DISEÑO DE UN PROTOTIPO DE UN CORRELACIONADOR DE EVENTOS DE SEGURIDAD PARA SERVIDORES

Modalidad: Exploratorio

RICARDO RESTREPO CORREA
JUAN SEBASTIAN VELASQUEZ SAAVEDRA

Trabajo de grado para optar al título de

Ingeniero de sistemas y computación

JUAN ESTEBAN VELASQUEZ MUNERA

Ingeniero informático

UNIVERSIDAD EIA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
ENVIGADO
2020

A nuestras familias, por su amor incondicional.

A mi padre: Álvaro León Velasquez Toro, a mi madre: María del Pilar Saavedra Palomino y mi hermano: Manuel Felipe Velasquez Saavedra por todo el cariño, apoyo y el ayudarme a cumplir esta importante meta.

— Juan Sebastian Velasquez Saavedra.

A mi madre: Claudia Patricia Correa Rendón, y a mi abuela: Martha Luz Agudelo, por hacer de mí la persona que hoy en día soy, sin ustedes este logro no hubiese sido posible.

— Ricardo Restrepo Correa.

AGRADECIMIENTOS

Gracias a Juan Sebastian Valencia, por habernos asesorado en lo que a inteligencia artificial respecta.

Gracias a Juan Esteban Velasquez, por habernos asesorado en lo que a seguridad informática respecta.

Gracias a Johan Vélez, por su acompañamiento y apoyo durante todos nuestros años de carrera

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

GLOSARIO

CORRELACIONADOR: Sistema para establecer un paralelismo entre dos o más cosas

LOG O EVENTO DE SEGURIDAD: Grabación secuencial de datos, ya sea en un archivo o bases de datos, de todos los acontecimientos o eventos que afectan a un proceso en particular.

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

PAYLOAD: Carga de datos transmitidos que se ejecutan para esa vulnerabilidad.

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

BAG OF WORDS O 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.

RESUMEN

En la actualidad existen muchos servicios web, es decir, páginas y funcionalidades que se prestan a través de internet, los cuales para funcionar hacen uso de servidores web, que no son más que un computador que almacena información y se encuentra conectado a una red, en este caso internet. Los clientes de estas plataformas acceden a estos servicios realizando peticiones para obtener los recursos requeridos, dicha conexión entre el cliente y el servidor queda registrada como un evento o log, que no es más que una colección de datos de procesamiento y transmisión entre la relación antes mencionada.

Así pues, un cliente puede intentar explotar alguna vulnerabilidad del servidor mediante una petición con el fin de hurtar información sensible, penetrar un inicio de sesión o hasta desconectar todo un portal. Sin embargo, dado lo anterior, su intento de ataque o su ataque en sí queda registrado como un evento o log, el cual puede contener información que dé a entender su finalidad maliciosa.

En el siguiente trabajo se plantea una solución para monitorear y dar un significado a estos eventos de seguridad, con el fin de aclarecer las intenciones de los usuarios del servicio web, por medio de un correlacionador de eventos de seguridad utilizando inteligencia artificial, el cual permita, en primer instancia, exponer de manera generalizada la información almacenada en los eventos de seguridad por medio de gráficos y contenidos visuales, como también un algoritmo inteligente que clasifique un evento de seguridad como una petición normal de un cliente común o como un ataque de un cliente malicioso, en este caso exponiendo el tipo de ataque ocurrido.

Para lograr el objetivo, se requirió de dos servidores, el primero, que fue expuesto a internet como servidor web mediante *Apache HTTP Server* con el fin de colectar eventos públicos, y el segundo, que sirvió como almacenamiento para dichos eventos utilizando una base de datos *MySQL*. Este último, se conectó a *Power BI*, donde se ilustró la información de interés con gráficos y contenidos visuales. Para el desarrollo del clasificador de eventos, se tomó un muestreo de 1000 datos por ataque y 1000 datos común y corrientes con el fin de entrenar tres modelos de inteligencia artificial y comparar sus resultados.

Los tres modelos de inteligencia artificial (árbol de decisión, regresión logística y perceptrón multicapa) clasificaron exitosamente los eventos de seguridad, dando viabilidad a esta técnica para determinar peticiones maliciosas en un servidor web.

Palabras claves: inteligencia artificial, correlacionador de eventos de seguridad, ciberataques.

ABSTRACT

Nowadays there are many web services, that is to say, pages and functionalities that are provided through the Internet, which in order to work make use of web servers, these are computers that store information and have a network connection, in this case the internet. The clients of these platforms access the services by requesting specific resources. Each attempt or successful connection between the client and the server is registered as an event or log, which is a collection of processing and transmission data between the relationship.

Thus, a client may try to exploit some vulnerability in the server by means of a request in order to steal sensitive information, penetrate a login or even disconnect an entire portal. However, given the above, their attack attempt or the attack itself is recorded as an event or log, which may contain information indicating its malicious purpose.

In the following work, a solution is proposed to monitor and provide meaning to these security events, in order to clarify the intentions of the users of the web service, this is achieved by means of a security event correlator using artificial intelligence, which allows, in the first instance, a generalized exposure of the information stored in the security events by means of graphics and visual content, as well as an intelligent algorithm that classifies a security event as a normal request from a common client or as an attack from a malicious client, in this case exposing the type of attack that has occurred.

