Facultad 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](#_Toc11420289)

[Agradecimiento 2](#_Toc11420290)

[Dedicatoria 2](#_Toc11420291)

[Índice 3](#_Toc11420292)

[Índice de Tablas y Gráficas 4](#_Toc11420293)

[Sinopsis 5](#_Toc11420294)

[Introducción 6](#_Toc11420295)

[Planteamiento del Problema 7](#_Toc11420296)

[Objetivo General 10](#_Toc11420297)

[Objetivos Específicos 10](#_Toc11420298)

[Alcance y Limitaciones 11](#_Toc11420299)

[Justificación 15](#_Toc11420300)

[Marco Referencial 16](#_Toc11420301)

[Marco metodológico. 23](#_Toc11420302)

[Desarrollo 25](#_Toc11420303)

[1. Captación de paquetes: 25](#_Toc11420304)

[Resultados 32](#_Toc11420305)

[Conclusiones 32](#_Toc11420306)

[Recomendaciones 32](#_Toc11420307)

[Referencias Bibliográficas 33](#_Toc11420308)

# Índice de Tablas y Gráficas

[**Ilustración 1 - Muestra de archivo .pcap** 27](#_Toc11419705)

# 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.

# Objetivo General

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

# Objetivos Específicos

1. Diseñar un algoritmo de aprendizaje basado en técnicas heurísticas capaz de reconocer el tráfico propio de una red.
2. Diseñar e implantar una base de datos positiva que represente el comportamiento propio de la red.
3. Desarrollar el módulo de aprendizaje.
4. Desarrollar el módulo de retroalimentación.
5. Desarrollar el módulo de traducción.
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.

# Alcance y Limitaciones

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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 estén contenidos 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

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.

Éste 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.

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:

1. 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.

1. Estrategia

* Detección de anomalías: Solo conoce el comportamiento “normal” de la red o sistema (todo lo que no se identifique como comportamiento normal sera 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.

1. 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).

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, los cuales son un tipo de 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".

Extraido 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.

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.

# 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

¿Intro de Metodología?

A continuación, se explicará cómo se procedió para la realización de este proyecto. Explicaremos paso a paso y en orden cronológico todas las actividades que se realizaron. Cabe destacar que se utilizó una adaptación (orientada a la investigación, equipo de trabajo reducido, costo reducido, no hay cliente, etc.) de la metodología de espiral para la realización de cada uno de los retos presentados. No en todos los pasos o retos se pudo realizar todos los pasos y los tiempos fueron variables, bla bla.

## Captación de paquetes:

El primer reto que se presento fue realizar un módulo que sea capaz de capturar los paquetes que transitan dentro de una red y poder acceder a sus atributos para su posterior análisis. Para lograrlo se utilizó una librería de Java llamada **Pcap4j**. La cual permite la lectura de un archivo **.pcap,** los cuales son una de las salidas que se obtienen al esnifar una red utilizando alguno de los programas que utilizan el API de captura de tráfico de red **PCAP** como por ejemplo **Wireshark**.

Se realizaron diferentes conexiones siguiendo diferentes criterios y se estudió detalladamente los paquetes que se generan cuando se completa cada conexión. Con el propósito de detectar patrones en los 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

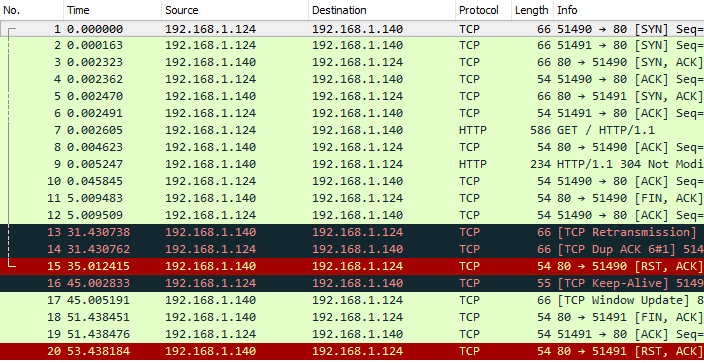
Nótese, la mayoría de los protocolos que se evaluaron son TCP ya que los paquetes UDP son menos comunes de conseguir, son mucho más variantes y más delimitados a tecnologías o servicios específicos por lo que se decidió descartarlos para este trabajo de investigación.

Se realizaron experimentos utilizando todas las combinatorias de estos criterios, por ejemplo: En una red local se realizó una conexión HTTP entre dos maquinas 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: Elaboración propia.

