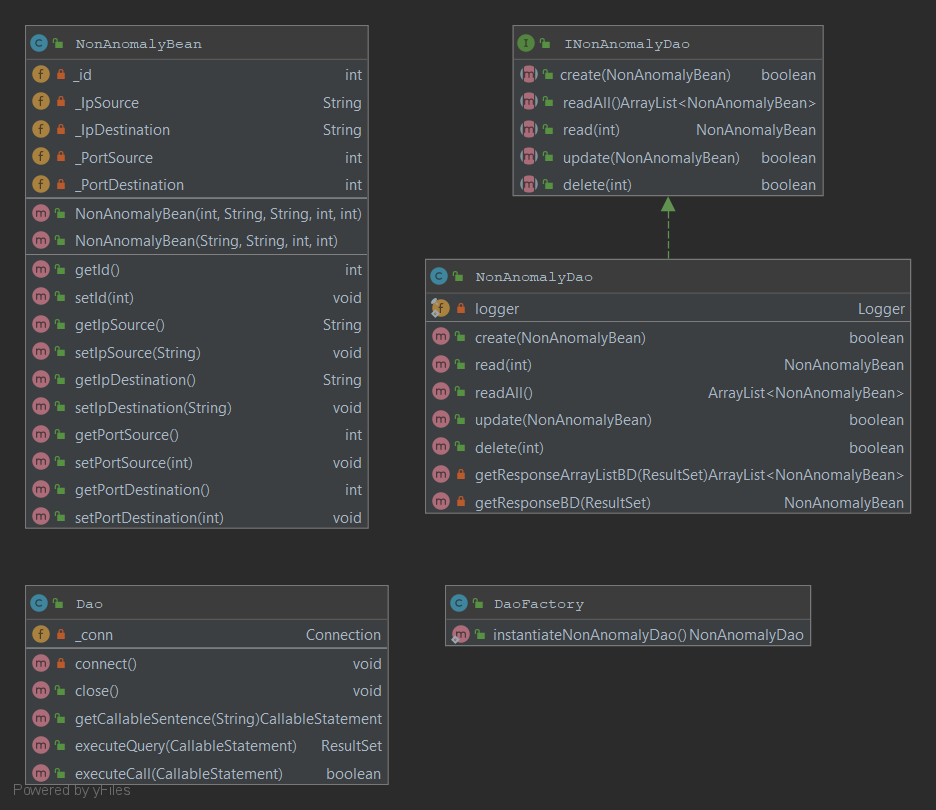
Se utilizó la herramienta MySQL WorkBench para la creación de la tabla **NonAnomaly** del inglés: No anomalía, el cual se refiere a tráfico propio.

Además, se utilizó el lenguaje PL/SQL para crear un conjunto de StoredProcedures encapsulan las funciones básicas que se pueden realizar con una entidad, los CRUD.

Se utilizó el patrón de diseño DAO para manipular la base de datos de una forma estandarizada.

A continuación, un diagrama de clase que ilustra la manipulación de dicha base de datos.

Ilustración 10. Diagrama de clases DAO.



Fuente: Exportado del código fuente usando IntelliJ IDEA.

### Pruebas.

Se utilizo la librería JUnit para ejecutar todas las operaciones que se implementaron para la manipulación de esta base de datos y se usaron reportes para probar que las operaciones realizadas impactaran de la forma deseada su persistencia.

### Conclusión.

Se desarrolló una base de datos que permite a la cual se tiene completo acceso y capacidad de alterar sus datos. No se descuidó la utilización formas estandarizadas y patrones de diseño para facilitar la escalabilidad y reusabilidad de futuras versiones de este proyecto.

### Planificación.

Debido a que el desarrollo de este base de datos fue exitoso, se procedió a avanzar con el siguiente objetivo.

## Iteración 6: Módulo de retroalimentación.

### Determinación de objetivos.

Este módulo es de suma importancia ya que le permitirá al administrador darle continuidad al análisis que haya realizado anteriormente. El administrador deberá ser capaz no sólo de modificar las reglas que serán generadas si no también redefinir las condiciones iniciales del módulo de aprendizaje para así volver a realizar el análisis. Todo esto con la finalidad de tener un conjunto de reglas que se acerquen a definir el trafico propio de la red a analizar de la forma mas acertada posible.

### Investigación.

### Análisis de riesgo.

El mayor riesgo que compromete el éxito de este módulo es la usabilidad. Una de los objetivos implícitos de este proyecto es facilitarle el trabajo al administrador de la red. Se debe diseñar las interfaces gráficas de tal forma que se pueda obtener información importante cómodamente y se puedan realizar acciones relevantes usando dicha información de una manera cómoda y efectiva.

### Implementación.

### Pruebas.

Las pruebas realizadas en este módulo

### Conclusión.

### Planificación.

Debido a que el desarrollo de esta base de datos fue exitoso, se procedió a avanzar con el siguiente objetivo.

## Iteración 7: Módulo de traducción.

### Determinación de objetivos.

Este módulo debe ser capaz de extraer toda la información de la base de datos positiva y traducirla a reglas con la sintaxis correcta que usa Snort.