To achieve the objective, two servers were required, the first one, which was exposed to the Internet as a web server using *Apache HTTP Server* in order to collect public events, and the second one, used as storage for these events within a *MySQL* database. The latter was connected to *Power BI*, where the information of interest was illustrated with graphics and visual content. For the development of the event classifier, a sampling of 1000 logs per attack and 1000 common logs was taken in order to train three artificial intelligence models, finally the performance of the models was compared.

The three artificial intelligence models (decision tree, logistic regression and multi-layer perception) successfully classified the security events, making this technique viable for determining malicious requests on a web server

Keywords: artificial intelligence, security event correlation, cyberattacks

INTRODUCCIÓN

De forma generalizada, en el siguiente trabajo usted como lector podrá encontrar información acerca de la realización de un prototipo de un correlacionador de eventos de seguridad que, permita facilitar la lectura de dichos *logs* para el personal encargado de la seguridad de un servidor web, y también posibilite la clasificación de un log como un ataque o como una petición cualquiera de un usuario normal.

Inicialmente se realiza una realiza una contextualización de los métodos más utilizados para correlacionar eventos, enseñando cómo se realiza la práctica actualmente y qué ventajas y desventajas tienen cada uno de ellos.

En el desarrollo del trabajo usted podrá entender cómo monitorear logs de un servidor web apache en tiempo real utilizando una base de datos *MySQL* y cómo procesar estos eventos para dar con información útil, que permita definirlos como ataques o no.

Finalmente, conocerá cómo presentar dichos datos de una forma visual utilizando *Power BI*, y podrá encontrar una manera de clasificar estos logs, bien sea como una petición común y corriente o como un ataque -en este caso conociendo qué ataqué fue-, todo esto utilizando inteligencia artificial mediante el lenguaje de programación *Python*.

1. PRELIMINARES

1.1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad existen muchos servicios web, es decir, páginas y funcionalidades que se prestan a través de internet; para los usos de comunicación, contabilidad, administración, entretenimiento, ocio, etcétera. Gracias a este se puede romper la barrera de la distancia y facilitar una conexión para el mundo y las empresas. Esta relación es de fundamental importancia para que un proyecto de como resultado una empresa, como fueron los casos de *Google*, *Facebook*, *Amazon*, *Netflix*, entre muchas otras compañías que se podrían tomar como ejemplos. Estos pequeños proyectos son conocidos con el nombre de "*start ups*" y suelen crear tecnologías genéricas que se pueden utilizar a nivel personal, en diferentes ámbitos a nivel empresarial y en campos delimitados a la negociación y afines. (Cano & Baena, 2015) y la razón principal es que estudios realizados desde 1997, entre los que destacan los realizados por Brynjolfsson y Hitt, Bresnahan, Jorgenson y Stiroh, desvelan que existe un importante cambio en la productividad de una empresa u hogar, fundamentalmente por el uso de las TIC (Eduardo Díaz Rodríguez, 2017)

Estas empresas suelen tener servidores; que son aquellas máquinas virtuales o físicas, que suelen ser muy potentes, y se ubican dentro de una red y funcionan como un lugar de almacenamiento para archivos y aplicaciones compartidas. En contraste, están las personas comunes que poseen computadoras, a estos se les suelen llamar clientes (Tanenbaum, 2003). Gracias a estos servidores hoy en día los clientes pueden visitar páginas como Facebook e Instagram, y relacionarse mediante de ellas. El rendimiento de estos servidores normalmente es mucho más potente que un computador medio (ESDS, 2010), ya que estos tienen la responsabilidad de dar disponibilidad a los usuarios que tratan de ingresar a una página web desde cualquier parte del mundo y permiten que más de mil personas revisen el mismo contenido simultáneamente. Para lograr esto. el cliente; que es un usuario final, o sea cualquier computador, impresora o celular (Oluwatosin, 2014), hace una petición para acceder a los recursos de una página web. Esta solicitud es recibida por el servidor y este revisa que él tenga los permisos necesarios. Si los tiene el servidor responde con un mensaje, enviando el recurso solicitado (Moiseenko et al., 2014). Todos estos eventos son registrados en el sistema operativo, para su posterior uso y análisis.

Los logs son una colección de datos que registra todos los datos de procesamiento y transmisión de datos (Bonatti et al., 2019), esto es, desde la petición de un cliente hasta el mensaje respuesta. Por lo tanto, si un cliente manda una petición *ICMP* (Protocolo de mensaje de control de internet); que es un mensaje que suele ser usado para revisar el estado de un cliente o servidor dentro de una red o el internet mismo (Tanenbaum, 2003).

Muchas personas y empresas desconocen o ignoran que esta dependencia que ha venido apareciendo con el tiempo trae consigo vulnerabilidades que puede llegar a causar

hasta la muerte de una compañía afectada (Valenzuela, 2018). Empezando con que cualquier empresa o persona puede realizar cualquier tipo de ataque, que podría resultar desde el hurto de información sensible, penetrar un inicio de sesión, hasta desconectar todo un portal (Cloudflare, 2020).