Se pudo notar que para las conexiones TCP, siempre se cumple el **Three Way Handshake** (Poner definición con cita) 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.

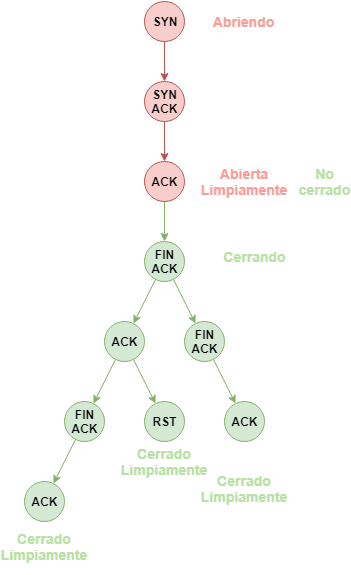
Por lo tanto, se ideó una estructura de árbol por cada protocolo la cual simboliza el ciclo de vida de cada conexión. Esencialmente se modeló el flujo de los paquetes tomando en cuenta sus banderas para que cada conexión siga con los siguientes estados:

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, de 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 el contenido de la conexión.
3. **No cerrado:** Significa que en la conexión se esta transmitiendo el contenido y que el proceso de cerrado no ha iniciado todavía.
4. **Cerrando:** Significa que la conexión comenzó a cerrarse bajo 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 esperarán mas paquetes de la misma.

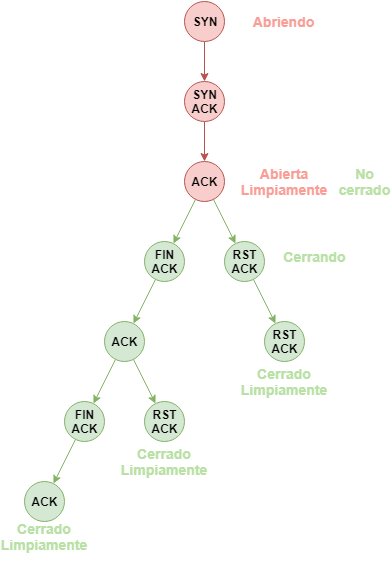
Cabe estacar que para el análisis sólo se tomará en cuenta aquellas conexiones que fueron abiertas y cerradas limpiamente. Ya que son las que realmente están aportando al tráfico propio de una red.

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

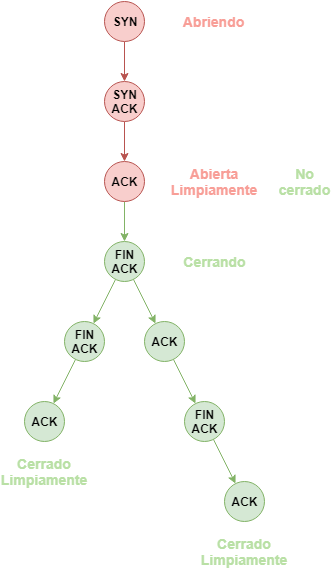
Ilustración 2. Árbol de telnet



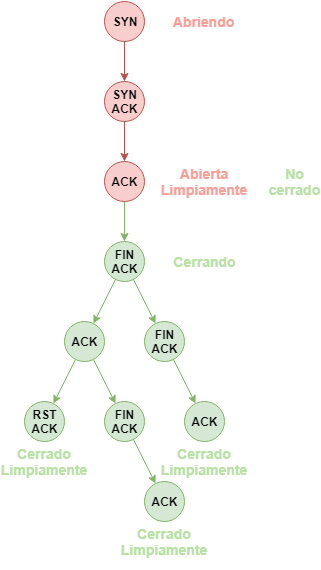
SSH:



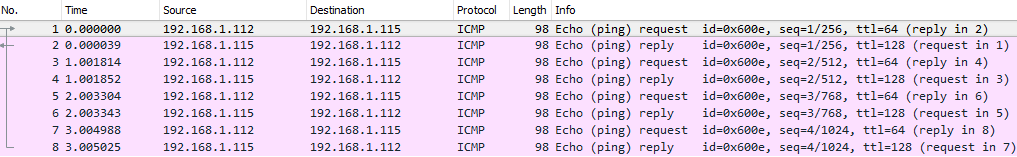
FTP:



HTTP:



PING: Debido a que este protocolo 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 el archivo **.pcap** de una conexión PING.



Luego de este levantamiento de requerimientos, se procedió a desarrollar el módulo de captación de paquetes el cual obtiene un archivo **.pcap** como entrada y lo recorre paquete por paquete fijándose en el Socket de origen y Socket destino.

# 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.

# 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