HONEYPOT INTELIGENTE DE DIFICULTAD INCREMENTAL

Modalidad: Exploratorio

JUAN ESTEBAN SIERRA LARA

OSCAR URIBE LONDOÑO

Trabajo de grado para optar al título de

Ingeniero de Sistemas y Computación

Juan Esteban Velásquez Múnera

Ingeniero Informático

UNIVERSIDAD EIA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
ENVIGADO
2019

Este trabajo es dedicado a nuestras familias, gracias por haber estado ahí y apoyarnos en nuestro camino hacia convertirnos ingenieros.

AGRADECIMIENTOS

Queremos agradecer a la doctora Isis Bonet por el apoyo que nos brindó su apoyo, sin ella este trabajo no hubiera sido posible.

Gracias a Juan Esteban por el acompañamiento y los consejos para encontrar el camino en el área que nos apasiona, la seguridad.

Gracias a Valentina Gallego por las revisiones y haber estado ahí incondicionalmente.

Finalmente, gracias a Johan Vélez por ser el mejor director de carrera que podríamos tener y por habernos apoyado y levantado el ánimo en los momentos difíciles.

CONTENIDO

pág.

INTRODUCCIÓN 9

1. PRELIMINARES 10

1.1 Planteamiento del problema 10

1.2 Justificación 11

1.3 Objetivos del proyecto 11

1.3.1 Objetivo General 11

1.3.2 Objetivos Específicos 11

1.4 Marco de referencia 12

1.4.1 Antecedentes 12

1.4.2 Marco teórico 13

2. METODOLOGÍA 18

3. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS 19

3.1 Simulación del ambiente para la recolección de datos 19

3.2 Automatización de la recolección de los datos 20

3.3 Análisis de los datos mediante algoritmos de inteligencia computacional 21

3.4 Creación de un servicio de clasificación 25

4. CONCLUSIONES Y CONSIDERACIONES FINALES 27

REFERENCIAS 28

ANEXO 1 ..... **¡ERROR! MARCADOR NO DEFINIDO.**

LISTA DE FIGURAS

pág.

Figura 1 Máquina virtual bWAPP *20*
Figura 2: Diagrama de clases de los scripts para el parsing *21*
Figura 3 Nube de palabras *22*
Figura 4 Ngrama con las palabras encontradas en los logs *23*
Figura 5 Entrenamiento de los modelos de clasificación en Orange *24*
Figura 6: Resultados de la predicción de los datos multiclase *25*
Figura 7 Resultados del servicio de clasificación *26*

GLOSARIO

MAQUINA VIRTUAL: Una máquina virtual es un archivo de PC, que suele denominarse “imagen”, que se comporta igual que un equipo real, es decir, es capaz de crear un equipo dentro de un equipo

HOST: Un Host es un computador que es accesible dentro de una red.

NODO: Un Nodo dentro de una red es un host que se comunica con otros hosts permitiendo así una interconexión.

SHELL: Un Shell es un programa de computadores con una interfaz que permite ejecutar comandos en un sistema.

UNIX: Unix es un sistema operativo desarrollado en 1960 y en el cual están basados muchos de los sistemas operativos modernos.

RSYSLOG: Rsyslog en un programa *open source* para la transferencia de logs a través de una red IP para sistemas UNIX.

SERVICIO: Un servicio es un medio por el cual dos computadores se pueden comunicar y realizar peticiones usando un protocolo preestablecido.

NUBE DE PALABRAS: Es una representación visual que permite identificar las etiquetas dentro de un texto, a la vez de la frecuencia con la que se repiten, la cual es detarminada por el tamaño de las etiquetas.

NGRAMAS: Se llama n-grama a una subsecuencia de de n elementos consecutivos en una secuencia dada.

BOLSA DE PALABRAS: Una bolsa de palabras es un método usado en procesamiento de lenguaje que permite tener la representación de un texto en función de las palabras que tiene.

PETICIÓN HTTP: Son un conjunto de métodos que indican la acción que se quiere realizar en un servidor Web.

RESUMEN

En esta época de globalización e hiperconectividad, se ha vuelto común escuchar el término ciberataque o hacker, dado a que hoy en día toda la información de las compañías está en la red. Los ataques cibernéticos se han vuelto un tema común con el que deben lidiar las compañías, quienes tienen que dedicar grandes inversiones para prevenir o mitigar estos ataques.

En este trabajo, se plantea la posibilidad eliminar la presencia continua de personal calificado para la identificación de los ataques mediante herramientas de inteligencia computacional que permitan el procesamiento y análisis de los datos obtenidos para poder mitigar los ataques a las infraestructuras reales gracias a la automatización de acciones correctivas pertinentes.

Para esto, se creó un ambiente controlado, el cual se sometió a múltiples ataques mientras se realizaba la recolección de la información de estos mismos, permitiendo un posterior análisis para el entrenamiento de una inteligencia artificial, la cual fue creada a partir de múltiples algoritmos, los cuales validan una decisión para catalogar una nueva entrada como algún tipo de ataque o no.

Finalmente, se expuso esta inteligencia a través de un servicio, el cual da una respuesta a partir de la información que se le envío, analizando esta para determinar si es un ataque o no.

Palabras claves: *Honeypots,* Inteligencia artificial, seguridad informática, ciberataques, hacking

ABSTRACT

In this time of globalization and hyperconnectivity, it has become common to hear the term cyberattack or hacker, given that today all the information of the companies is on the net. The cyberattacks have become a common subject that the companies must deal with, and they have to make big investments to prevent or mitigate those attacks.

In the following report, the possibility of eliminating the continuous presence of qualified personnel is raised though tools of artificial intelligence that allow the processing and analytics of the data obtained to mitigate the attacks to the real infrastructure thanks to the automatization of the relevant corrective actions.

For this, a controlled environment was created, which underwent multiple attacks while collecting their information, allowing its posterior analysis for the artificial intelligence training, which was created using multiples algorithms, which validate a decision to catalog a new entry as an attack or not.

Finally, this artificial intelligence was exposed through a service, which gives an answer through the information that was sent, analyzing it to determine whether it is an attack or not.

Keywords:

Honeypot, Artificial Intelligence, information security, cyber-attacks, hacking

INTRODUCCIÓN

En este trabajo podrá encontrar una forma de realizar la clasificación de los *logs* que generan los servidores, para determinar cuando cierta entrada se puede considerar un ataque o no y tomar las acciones necesarias para evitar o mitigar un posible ataque.

Inicialmente, podrá encontrar como se realizó la recolección de información, simulando diferentes ambientes y determinando cual fue más apropiado para la toma de datos. Luego podrá encontrar como se realizó un análisis de los datos, luego de haber realizado un preprocesamiento, para determinar que palabras o que grupos de palabras eran importantes.

Podrá encontrar como se determinó cuales algoritmos de inteligencia artificial eran más indicados para el problema actual con los resultados de entrenamiento de cada uno de estos.

Finalmente podrá encontrar el microservicio realizado para la clasificación, el cual utiliza todos los algoritmos que fueron evaluados para determinar si una nueva entrada es un posible ataque o no, devolviendo la respuesta correspondiente.