Hoy en día estamos en un mundo donde las TIC se ha convertido una herramienta indispensable en las organizaciones, a nivel personal y en el hogar. La ciberseguridad no puede ser menos importante, ya que se hace imperativo cuidar la información personal y organizacional. Para lograr esto hay que estar buscando constantemente una manera de mitigar los ataques y estar prevenido ante cualquier anomalía (Pérez Pérez, 2016). Un camino para estar en un estado de alerta frete a estos sucesos es correlacionar lo eventos que ocurren en el lado del servidor y mantener esta información aislada, protegida y actualizada en todo momento en un servidor diferente al principal, para su posterior análisis, categorización y toma de decisiones. Entonces, surge como pregunta ¿Cómo diseñar un correlacionador de eventos de seguridad para un servidor?

1.2 JUSTIFICACIÓN

Hoy en día muchas de las empresas existentes han migrado o extendidos sus negocios a lo que se conoce como la nube, y aquellas que no lo han hecho, se verán obligadas a hacerlo. Según estudios realizados por la alianza de software, la tasa de crecimiento de la demanda de los mercados relacionados con la tecnología de la información en los últimos años ha sido de un 70% y se espera que para el 2020 la esta tasa sea del 60% (BSA-The Software Aliance, 2018). Esto solo nos confirma la dependencia que ha surgido entre las tecnologías en la nube y las compañías, lo cual podemos evidenciar en Europa, donde más de la mitad de las empresas han adoptado las herramientas en la nube (Nelson et al., 2018).

Estas herramientas, aplicaciones, soluciones que se alojan en la nube deben tener un servidor público para su funcionamiento. He aquí donde comienzan los problemas, ya que el hecho tener una infraestructura montada en la nube hace a una empresa o persona vulnerable. Esto se puede evidenciar en las estadísticas en tiempo real que nos muestra Karpersky Lab (Kaspersky, 2018) la cual nos muestra solo una cantidad limitada de los ataques, malwares, vulnerabilidades e intrusiones detectadas, pero un gran número de cada uno de ellos.

El problema en general de la ciberseguridad no se limita únicamente a un país o una región del globo terráqueo, pues en todas partes del mundo y en todo momento, se registran alrededor de 8,580,549 ciberataques de algunos tipos (Kaspersky, 2018) que terminan por explotar vulnerabilidades en los sistemas o incluso negar el acceso a ellas, permitiendo así una extorción a cambio de recuperar la plataforma, es por esto que surge la necesidad de categorizar cualquier anomalía de la mejor manera para que su análisis sea práctico y tanto el diagnóstico como la medida asociada tengan éxito (Sobers, 2020).

Lo anterior se puede evidenciar en una de las áreas de los sistemas de información; La IoT. Este es un campo que ha venido ganando mucha fama en los últimos años, es tanto así que hoy en día es reconocida como uno de los temas con más tendencia (Lee & Lee, 2015). Se estima que para el 2021 hayan alrededor de 3.500 millones de dispositivos operando (Statista, 2020). Esto crea una brecha de seguridad, ya que todos estos dispositivos se comunican inalámbricamente y hoy no hay una forma escalable que nos permita analizar y relacionar el flujo de toda la información. Por tanto, hay una necesidad imperativa de crear una buena herramienta que permita relacionar y ver anomalías en los flujos de datos (Vaarandi et al., 2015).

Este método puede llegar a ser excelente a la hora de detectar actividad sospechosa, ya que permite revisar y analizar la mayoría de los eventos que ocurren en un sistema de información. Esto se puede observar claramente en el estudio de Risto Vaarandi. Esta muestra la efectividad de un correlacionador basado en reglas durante 172 días. Se puede observar que, de 1,636,805,087 registros recolectados se alcanzaron a clasificar y analizar unos 1,331,412,766 eventos en tan solo 14,881,059 segundos de ejecución, esto es alrededor de un 81%, de los datos recolectados. Esto equivale a unos 110 eventos por segundo y a un 3% de capacidad de procesamiento con una CPU determinada. A pesar de que estos podrían ser mejores, da una clara perspectiva y nos permite afirmar que semejante trabajo no podría ser hecho por una persona. Cada uno de estos eventos pueden tener intenciones maliciosas por lo que se hace necesario mejorar el porcentaje de clasificación y detección de flujos de datos con anomalías.

1.3 OBJETIVOS DEL PROYECTO

1.3.1 Objetivo General

Diseñar un prototipo de un correlacionador de eventos de seguridad para un servidor web en la nube.

1.3.2 Objetivos Específicos

- Identificar y caracterizar los eventos a capturar
- Definir el método de monitoreo
- Diseñar el correlacionador de eventos de seguridad
- Analizar los resultados obtenidos

1.4 MARCO DE REFERENCIA

1.4.1 Antecedentes

Amruta Ambre y Narendra shekokar en su trabajo *Insider threat Detection using Log analysis and Event Correlation*, postulan qué un correlacionador de eventos es una buena herramienta para el análisis de probabilidad de activada maliciosa o mal intencionada. Además, adicionan que una buena investigación en esta área podría desembocar en una herramienta que ayudaría a analizar actividades de diferentes dispositivos en una red y hasta el método de prevención que se debe optar en determinada situación (Ambre & Shekokar, 2015).