### Investigación.

Se investigó la sintaxis que usa Snort para definir sus reglas, así como los modos de configuración de reglas que se pueden utilizar. En este caso se usará el modo de *whitelist*. El cual quiere decir que sólo se aceptarán los paquetes que estén listados en dicho archivo.

### Análisis de riesgo.

Debido a que anteriormente se pensó por adelantado y se investigó cuales son los atributos que Snort permite para la definición de sus reglas y se diseño los módulos de aprendizaje y la base de datos para que exista correlación entre los atributos que se analizaron y los que permite Snort, no existe riesgo alguno para el correcto desarrollo de este módulo.

### Implementación.

Se creó un comando llamado **Translate** el cual se encarga de hacer una petición de todos los registros de la tabla **NonAnomaly** utilizando el DAO, luego crea una regla por cada registro obtenido utilizando la siguiente sintaxis:

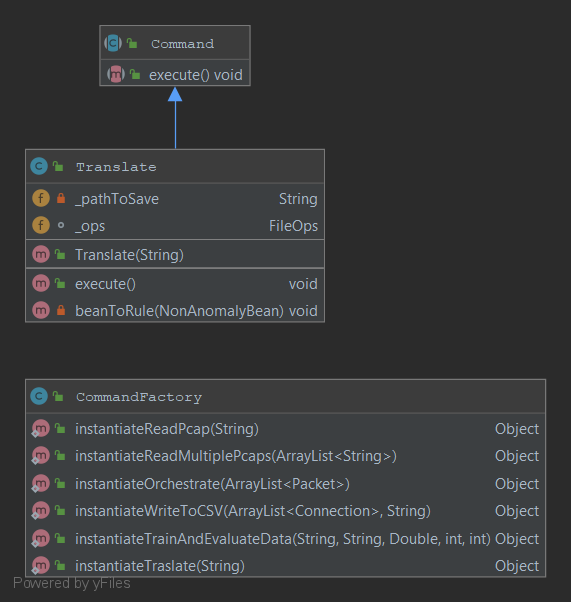
**alert tcp !IPOrigen PuertoOrigen -> IPDestino PuertoDestino (msg: “Anomalia Detectada!”)**

Por ejemplo: alert tcp !192.168.1.115 23 -> 192.168.1.107 50841 (msg:" Anomalia Detectada!")

Cada regla será escrita una por una en un archivo de texto llamado *whitelis.rules.* Este archivo es la salida final del programa ya que es el archivo de configuración para Snort.

A continuación, un diagrama de clases que ilustra el diseño de este módulo.

Ilustración 11. Diagrama de clase de módulo de traducción.



Fuente: Exportado del código fuente usando IntelliJ IDEA.

### Pruebas.

Se probó que toda la base de datos se escribiera en el archivo de configuración contando la cantidad de reglas y verificando que se hayan creado la misma cantidad de reglas que registros en la base de datos. También se verificó que la sintaxis sea correcta y que no se haya agregado información extra e innecesaria al archivo que pueda comprometer la validez del mismo.

### Conclusión.

Ya que las pruebas se verificaron correctamente, se puede decir que el desarrollo de este módulo fue exitoso.

El comportamiento deseado es que Snort levante una alerta cada vez que se encuentre con un paquete que **NO** este en la lista de las reglas.

Cabe destacar que inicialmente este archivo estará vacío y que para que una regla se pueda agregar correctamente debe suceder al menos una vez la conexión pertinente.

### Planificación

Debido a que el desarrollo de este módulo fue exitoso, se procedió a avanzar con el siguiente objetivo.

## Iteración 8: Integración.

### Determinación de objetivos.

### Investigación.

### Análisis de riesgo.

### Implementación.

### Pruebas.

### Conclusión.

### Planificación.

## Iteración 9: Instalación en ambiente real y pruebas.

Instalar el sistema en un ambiente real para realizar pruebas y comprobar que el funcionamiento del programa es el esperado. Para esto se observarán los siguientes tres aspectos.

* Facilidad para la creación de reglas.
* Cantidad de falsos positivos.
* Cantidad de falsos negativos.

### Investigación.

Se buscará una red en la que se genere una cantidad considerable de tráfico y que se pueda asegurar que la mayoría de dicho tráfico esta conformado por conexiones que pertenezcan a los protocolos estudiados. Además, se quiere que la red sea lo suficientemente estable para que no existan cambios significativos desde el momento de la captación de paquetes hasta el momento del establecimiento de las reglas.

### Análisis de riesgo.

Existen numerosos riesgos en esta iteración debido a que el software desarrollado hasta el momento, solo ha sido probado con una cantidad de data relativamente pequeña. Debido a esto, uno de los principales problemas que se puede presentar es que, debido a que la captación de paquetes será por un tiempo más prolongado y la el tráfico será un poco más complejo y numeroso, la data que se obtendrá será bastante mas grande. Por esto, los tiempos de respuesta no sólo del módulo de aprendizaje si no también de la captación de paquetes serán bastante largos. Además, existe una posibilidad de que los parámetros de la red neural definida con la data de prueba, no sean los mejores para esta otra data generada. Si esto ocurre, se tendrá que probar repetidas veces el proceso de aprendizaje para de esta forma ajustar dichos parámetros de la mejor manera posible.