1. PRELIMINARES

1.1 PLANTEAMIENTO DEL PROBLEMA

En un mundo globalizado y totalmente conectado, las amenazas son cada día más variadas y latentes, por esto, hoy en día se habla de ciberataques y ciberguerra, pero ¿qué se puede entender por un ciberataque? Bueno, según IBM, un ciberataque es la explotación de un sistema y sus redes usando *software* malicioso (*malware*) para comprometer los datos o inhabilitar las operaciones. Los ciberataques se despliegan en cibercrimen, como robo de información, fraude o secuestro de información (*ransom)* (IBM Services, 2018).

A nivel mundial las organizaciones se ven continuamente expuestas al riesgo de presentar algún tipo de ciberataque a su infraestructura. Según Symantec en su reporte anual del 2017 se encontraron 528 organizaciones alrededor del mundo que fueron afectadas o comprometidas por algún tipo de ciberataque. Colombia fue el sexto país de Latinoamérica con la mayor cantidad de ataques cibernéticos y en el 2016 fue el cuarto en la misma región (El Colombiano, 2018). Lo anterior prueba que es necesario que las organizaciones tengan planes de acción y de respuesta ante cualquier situación que puede poner en riesgo la continuidad de operación de la organización.

Uno de los retos actuales de la ciberseguridad dentro de las organizaciones es mitigar o eliminar el riesgo dentro de su infraestructura, para lograr esto es importante entender que tarde o temprano la organización va a ser atacada y es importante detectar el ataque tan pronto como sea posible (Cole & Northcutt, 2018). Una de las estrategias que se pueden tener en cuenta es el uso de *honeypots*, cuya función es simular entidades dentro de una red para mejorar la detección y monitorear sus conexiones (Vollmer & Manic, 2014).

Los *honeypots* son sistemas que guardan información relevante sobre posibles ataques para que luego las organizaciones utilicen toda esta información para mejorar sus defensas contra nuevos posibles ataques (Cole & Northcutt, 2018).Los *honeypots* son sistemas que guardan información relevante sobre posibles ataques para que luego las organizaciones utilicen toda esta información para mejorar sus defensas contra nuevos posibles ataques (Cole & Northcutt, 2018).

Para llevar a cabo estas mejoras es necesario la intervención de una persona calificada que interprete estos datos y pueda tomar decisiones sobre qué medidas se deben admitir para mitigar los riesgos dentro de la infraestructura real, ya que la función del *Honeypot* se limita a almacenar información. Debido a esto existe la necesidad de crear *Honeypots* inteligentes de dificultad incremental que por medio de herramientas de inteligencia computacional ayuden a mitigar los ataques cibernéticos mediante el análisis y el procesamiento de los datos provenientes de este, para luego ser implementados en infraestructuras reales.

1.2 JUSTIFICACIÓN

En el mundo moderno con los desarrollos tecnológicos actuales, la ciberseguridad se ha convertido en una necesidad real para las instituciones para garantizar la continuidad de sus operaciones (Baykara & Das, 2018).

Según la revista Forbes (Eubanks, 2017) se estima que el cibercrimen costará aproximadamente 3 trillones de dólares por año hasta el 2021, además de que las empresas gastan un 87% del presupuesto digital en su seguridad.

La prevención y detección de posibles ataques son los dos pilares fundamentales para la implementación de la seguridad de una organización. La detección tiene un grave problema en la actualidad ya que toma aproximadamente 101 días darse cuenta que un sistema fue comprometido (Mandiant, 2018)

Frente a la prevención el panorama también es complicado todo esto debido en que diariamente surgen nuevos tipos de ataques o variaciones a versiones anteriores para los cuales los sistemas no se encuentran protegidos.

Los Honeypots son efectivos en la detección de ataques porque al tratarse de sistemas aislados cualquier conexión es sospechosa y debidamente registrada, lo que permite un posterior análisis y posible descubrimiento de nuevas herramientas y ataques usados por los Cibercriminales (Spitzner, 2002). Pero para la prevención es necesario el estudio de la información recaudada por el Honeypot para tomar las medidas necesarias, debido a que el Honeypot solo es capaz de almacenar dicha información.

Si el Honeypot tuviese cierta autonomía y fuese capaz de tomar decisiones que corrigieran las brechas de seguridad que en él se encuentren, el tiempo entre la detección de un nuevo ataque y la prevención podría reducirse, ya que solo bastaría con copiar las configuraciones que el Honeypot ha tomado e implementarlas en la infraestructura real.

1.3 OBJETIVOS DEL PROYECTO

1.3.1 Objetivo General

Diseñar un prototipo de un *honeypot* inteligente mediante algoritmos de inteligencia computacional que sea capaz de aprender y aumentar la dificultad durante el proceso de ataque a sistemas.

1.3.2 Objetivos Específicos

1. Simular un ambiente en el que el *honeypot* se encuentre imitando un servidor real.

2. Automatizar el proceso de recolección de información dentro del *honeypot*.

3. Procesar la información obtenida mediante un algoritmo de inteligencia computacional previamente definido.

1.4 MARCO DE REFERENCIA

1.4.1 Antecedentes

En el artículo "*A review of dynamic and intelligent honeypots",* por Wira Zanoramy Ansiry Zakaria, Miss Laiha Mat Kiah, se menciona el tema de *honeypots* dinámicos o inteligentes, los cuales utilizan algoritmos de inteligencia artificial a la hora del despliegue de los mismos, los *honeypots* dinámicos, a diferencia de los estáticos continúan monitoreando la red en busca de cambios y en caso de ser así se actualiza su configuración (Zakaria & Kiah, 2013). Allí se explican algunas herramientas desarrolladas para la recolección de información relacionada con los ataques recibidos por los sistemas y crear *scripts* de configuración dentro de los servidores. Adicionalmente se plantea un panorama futuro donde se realizan investigaciones en temas relacionados usando diferentes algoritmos de inteligencia artificial.

En otro trabajo llamado "*Applying AI to Improve The Performance of Client Honeypots",* realizado por Van Lam Le, Peter Komisarczuk, Xiaoying Sharon Gao en la *Victoria University of Wellington,* también se utiliza la inteligencia artificial, pero en este caso es usada para la detección de sitios web maliciosos que buscaban infectar a quienes los visitan (Le, Komisarczuk, & Gao, 2009).

Para esta investigación se utilizaron honeypots que simulaban ser clientes que visitan un sitio web y analizaban su respuesta. Inicialmente se utilizaron honeypots de bajo nivel, quienes, utilizando una heurística, clasifican como posibles sospechosos a aquellos sitios web con algún indicio de respuesta maliciosa. Luego, honeypots de alto nivel eran usados para interactuar con estos sitios web simulando un sistema operativo real, finalmente toda esta información era recolectada y procesada por medio de algoritmos de inteligencia artificial para su posterior clasificación (Le et al., 2009).