La idea de un correlacionador de eventos es también apoyada por Meera y geethakumari en su estudio *Event Correlation for Log Analysis in the Cloud,* en el cual afirman que estamos en una época en la cual los servicios en la nube son la tendencia de todas las empresas y por tanto hay información almacenada y procesos de minados de gran importancia. Entonces, se hace necesario un correlacionador de eventos que solucione uno de los mayores problemas de los sistemas de información, la falta de segregación. De todas maneras, el solo hecho de separar estos los eventos, no es suficiente. Se hace necesario analizar los eventos en el acto. Con esto en mente podemos desarrollar técnicas para la detección de anomalías, para identificar eventos mal intencionados que podrían concluir en un crimen (Meera & Geethakumari, 2016).

Robijah Ysof, siti Rayahayu, Shahrin son también personas que apoyan el ideal de hacer más investigaciones sobre el tema. En el trabajo *Intrusion Alert Correlation Technique Analysis for Heterogeneous Log* recomiendan profundizar en el tema, haciendo énfasis en mejorar la detección de amenazas conocidas y desconocidas, además de bajar el índice de alertas falsas que se emiten (Yusof et al., 2008).

En este orden de ideas, cristina abad, jed Taylor, Cigdem Sengul, William Yurcik, Yuanyuan zhou y ken robe afirman que los resultados de estas investigaciones son prometedores y que se debe seguir desarrollando el tema, ampliando la etapa de pruebas, es decir, obtener más datos para determinar el tamaño de la secuencia optima. Además, mencionan que sería bueno probar otros métodos y modelos de minado para la correlación. Todo esto con el fundamento que se debe mejorar la efectividad a la hora de relacionar los eventos. Es por esto que se hace necesario una correcta elección de entradas, por lo tanto, la minería de datos es necesaria para extraer la información relevante de las entradas y dejar de lado todo el ruido que generan la enorme cantidad de eventos que hay en el sistema (Abad et al., 2003).

Lavrona junto con sus compañeros nos muestran que está es una investigación para la aplicación en muchas áreas de la informática. La IoT (internet de las cosas) es solo una de

estas, gracias al gran flujo de datos que esta tiene y la importancia de estos se hace necesario analizar el flujo de la información, y detectar cualquier anomalía (Lavrova et al., 2015).

Para finalizar, se tiene el trabajo de investigación de Risto y Emin que reafirman la efectividad de este estudio, mostrando las mejores prácticas que se pueden optar cuando de correlacionar eventos se trata. Sin embargo, concluyen que aún hay mucho por investigar, para que se pueda dar una solución buena y escalable (Vaarandi et al., 2015).

1.4.2 Marco teórico

Muchos estudios se han hecho en este campo con el pasar de los años. Todos estos con el objetivo claro de mejorar las etapas en la seguridad de la información; la prevención, esto es evitar cualquier intrusión si es posible, es por esto por lo que es importante identificar las vulnerabilidades y repararlas. La detección, que nos permite saber si una intrusión ha sido detectada o no. Por último, la reacción que permite tomar acciones y responder a determinadas situaciones para que una empresa sufra lo menos que sea posible (Abad et al., 2003).

Es por esto por lo que se han desarrollado y analizado muchos métodos para descubrir la manera más optima, efectiva y con el menor número de errores posibles, para correlacionar eventos, ya que estos, implementados de forma correcta pueden llegar a decir que tipos de ataque se están realizando en cualquier momento(Abad et al., 2003). Algunos de los métodos más famosos para realizar esto son:

- **Métodos basados en similitud**

Este método compara una alerta con todas las alertas de amenaza que tengan atributos similares, por ejemplo, IP fuente y destino. Luego se relaciona con la que más similitud haya, de no sé así se crea una nueva amenaza (Yusof et al., 2008). Esto tiene ciertas ventajas y desventajas, las cuales se mostrarán a continuación en la *tabla 1*.

Tabla 1. *Ventajas y desventajas de métodos basados en similitud*

Ventajas	**Desventajas**
• Puede reducir el alto número de redundancias que hay en los datos recolectados de los múltiples sensores.	• Algunas alertas solo pueden ser detectadas si múltiples sensores se pueden detectar un mismo ataque. • No puede identificar ataques que son realizados por etapas. • No puede hacer detección de anomalías.

Tomado de (Yusof et al., 2008).

- **Escenarios predefinidos de ataques.**

Este escenario parte del hecho que existen ataques que son realizados por etapas. Se establecen una cantidad de escenarios de ataques, cada uno con una cantidad especifica de pasos. Luego cada evento es comparado con dichos escenarios y posteriormente se correlaciona. En la tabla 2 se mostrarán ventajas y desventajas de este método.

Tabla 2. *Ventajas y desventajas del método por escenarios de ataque*

Ventajas	**Desventajas**
• Puede reducir un gran número de redundancias • Puede agrupar varios grupos de alertas	• Puede generar un gran número de alarmas falsas. • Requiere que las personas agreguen los escenarios de ataques manualmente.

• Puede detectar precisamente ataques en las diferentes etapas de este.	• Es limitado a los ataques que están registrados

Tomado de (Yusof et al., 2008)

De esta misma forma hay muchos otros métodos que como se puede observar son eficaces en ciertas áreas, pero no tienen muchas falencias en otras. Es por esto que Massimo Ficco en su trabajo propone un enfoque basado en ontología, el cual hace uso de un proceso para las diferentes etapas del trabajo que son: Monitoreo, clasificación, normalización, fusión, filtrado y diagnostico (Ficco & Romano, 2010).