También existen riesgos al momento de obtener la data, ya que, como se ha mencionado anteriormente, durante el proceso de captación de tráfico, no debe haber ningún ataque al sistema evaluado. Si esto llegase a ocurrir, el módulo de aprendizaje tomará en cuenta esta data corrupta como tráfico propio y se construirán las reglas incluyendo como tráfico permitido un ataque, por lo que se estaría en presencia de un falso negativo.

### Implementación.

Se instaló Wireshark en una máquina que tiene acceso a toda la red de XSITIO y se puso a captar paquetes en modo promiscuo durante un periodo de XTIEMPO.

Se obtuvo ese archivo .pcap que pesa XBytes y tiene XCantidad de paquetes. Este archivo será nuestra data de aprendizaje.

Luego, se puso a captar paquetes en modo promiscuo durante un periodo de XTIEMPO.

Se obtuvo ese archivo .pcap que pesa XBytes y tiene XCantidad de paquetes. Este archivo será nuestra data de evaluación.

Se procedió a agregar ambos archivos al sistema. Se aplicaron los filtros deseados y se ejecutó el módulo de aprendizaje. Luego de XTIEMPO,

### Pruebas.

### Conclusión.

### Planificación.

Cambio aquí

asdasd

rtuyrtuyuuuuuu

Rtuyrtuyrtuyrtuy

rtuyrtuyrtuyrtuyrtuy

# Resultados

# Conclusiones

# Recomendaciones

Posibles recomendaciones:

* Para cuando se están captando los paquetes y armando las conexiones, podemos fijarnos también en el SYN y ACK de los paquetes para evitar ataques de ¿Suplantación de identidad?
* Utilizar un algoritmo de programación genética para crear la red neuronal con mejor score.
* Utilizar directamente la red neuronal como el IDS ya que en este caso Snort nos limita la cantidad de atributos a utilizar para detectar las anomalías, pero mencionando (confirmar esto) que puede ser más lento.
* Se puede cambiar las reglas que no sean sólo alertar y también se puede customizar el mensaje de la anomalia detectada en la regla.

# Referencias Bibliográficas

IDS-Tipos de IDS. *Administración de sistemas operativos* [en línea]. Disponible en:

<http://www.adminso.es/index.php/IDS-Tipos_de_IDS>

Host-based intrusion detection system. *Wikipedia* [en línea]. Octubre 2017. Disponible en:

<https://es.wikipedia.org/wiki/Host-based_intrusion_detection_system>

Sistema de detección de intrusos. *Wikipedia* [en línea]. Enero 2018. Disponible en: https://es.wikipedia.org/wiki/Sistema\_de\_detección\_de\_intrusos

Sistemas de detección de intrusos. *RedIris* [en línea]. Julio 2002. Disponible en: https://www.rediris.es/cert/doc/unixsec/node26.html

Programación genética. *Wikipedia* [en línea]. Noviembre 2017. Disponible en:

https://es.wikipedia.org/wiki/Programaci%C3%B3n\_gen%C3%A9tica

ANGUIANO, Eloy. Programación genética. *PCWorld* [en línea]. Octubre 1997. Disponible en:

http://www.pcworld.es/archive/programacion-genetica

ARRANZ, Jorge, PARRA, Antonio. ALgoritmos Genéticos. [en linea]. Disponible en: http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf

ROUSE, Margaret. Inteligencia artificial, o IA. *TechTarget* [en linea]. Abril 2017. Disponible en:

http://searchdatacenter.techtarget.com/es/definicion/Inteligencia-artificial-o-AI

ORTIZ, Carlos. Algoritmos heurísticos y metaheurísticos para el problema de localización de regeneradores. [en linea]. 2010. Disponible en:

https://eciencia.urjc.es/bitstream/handle/10115/4129/memoriaPFC%20Carlos%20Rodr%C3%ADguez.pdf?sequence=1&isAllowed=y

Técnicas heurísticas aplicadas al problema del cartero viajante (TSP). [en linea]. Mayo 2004. Disponible en:

<http://revistas.utp.edu.co/index.php/revistaciencia/article/viewFile/7279/4311>

Red neuronal Artificial. *Wikipedia* [en línea]. Julio 2019. Disponible en:

<https://es.wikipedia.org/wiki/Red_neuronal_artificial>

Julián Guillermo. Las redes neuronales: qué son y por qué están volviendo. *Xataka* [en línea]. Diciembre 2014. Disponible en:

<https://www.xataka.com/robotica-e-ia/las-redes-neuronales-que-son-y-por-que-estan-volviendo>

Calvo, Diego. Perceptrón – Red Neuronal. [en línea]. Diciembre 2018. Disponible en: <http://www.diegocalvo.es/perceptron/>

Función de activación. *Wikipedia*. [en línea]. Abril 2019. Disponible en:

<https://es.wikipedia.org/wiki/Función_de_activación>