En el *paper* "*HSNORT: A Hybrid Intrusion Detection System using Artificial Intelligence with Snort"* por Divya Surender Lakra se propuso una metodología para aplicar inteligencia artificial dentro de sistemas de detección de intrusiones, para detectar ataques mientras o luego de que ocurran. Para este trabajo se utilizó una herramienta llamada *Snort* la cual permite almacenar todo el tráfico de red para su posterior análisis. En este se usó una red neuronal para clasificar paquetes de red comparándolos con patrones de ataques usados previamente para su entrenamiento. La idea de la red era predecir la siguiente acción o comando de un usuario, teniendo en cuenta las *n* acciones o comandos anteriores (Lakra, 2013).

1.4.2 Marco teórico

1.4.2.1 Ataques informáticos

Según la ISO/IEC en el estándar 27000, un ataque está definido como el intento de destruir, exponer, alterar, deshabilitar, robar o ganar acceso o hacer uso no autorizado de cualquier elemento tecnológico que tenga valor para la organización (ISO/IEC, 2009).

- **Inyección SQL (SQLi)**

  Un ataque de inyección SQL consiste en la inserción de una consulta SQL a través de un campo de datos en un cliente de aplicación. Una inyección SQL exitosa permite al atacante leer y modificar datos sensibles de la base de datos, así como ejecutar comandos en el sistema operativo. Estos comandos son ingresados en entradas de texto plano para afectar la ejecución de comandos SQL predefinidos por los desarrolladores (OWASP, 2016).

- **Inclusión de archivos locales (LFI)**

  La inclusión de archivos locales es el proceso mediante el cual se introducen archivos que ya existen localmente en el servidor, a través de la explotación de procesos de inclusión mal implementados dentro de la aplicación. Esta vulnerabilidad ocurre cuando una página recibe como entrada la ruta a un archivo presente en el servidor (OWASP, 2017).

- ***Cross-site Scripting* (XSS)**

  Los ataques de *Cross-site Scripting* son un tipo de inyección en la cual se inserta código malicioso en un sitio web de confianza. Los ataques XSS ocurren cuando un atacante usa una aplicación web para enviar código malicioso, en forma de *scripts* que se ejecutarán del lado del cliente, estos ataques ocurren cuando la aplicación usa información ingresada por el usuario y con la salida renderiza nueva información en la página (OWASP, 2018b).

- **Inyección de comandos**

  La meta de los ataques de inyección de comandos es ejecutar comandos arbitrarios dentro del sistema operativo *host* a través de la aplicación vulnerable. Estos ataques son posibles gracias a que la aplicación pasa entradas inseguras provistas por el usuario (formularios, *cookies*, encabezados de HTTP, entre otros) a una *Shell* en el sistema. En este ataque, los comandos suelen ser ejecutados con los permisos que tiene la

aplicación vulnerable. La vulnerabilidad es posible, en gran parte, por la falta de validación de los datos que ingresa el usuario (OWASP, 2018a).

1.4.2.2 Modelo OSI

Según Olifer y Olifer (2009) en el libro Redes Computacionales, a finales de la década de 1970 existían múltiples protocolos propietarios de comunicación entre computadoras, esta gran variedad de herramientas generó gran incompatibilidad de dispositivos que utilizaban diferentes protocolos. Por esta razón, se creó una pila de protocolos unificados creados con el fin de compensar las desventajas de las pilas existentes, de esta forma se originó el modelo OSI (o modelo de referencia), el cual no contiene descripciones de ninguna pila de protocolos en específico, en su lugar, proporciona una descripción generalizada de las herramientas de interconectividad entre redes (Olifer & Olifer, 2009).

Las aplicaciones establecen los protocolos de interacción mediante el uso del conjunto de las siete capas de herramientas en el modelo, estas capaz son

- Capa física: Transmisión de datos mediante enlaces físicos, como cables, entre otros.
- Capa de enlace de datos: Capa que se encarga de la conmutación de los paquetes.
- Capa de red: Genera un sistema de transporte unificado que conecta varias redes, también conocido como Internet.
- Capa de transporte: Esta capa proporciona a las superiores del modelo un servicio de transmisión con el nivel de confiabilidad requerido.
- Capa de sesión: Esta capa registra la instancia activa y proporciona las herramientas necesarias para sincronizar la sesión.
- Capa de presentación: se encarga de presentar la información, sin cambios, que se transmitió a través de la red.
- Capa de aplicación: Esta capa consiste en un conjunto de protocolos que le permiten al usuario acceder a diferentes recursos en la red, como lo son archivos, impresoras, entre otros.

1.4.2.3 Protocolos de capa de aplicación

La computación orientada a servicios (SOC) es un paradigma que permite a las organizaciones exponer sus competencias como servicios para facilitar la cooperación con sus socios. Este paradigma hace posible agregar múltiples como componentes para crear sistemas más complejos (De Madeiros, Rosa, & Ferreira Pires, 2012).

- **Correo (SMTP y POP3)**

El correo fue una de las aplicaciones por las que se creó el Internet, para soportar otros protocolos existentes como la transferencia de archivos y el acceso remoto a computadores. El correo es un set de protocolos interconectados que corren en los clientes y servidores para proveer una mezcla de buzones, lectores y escritores en los que depende el correo (Goralski, 2017).

- **Servidor Web**

  El servidor web está basado en el protocolo de capa de aplicación HTTP, que a su vez utiliza TCP, el cual espera por las consultas de los clientes web (navegadores) que exploran el puerto bien conocido 80.

  Un servidor web está basado en un protocolo de la capa de aplicación llamado HTTP, este está embebido en el protocolo TCP. Este servidor espera que los clientes web, como navegadores, exploren su contenido por el puerto 80, o bien, por el puerto 4431 que utiliza el protocolo HTTPS (Olifer & Olifer, 2009).

1.4.2.4 Honeypot

Un honeypot es un recurso de seguridad en el cual su valor reside en ser investigado, atacado o comprometido (Spitzner, 2002). Los honeypots son sistemas engañosos, que emulan entidades de una red crítica donde han sido desplegados, para mejorar la detección y el monitoreo de ataques informáticos (Vollmer & Manic, 2014).

Los honeypots pueden ser divididos en tres grupos en dependencia al nivel de interacción que le puedan ofrecer al atacante. Por lo tanto, se pueden clasificar en bajo, medio y alto nivel (Spitzner, 2002)

- Honeypots de bajo nivel de interacción

  Estos honeypots son en general fáciles de instalar debido a su simple diseño y básica funcionalidad. La gran mayoría puede llegar a simular una gran cantidad de servicios, permitiéndole al atacante cierta interacción con éstos. Al tratarse de un ambiente simulado y al no existir ningún sistema operativo este tipo de honeypot tiene el menor nivel de riesgo ya que no se puede llegar a tener acceso al servidor para posteriores ataques (Spitzner, 2002).

- Honeypots de nivel medio de interacción

  Los honeypots de nivel intermedio tienen una mayor cantidad de funcionalidades en comparación con los de bajo nivel. En estos honeypots al atacante se le muestra lo que está esperando recibir luego de haber realizado una acción. Estos honeypots tienen un nivel más elevado de riesgo ya que

al permitirle mayor interacción al atacante, éste podría llegar a tener acceso al sistema operativo real (Spitzner, 2002).