Para la etapa de monitoreo se propone crear un método de monitoreo que permita observar lo que se conoce como los diferentes síntomas de ataques, este método calcularía una métrica de intensidad que se llamaría "*intensity score (IS*)", este refleja que tan probabilidad es que un ataque tenga un comportamiento malicioso o no. Luego se procede a la etapa de clasificación, en la cual los síntomas son analizados, filtrados y agregados a categorías. Estos son clasificados en base en unos grupos que son propios de un método basado en ontología, los cuales son: mal uso, basados en anomalías, basados en conocimientos y actos sospechosos. Posteriormente se hace una normalización de todos estos datos que ya están monitoreados y clasificados, esto es proveer un mismo formato a todos esos registros que provienen de fuentes diferentes y que se pueden presentar en diferentes estructuras. Además, se les deben añadir información como: tiempo de registro, identificador, fuente y destino (Ficco & Romano, 2010). La etapa de fusión recibe los síntomas clasificados y los une usando cauterización basado en reglas de correlación. Al final de esta etapa se tiene se genera para cada evento se tiene una lista, donde cada ataque posible contenga los síntomas correlacionados durante una ventana de tiempo (Ficco & Romano, 2010). Posteriormente se realiza el filtrado se adopta un acercamiento basado en la confianza de los eventos, esto se hace con el objetivo de reducir el número de falsos positivos producidos en este paso. El siguiente paso es fundamental, la correlación es identificar o más bien relacionar ciertas secuencias de patrones lógicos con un ataque complejo, estos patrones están lógicamente conectados y suelen realizarse para alcanzar un objetivo específico. Por último, está la etapa más importante para las empresas; el diagnostico, este consiste en hacer y construir un reporte completo y compacto si un ataque es detectado. Este informe contiene toda la información recolectada en las fases anteriores.

2. METODOLOGÍA

1. **Identificar y caracterizar los eventos a capturar**

1.1. Se realizó una investigación con el fin de encontrar los ataques web más comunes

1.2. Se filtraron estos ataques para saber cuáles eran más factibles de replicar en un ambiente controlado

2. **Definir el método de monitoreo**

2.1. Se implementó y desplegó un servidor web

2.2. Se definieron unas propiedades por evento, para ser monitoreadas

2.3. Se implementó y desplegó otro servidor web para salvaguardar los eventos

2.4. Se implementó una conexión a una base de datos *MySQL*, en el otro servidor, que permitiera almacenar ciertas propiedades de los eventos en tiempo real

3. **Diseñar el correlacionador de eventos de seguridad**

3.1. Se implemento un tablero en la herramienta *Power BI* para visualizar características importantes de los eventos

3.2. Se generaron eventos de ataque y eventos comunes con el fin de entrenar y probar los modelos de inteligencia artificial

3.3. Se realizó un preprocesamiento de texto en cada uno de los eventos

3.4. Se implementaron tres modelos de inteligencia artificial con la información ya preprocesada utilizando el lenguaje de programación *Python*.

4. **Analizar los resultados obtenidos**

4.1. Se analizaron los resultados de cada modelo

3. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

3.1 IDENTIFICAR Y CARACTERIZAR LOS EVENTOS A CAPTURAR

Se realizó una investigación con el fin de identificar cuáles ataques son los más frecuentes de encontrar en los eventos de un servidor web. Se encontró que en las principales fundaciones relacionadas con ciberseguridad tienen como ataques más comunes los siguientes:

1) *Injection*
2) *Broken authentication*
3) *Sensitive data exposure*
4) *XML External entities*
5) *Broken Access control*
6) *Security misconfiguration*
7) *Cross-Site Scripting XSS*
8) *Insecure deserialization*
9) *Using components with known vulnerabilities*
10) *Insufficient loggin and monitoring*

Tomado de (OWASP, 2017)

Sin embargo, realizamos un estudio sobre estos para conocer cuáles serían los más factibles de replicar en un ambiente controlado mediante *scripts* automatizados. Los eventos seleccionados fueron los siguientes: *Command Injection, CSV Injection, HTML Injection, SQL Injection, XML Injection y XPATH Injection.*

3.2 DEFINIR EL MÉTODO DE MONITOREO

Se implementó y desplegó un servidor web utilizando *Apache HTTP Server*, el cual fue expuesto a internet mediante una IP púbica. *Apache*, automáticamente genera y guarda todas las peticiones del cliente como eventos de acceso en una ubicación personalizable por el usuario. Una vez hecho esto, se implementó el módulo de *Apache* llamado *mod_log_config*, el cual sirve como intermediario y permite personalizar y reenviar los eventos de acceso a un archivo específico o a un programa externo.

Se implementó y desplegó otro servidor web, al cual se le instaló el gestor de bases de datos relacionales *MySQL*. Se diseñó la siguiente tabla para almacenar los eventos:

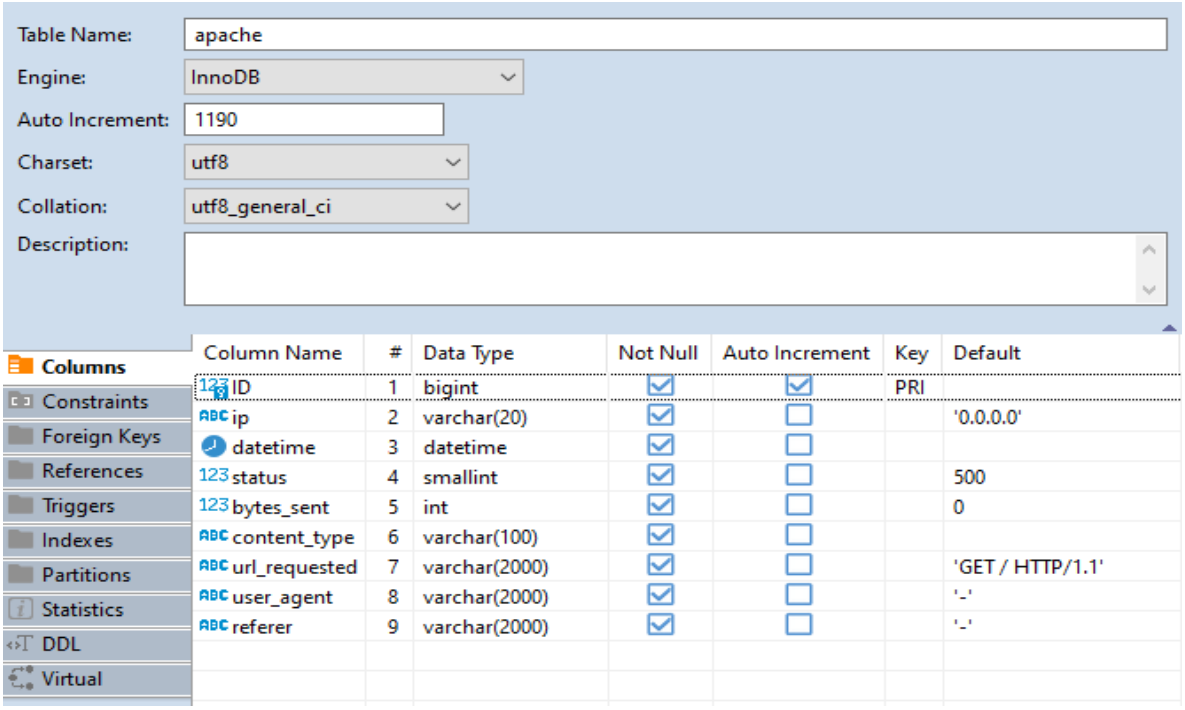

Figura 1 Tabla de MySQL de los datos de los logs

Se seleccionaron las siguientes propiedades de los eventos de acceso: *ip*, *datetime*, *status*, *bytes_sent*, *content_type*, *url_requested*, *user_agent* y *referer*. Se personalizó el módulo mod_log_config, para generar un objeto con las propiedades anteriormente descritas y comunicarse con el servidor web con **MySQL** para insertar el objeto en la tabla también mencionada.

Con lo anterior descrito, se logró almacenar la información más importante de los eventos en tiempo real y poner dicha información disponible mediante consultas *SQL***.**

3.3 DISEÑAR EL CORRELACIONADOR DE EVENTOS DE SEGURIDAD

Para definir un correlacionador de eventos de seguridad quisimos definir dos cosas: tener acceso a la información más relevante del servidor mediante un tablero visual con gráficos e información de interés, como también una manera de caracterizar un evento como un ataque específico mediante propiedades e información de este.

Para visualizar la información, utilizamos el servicio de *Microsoft* llamado *Power BI*, el cual permite generar un tablero con visualización de información interactiva y analítica. Para lograr esto, conectamos a la tabla de *MySQL* antes mencionada y generamos reportes y gráficas de interés.