- Honeypots de alto nivel de interacción

Los honeypots de alto nivel permite obtener una gran cantidad de información del atacante, pero son mucho más complicados de configurar y de mantener, además de tener el nivel más alto de riesgo al tratarse de un sistema operativo real. Estos honeypots pueden llegar a permitir identificar nuevas herramientas y tipos de ataques que utilizan los atacantes ya que se interactúa con un ambiente muy similar a los que existen en la industria (Spitzner, 2002).

1.4.2.5 Inteligencia artificial

La inteligencia artificial se define como "La automatización de actividades que vinculamos con procesos del pensamiento humano, actividades tales como toma de decisiones, resolución de problemas, aprendizaje…" (Bellman, 1978).

- Clasificadores

Describe el problema de asignar una categoría a cada ítem, el número de categorías suele ser un poco menos que unos cientos, pero puede volverse mucho más complejo e incluso no tener relaciones como en la clasificación de texto o reconocimiento de voz (Mohri, Rostamizadeh, & Talwalkar, 2018).

- Problemas supervisados
  El sistema de aprendizaje recibe un set de datos de ejemplo etiquetados y realiza predicciones de todos los nuevos puntos que aún no han sido vistos. Este escenario es muy común en problemas asociados a clasificación, regresión y puntaje (Mohri et al., 2018)

- Problemas no supervisados
  El sistema de aprendizaje recibe exclusivamente datos sin etiquetar y realiza las predicciones sobre todos los datos no vistos, aunque puede ser complicado medir cuantitativamente el desempeño del aprendizaje (Mohri et al., 2018).

- Métodos de Clasificación

  - Naive Bayes y Redes Bayesianas

El aprendizaje Bayesiano ataca el problema de construir una hipótesis de los datos como un subproblema de una problemática más fundamental como lo es realizar predicciones. La idea es usar la hipótesis como intermediario entre los datos y la hipótesis. Primero, la probabilidad de cada hipótesis es estimada, dados los datos, las predicciones son hechas a partir de la hipótesis usando las probabilidades posteriores de la hipótesis como peso de las predicciones (Russel & Norvig, 2011).

- Árboles de decisión
  Los árboles de decisiones son algunos de los algoritmos de aprendizaje más simples, pero más efectivos. Primero se le describe un elemento de rendimiento y luego se le enseña como aprenderlo (Russel & Norvig, 2011).

- Sistemas basados en casos

- Redes Neuronales

  La diferencia entre una máquina conexionista, es decir, una máquina neuronal y los programas de computador convencionales es que éstas "elaboran" en cierta medida, la información de entrada para obtener una salida o respuesta.

  Existen modelos muy diversos en los cuales se siguen diferentes filosofías de diseño, reglas de aprendizaje y funciones de construcción. Una primera clasificación se realiza en función del recorrido que sigue la información dentro de la red, así se distinguen redes alimentadas hacia adelanta y redes con reto-alimentación (Isasi & Galván, 2004).

- Máquina de soporte vectorial (SVM)

  El método SVM es realizado para estimar la función que clasifica los datos en dos clases. SVM Está basado en la idea de minimizar el error de generalización cuando se aplica el clasificador a muestras para pruebas, que no concuerdan con ninguna muestra usada para entrenar el clasificador. En comparación la mayoría de los clasificadores, como redes neuronales, tratan de minimizar el error de entrenamiento y tienen a sobrentrenarse con los datos de entrenamiento (De Oliveira Martins, Braz Junior, Correa Silva, Cardoso de Paiva, & Gattass, 2009).

2. METODOLOGÍA

1. Simular un ambiente en el que el honeypot se encuentre imitando un servidor real.
   1.1. Se desplegó un ambiente que permitió la recolección de datos de los diferentes servicios dentro del servidor.
   1.2. Se creó un segundo servidor donde se almacenó la información del servidor de prueba.
   1.3. Se creó una base de datos donde se guardaron los registros de los ataques realizados sobre el servidor de prueba.
2. Automatizar el proceso de recolección de información dentro del honeypot.
   2.1. Se realizaron *scripts* que realizan el análisis necesario por cada tipo de *log*.
   2.2. Se insertó en la base de datos la información ya *parseada.*
3. Procesar la información obtenida mediante un algoritmo de inteligencia computacional previamente definido.
   3.1. Se realizó preprocesamiento de la información.
   3.2. Se realizó un análisis de la información mediante diferentes técnicas.
   3.3. Se realizaron pruebas con diferentes algoritmos
   3.4. Se escogieron múltiples algoritmos a partir de los resultados de las pruebas

3. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

3.1 SIMULACIÓN DEL AMBIENTE PARA LA RECOLECCIÓN DE DATOS

En un principio, se hizo una recolección de datos, los cuales serían empleados para realizar un proceso de entrenamiento de una Inteligencia computacional. Para esto, se planteó la posibilidad de exponer un servidor con diferentes servicios web, como servidores HTTP, FTP, MySQL, entre otros. La idea de tener el servidor en un ambiente público era permitir que fuese atacado desde internet, dado que poseía una IP pública, cualquier persona o bot que estuviera recorriendo la red podría encontrarlo y realizar diferentes ataques sobre este.

Este primer servidor se encontraba alojado en *Linode,* y se comunicaba con otro servidor mediante Rsyslog. Este segundo servidor se encargaba de almacenar la información correspondiente a los logs de los servicios que habían sido expuestos en el primer servidor de prueba. Al cabo de unos meses de observación, se obtuvo que los registros de los ataques que habían sido realizados sobre este servidor, aunque representaban una cantidad de información bastante extensa, al haber sido realizados por personajes sin identificación no era factible analizar las secuencias de los ataques y determinar de esta manera, esa firma única que crea un atacante al intentar vulnerar un sistema.

Por esto se optó por desplegar una máquina virtual conocida como bWAPP. esta es distribuida por *IT Sec Games* y posee múltiples vulnerabilidades reportadas y documentadas. Esta máquina se desplegó mediante el uso de tecnologías de virtualización dentro de los equipos personales, donde no era accesible por nadie externo de la red para luego obtener nuevamente los registros de sus ataques. Estos ataques se registraron/almacenaron en una base de datos para su posterior procesamiento y clasificación.

Figura 1 Máquina virtual bWAPP

En esta fase, se pudo recrear un ambiente donde se realizaron las pruebas y los ataques que generarían los datos para su posterior procesamiento, finalmente fue más efectivo realizar estos ataques de forma controlada en lugar de permitir tráfico anónimo en internet, ya que se podían revisar los logs e identificar que había sido un ataque realizado por nuestra parte, en lugar de tratar de descifrar que estaba realizando un atacante anónimo o un *bot* recorriendo la red.

3.2 AUTOMATIZACIÓN DE LA RECOLECCIÓN DE LOS DATOS

Para el preprocesamiento de la información se realizaron múltiples *scripts* que permitían el correcto análisis, para posterior almacenamiento, de los diferentes *logs* del sistema en diferentes atributos, estos atributos son: fecha, IP fuente y destino, puerto fuente y destino, estado TCP de la conexión, log del cual se originó el registro y una descripción donde se almacena la información adicional que brindan los logs como el qué ejecutó la petición realizada.