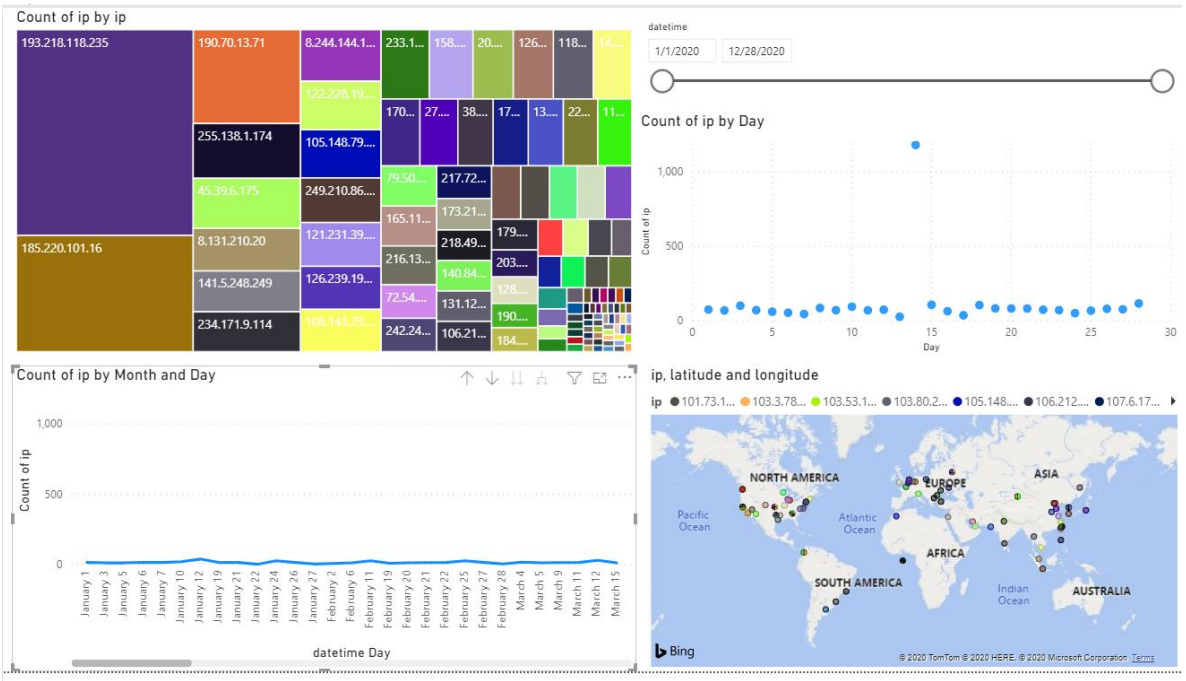

Figura 2 Tablero de PowerBI con la información recolectada

Por otro lado, para clasificar un evento como una petición común y corriente o como un ataque específico, comenzamos generando un flujo de trabajo visual utilizando la herramienta de machine learning, data mining y visualización de información, *Orange Data Minining*. Con el cual logramos entrenar y probar varios modelos de inteligencia artificial para su posterior análisis.

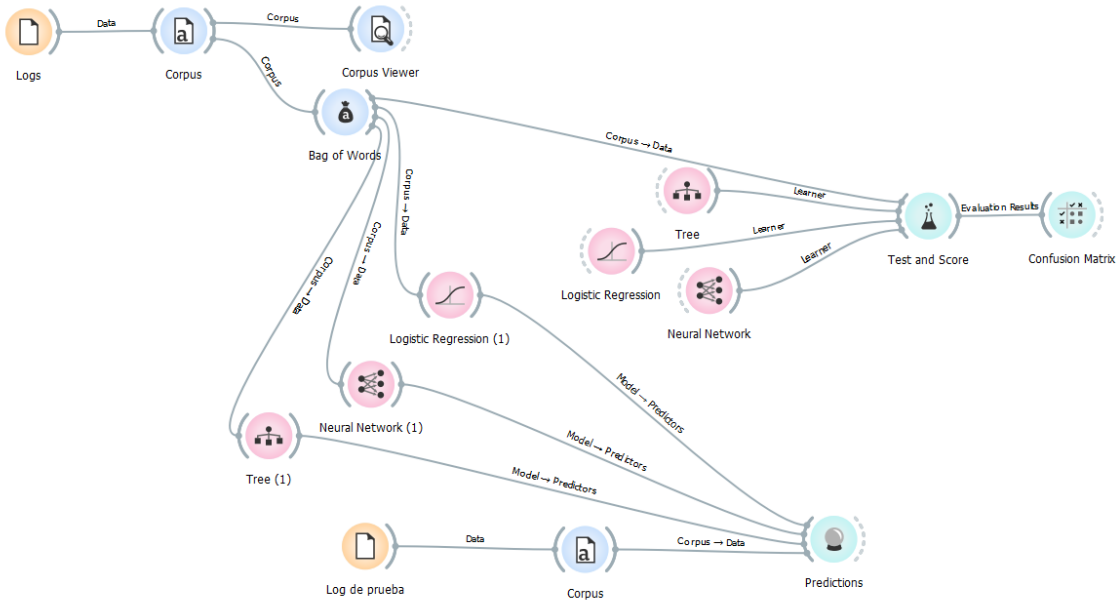

Figura 3 Flujo de trabajo en Orange Data Mining

Se logró evidenciar entonces, que los tres modelos de clasificación lograron un buen rendimiento a la hora de interpretar un registro o log como ataque.