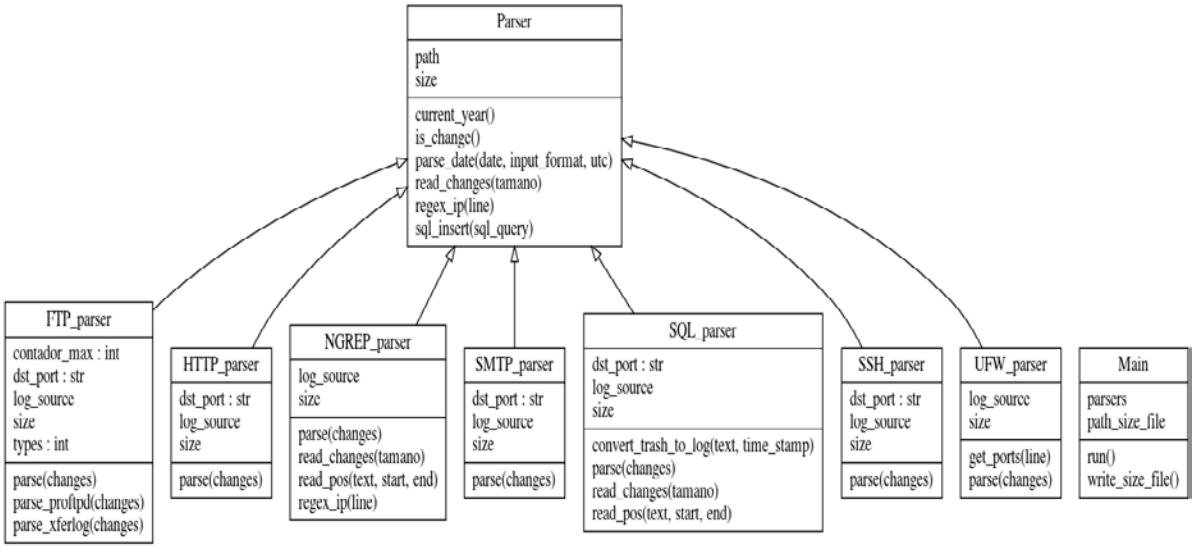

Figura 2: Diagrama de clases de los scripts para el parsing

Como se puede observar en la Figura 2, existen diferentes clases que cumplen con las diferentes funcionalidades del programa. La clase padre, *Parser*, se encarga de la inserción de los datos en la base de datos, también se encarga de validar si existen cambios dentro de los archivos de *log* del sistema para realizar la inserción de la nueva información. De esta clase extienden múltiples clases hijo, las cuales están encargadas de leer la información del servicio correspondiente, por ejemplo, la clase "*SQL parser*" es la encargada de leer el archivo de *logs* del servicio de MySQL, encontrar donde se encuentra cada uno de los atributos anteriormente mencionados, separar y almacenar cada uno de estos temporalmente, para luego ser insertados en la base de datos.

Este funcionamiento es similar para cada uno de los servicios expuestos en la máquina virtual, pero es dependiente de la forma en la que éstos registran la información dentro del sistema, por lo que no se podía realizar un *parser* único para cada servicio.

Finalmente existe la clase "*Main*", la cual es la encargada de crear los objetos de cada uno de los *parsers* y estar continuamente revisando si existe un nuevo cambio dentro de alguno de los *logs* para invocar los métodos correspondientes del *parser* donde se encuentre una nueva entrada.

Los datos de los *logs* de cada uno de los servicios vienen en un texto plano, donde una línea indica una acción dentro del servicio correspondiente, esta información es difícil de procesar de esta forma, por lo que era necesario realizar este código para poder almacenar la información de forma organizada y que fuese útil para el entrenamiento de la inteligencia computacional a seleccionar.

3.3 ANÁLISIS DE LOS DATOS MEDIANTE ALGORITMOS DE INTELIGENCIA

COMPUTACIONAL

En esta sección, se realizaron múltiples intentos para determinar cuál sería la mejor opción para desarrollar una inteligencia computacional que pudiese diferenciar entre una petición clasificada como ataque y una común, por lo que se recurrió a técnicas de analítica de datos, como lo son: nubes de palabras, expresiones regulares, histogramas de palabras, *bigrams* y *ngrams* y funciones de distancia.

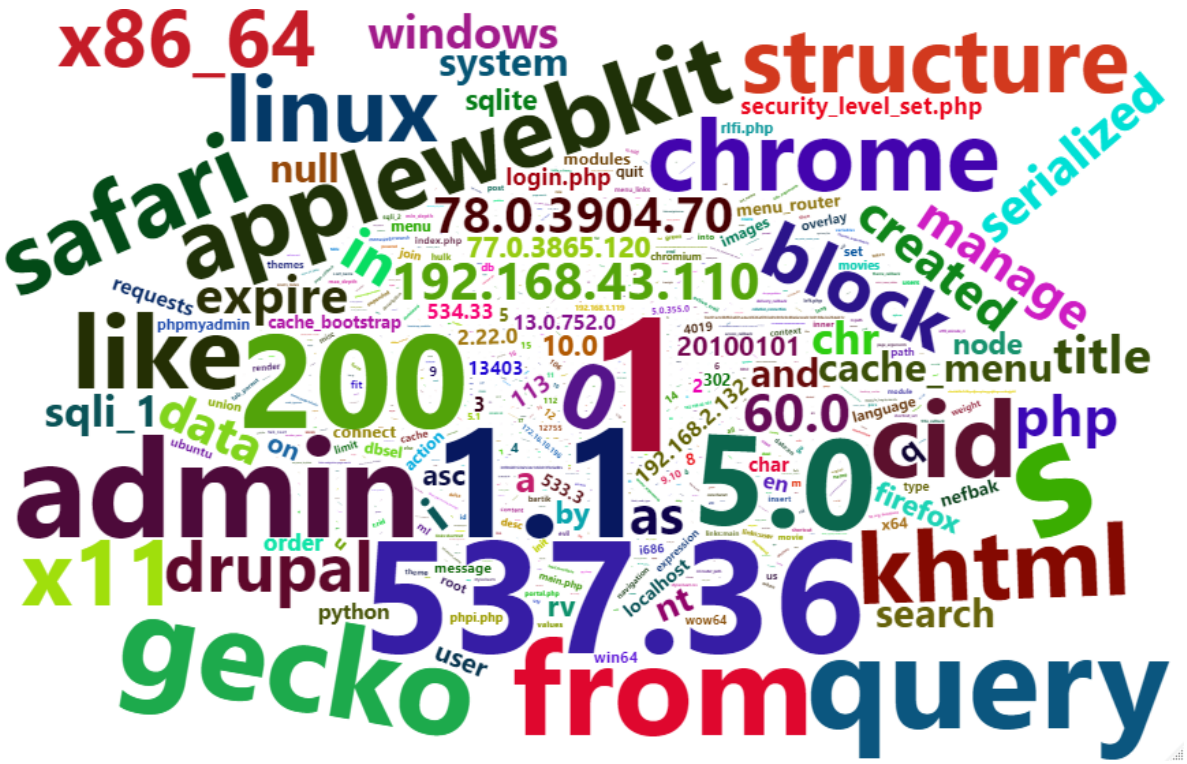

Figura 3 Nube de palabras

Inicialmente se analizó la información haciendo uso de R© (v 3.6.1), allí se realizó una nube de palabras donde se podía observar la frecuencia de cada una de las palabras, para así poder determinar cuáles palabras estaban generando ruido y cuales eran realmente importantes, también se generaron *bigrams* y *ngrams*, donde se podía observar la relación entre dos o más palabras, verificando si estás podían generar patrones que pudiesen ser utilizados al momento del procesamiento de los datos.

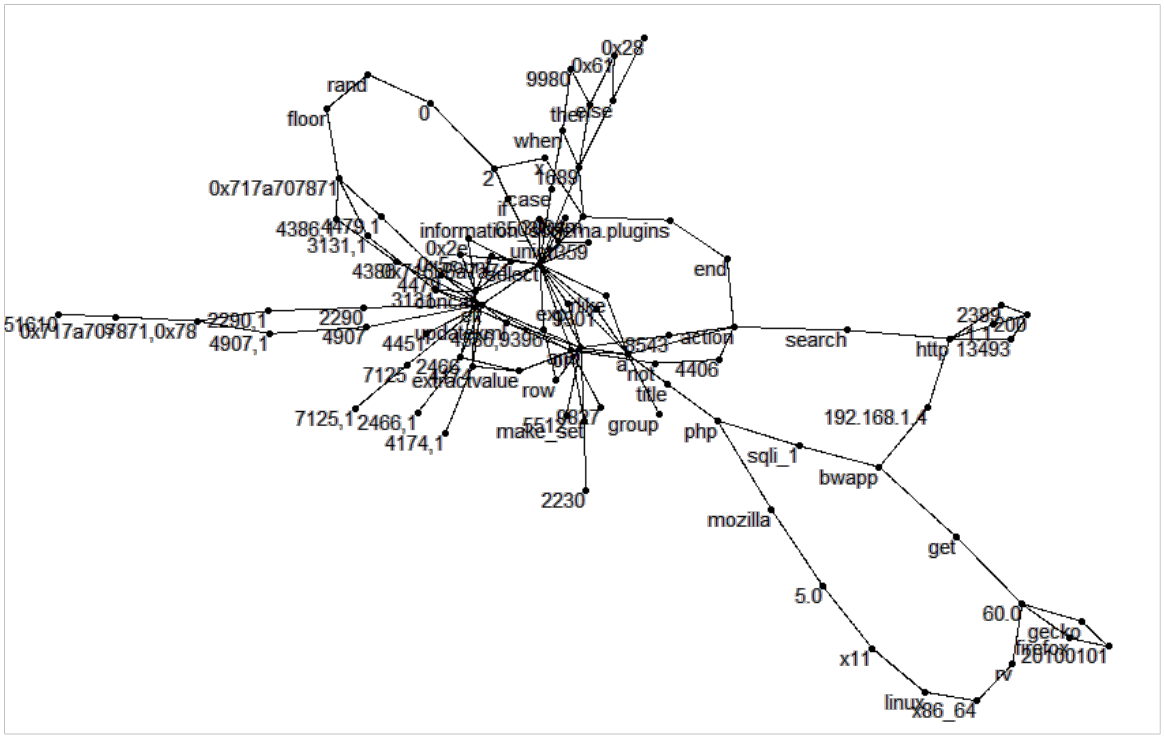

Figura 4 Ngrama con las palabras encontradas en los logs

Luego de llevar a cabo estos procedimientos, se observó que debido a la gran cantidad de datos y la cantidad de palabras repetidas era complejo identificar lo que anteriormente se planteaba, ya que había demasiadas conexiones como se puede observar en la Figura 4.

Por lo tanto, se optó por realizar expresiones regulares para identificar cuando una petición era maliciosa, pero debido a la alta complejidad de estas expresiones y a las variaciones de los ataques se consideró que no era la mejor opción.

Finalmente, se utilizó el *framework* de analítica de datos, Orange, donde usando las diferentes herramientas se creó una “bolsa de palabras”, la cual luego fue procesada por diferentes algoritmos de inteligencia artificial. Este *framework* nos permite realizar la comparación del rendimiento de múltiples algoritmos de inteligencia computacional para poder determinar cuál era el que más se adecuaba a los datos que poseíamos.

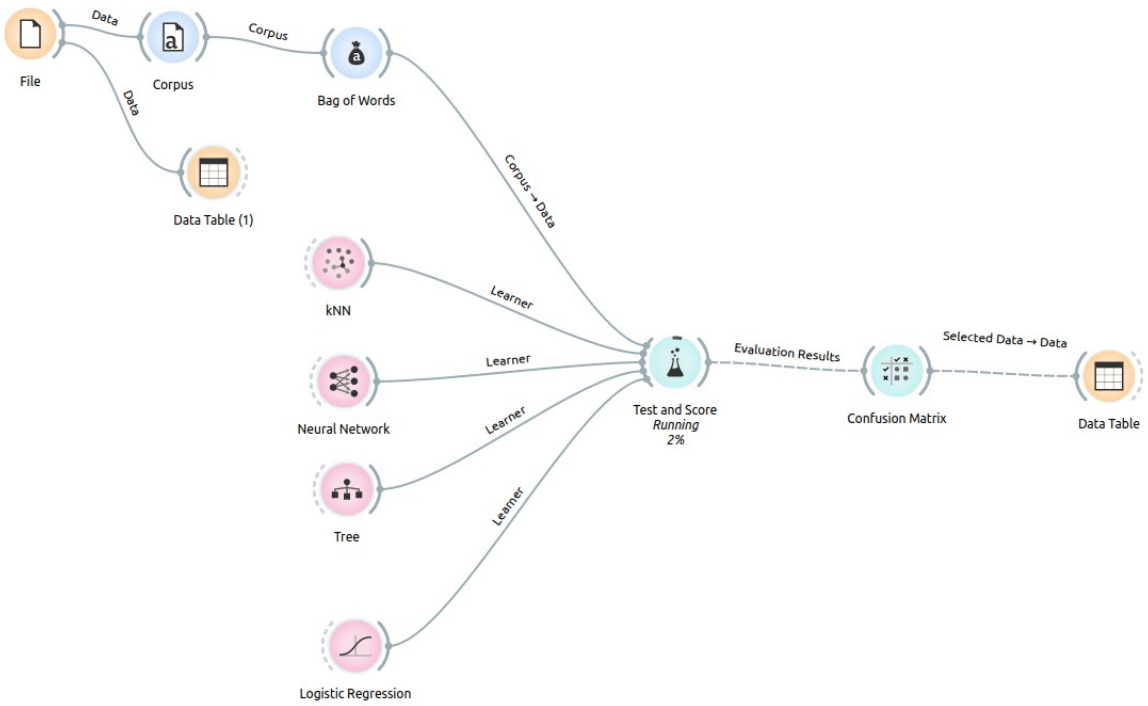

Figura 5 Entrenamiento de los modelos de clasificación en Orange

Para esto, se combinó en una tabla la información de los diferentes ataques que se poseían, indicándole a los algoritmos que entrada era considerada un ataque y que tipo de ataque. Todo esto con el fin de usar un algoritmo de inteligencia artificial supervisado, el cual necesita que la información de entrada esté clasificada, para luego realizar el entrenamiento necesario y determinar si una nueva entrada corresponde a alguno de los ataques ingresados en la fase de entrenamiento.

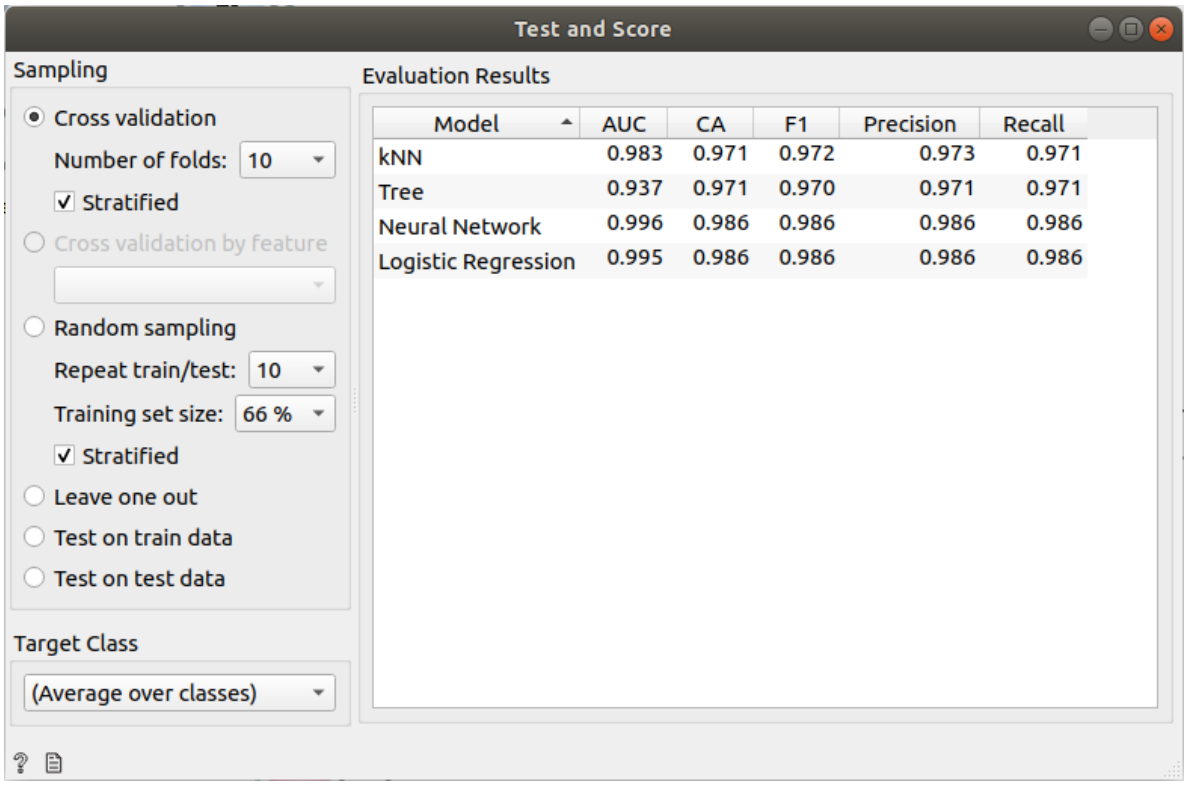

Figura 6: Resultados de la predicción de los datos multiclase

En la Figura 6 se puede observar el rendimiento de cada uno de los algoritmos utilizados, probando cual proveía una mejor clasificación de la información, esto nos dio una idea de que algoritmo podría ser usado para clasificar la información obtenida.

Se pudo obtener resultados buenos en cuanto a la clasificación de los *logs*, por lo que es posible utilizar alguno de los algoritmos presentados en la Figura 6, cualquiera de estos realizará una correcta clasificación de los logs, indicando a qué tipo de ataque pertenece según los datos iniciales que le fueron otorgados.

3.4 CREACIÓN DE UN SERVICIO DE CLASIFICACIÓN

Como paso final, los modelos anteriores fueron programados en el lenguaje de programación Python (v 3.6), para esto se utilizó la librería *Sklearn*, con esta se construyó un modelo *bag of words* usando la función *count vectorizer*, la cual permite crear un arreglo donde se cuentan las repeticiones de cada característica para cada uno de los datos de entrenamiento, convirtiéndose esto en los datos de entrada de los modelos.

Se usaron cuatro modelos que fueron entrenados con los datos recolectados y estos fueron almacenados dentro de archivos para poder ser usados a la hora de predicción. Los cuatro modelos tuvieron un desempeño por encima del 90%.

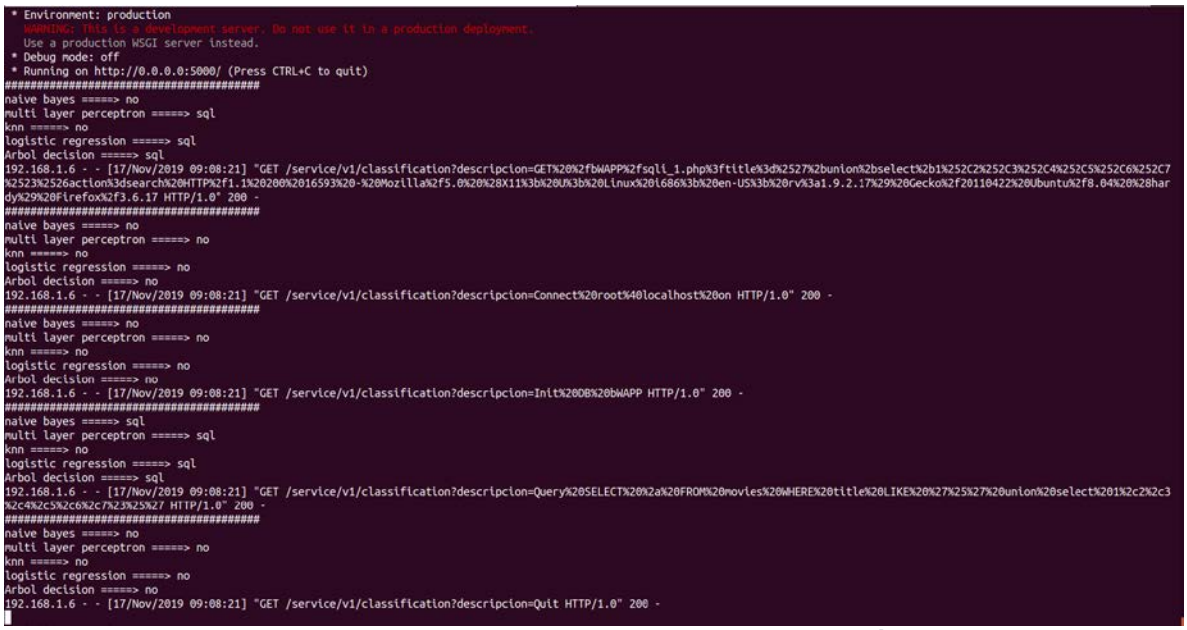

Figura 7 Resultados del servicio de clasificación

Para que la clasificación pudiera ser usada en cualquier momento, se creó un microservicio en otro servidor que es consumido por el honeypot en tiempo real, permitiendo a este saber si el log actual es o no un ataque. Usando la librería *Flask*, se creó el servicio que recibe la descripción del log vía GET, la cual primero es procesada para extraer de ella los mismos atributos del modelo, es decir la cantidad de veces que se repiten las palabras que fueron extraídas por el modelo durante la fase de entrenamiento, como paso siguiente utiliza esto para predecir el resultado en cada uno de los modelos y retorna la predicción hecha por cada uno.

4. CONCLUSIONES Y CONSIDERACIONES FINALES

Se pudo evidenciar que los diferentes algoritmos que se probaron daban resultados bastante buenos, ya que lo índices de precisión y las matrices de confusión indicaban que había poco error. También se pudo evidenciar que el valor de precisión no era alto debido a un sobre entrenamiento, porque el algoritmo continuaba clasificando correctamente nuevos intentos de ataques.

Debido a que la clasificación de hizo de manera correcta, fue posible crear un servicio que facilitara la clasificación de nuevos ataques, donde solo es necesario enviarle la información de una nueva entrada de los logs.

Como consideraciones, se debe tener en cuenta utilizar información lo más general posible, ya que inicialmente se encontraron sesgos en la información debido a la naturaleza de esta misma, se deben tratar de generalizar la extracción de características, ya que se estaba entrenando con información propia de la máquina virtual, la cual generaba estos sesgos. Pero finalmente, fue posible limpiar la información para eliminar errores.

Otra consideración para tomar en cuenta que, debido a la falta de tiempo y la amplia gama de ataques existentes, en esta tesis, se realizó el entrenamiento de la inteligencia artificial con tres tipos de ataques principalmente, los cuales fueron inclusión de archivos locales, inyección de código (PHP) y inyección SQL. Queda abierta la posibilidad de realizar el entrenamiento de la inteligencia artificial para otro tipo de ataques, pero se deben realizar filtros sobre la información para evitar el sobre entrenamiento y la posible clasificación errónea de información.

Finalmente, se puede concluir que es posible realizar la correcta clasificación de los *logs* de un entorno web, evidenciando cuáles son posibles ataques a la infraestructura y cuáles no, todo esto mediante algoritmos supervisados de clasificación.

REFERENCIAS

Baykara, M., & Das, R. (2018). A novel honeypot based security approach for real-time intrusion detection and prevention systems. *Journal of Information Security and Applications*, *41*, 103–116. https://doi.org/10.1016/j.jisa.2018.06.004

Bellman, R. (1978). *An Introduction To Artificial Intelligence*. Recuperado de https://mitpress.ublish.com/ereader/7093/?preview=#page/1

Cole, E., & Northcutt, S. (2018). Honeypots: A Security Manager's Guide to Honeypots. Recuperado el 15 de agosto de 2018, de https://www.sans.edu/cyber-research/security-laboratory/article/honeypots-guide

De Madeiros, R. W. A., Rosa, N. S., & Ferreira Pires, L. (2012). *International Journal of Services Computing (ISSN 2330-4472) Vol. 4, No. 4, October-December2016*. *4*(4).

De Oliveira Martins, L., Braz Junior, G., Correa Silva, A., Cardoso de Paiva, A., & Gattass, M. (2009). Detection of Masses in Digital Mammograms using K-Means and Support Vector Machine. *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, *8*(2), 39. https://doi.org/10.5565/rev/elcvia.216

El Colombiano. (2018). Colombia, el sexto país con más ciberataques en 2017. Recuperado el 14 de agosto de 2018, de http://www.elcolombiano.com/colombia/ciberataques-en-colombia-sexto-pais-mas-vulnerable-en-la-region-AB8535174

Eubanks, N. (2017). The True Cost Of Cybercrime For Businesses. Recuperado el 14 de agosto de 2018, de https://www.forbes.com/sites/theyec/2017/07/13/the-true-cost-of-cybercrime-for-businesses/#1c966a0a4947

Goralski, W. (2017). *The Illustrated Network*. 637–659. https://doi.org/10.1016/B978-0-12-811027-0.00025-4

IBM Services. (2018). Learn about cyber attacks and how to defend against them. Recuperado el 17 de noviembre de 2019, de https://www.ibm.com/services/business-continuity/cyber-attack

Isasi, P., & Galván, I. (2004). *Redes de neuronas artificiales Un enfoque práctico*.

ISO/IEC. (2009). *Information technology-Security techniques-Information security management systems-Overview and vocabulary*. https://doi.org/10.1038/nnano.2008.304

Lakra, S. (2013). HSNORT : A Hybrid Intrusion Detection System using Artificial Intelligence with Snort. *International Journal of Computer Technology & Applications*, *4*(June), 466–470. Recuperado de www.ijcta.com

Le, V. L., Komisarczuk, P., & Gao, X. S. (2009). Applying AI to Improve The Performance of Client Honeypots. *Victoria*.

Mandiant. (2018). M-Trends 2018 Report. *M*, 1–52.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of Machine Learning*. Recuperado de https://mitpress.ublish.com/ereader/7093/?preview=#page/1

Olifer, N., & Olifer, V. (2009). *Redes de computadoras*. https://doi.org/10.15713/ins.mmj.3

OWASP. (2016). SQL Injection. Recuperado el 27 de octubre de 2019, de https://www.owasp.org/index.php/SQL_Injection

OWASP. (2017). Testing for Local File Inclusion. Recuperado el 27 de octubre de 2019, de https://www.owasp.org/index.php/SQL_Injection

OWASP. (2018a). Command Injection. Recuperado el 15 de noviembre de 2019, de https://www.owasp.org/index.php/Command_Injection

OWASP. (2018b). Cross-site Scripting (XSS). Recuperado el 27 de octubre de 2019, de https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

Russel, S. J., & Norvig, P. (2011). *Artificial Intelligence A modern approach*.

Spitzner, L. (2002). *Honeypots: Tracking Hackers* (Vol. 9).

Vollmer, T., & Manic, M. (2014). Cyber-physical system security with deceptive virtual hosts for industrial control networks. *IEEE Transactions on Industrial Informatics*, *10*(2), 1337–1347. https://doi.org/10.1109/TII.2014.2304633

Zakaria, W. Z. A., & Kiah, M. L. M. (2013). A review of dynamic and intelligent honeypots. *ScienceAsia*, *39*(SUPPL.1), 1–5. https://doi.org/10.2306/scienceasia1513-1874.2013.39S.001