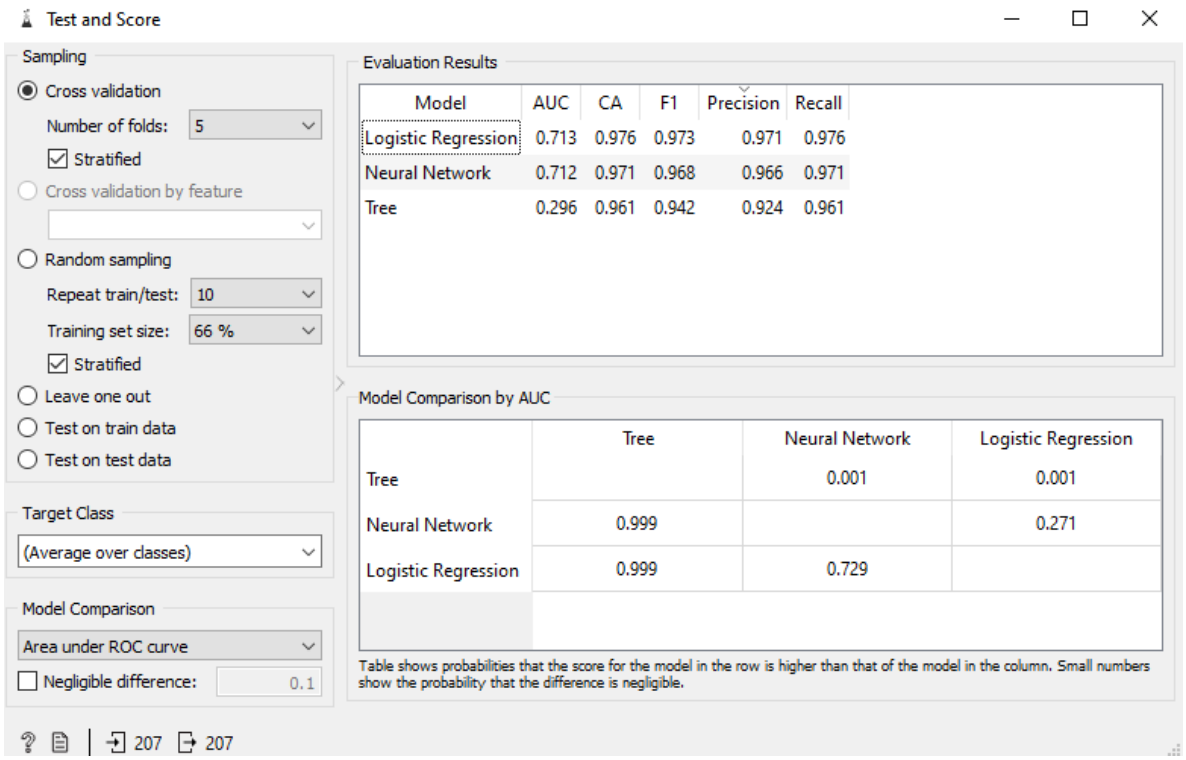

Figura 4 Resultados y puntuación de los modelos en Orange Data Mining

Para la implementación en código utilizamos el lenguaje de programación *Python*, y una librería de machine learning para el mismo lenguaje, llamada *scikit-learn*.

Se tomó un muestreo aleatorio de 1000 registros para cada tipo de petición o ataque; es decir, 1000 registros que representaran una petición normal o común y corriente, y 1000 registros que dieran lugar a un ataque específico de los mencionados anteriormente (*Command Injection, CSV Injection, HTML Injection, SQL Injection, XML Injection y XPATH Injection)*. Dichos registros fueron generados utilizando información real que llegaba al servidor y con ayuda de varias herramientas que permitieron reproducir ataques de estos tipos.

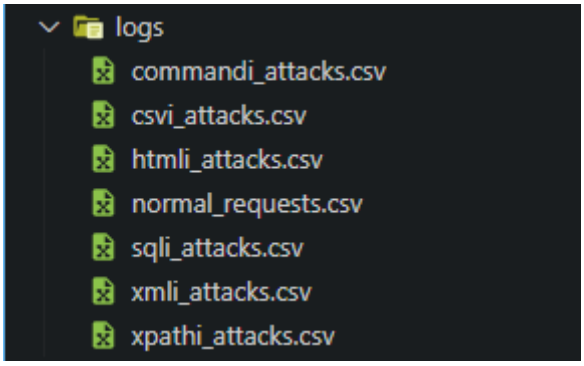

Figura 5 Árbol de archivos con los logs

Para empezar con la clasificación, se mezcló arbitrariamente la información y se dividió en un 80% para entrenamiento y un 20% para la realización de pruebas en los modelos.

```
1  X_train, X_test, y_train, y_test = train_test_split(dataset, labels, test_size=0.2, random_state=4)
```

Figura 6 Código de mezcla y división de datos

Para generar la entrada de los modelos se concatenaron las tres propiedades que podrían contener código malicioso: *url_requested*, *user_agent* y *referer*. Luego se procedió a preprocesar esta cadena de texto para generar un *bag of words* o *bolsa de palabras*, con el propósito de extraer características de interés y utilizarlas como entrada del modelo de inteligencia artificial, para esto se utilizó el módulo *CountVectorizer* de *scikit-learn* con la siguiente configuración.

```
1  CountVectorizer(binary=True, token_pattern=
   '[a-zA-Z0-9$&+,:;=?@#|<>.^*()%!\-\/"]|[a-zA-Z]+|[0-9]+|[$&+,:;=?@#|<>.^*()%!\-\/"'‘’“”']+')
```

Figura 7 Código del modelo de bag of words

Como se aprecia en la figura, la expresión regular del token encuentra y guarda la información en orden, empezando por las mixturas entre palabras, números y ciertos símbolos, de modo tal que expresiones del estilo *<html>* o *<script>document.vulnerable=true;</script>* tengan un mayor peso para el modelo, después trata de buscar solo palabras, solo números y por último todo lo que sea considerado como un símbolo. Luego, el parámetro *binary*, hace que el modelo cuente las apariciones discretamente, se probó con los valores *true* y *false*, es decir, con apariciones discretas y continúas, encontrando un mejor rendimiento con las discretas.

Una vez preprocesado el texto, se implementaron tres modelos de inteligencia artificial: árboles de

decisión, neuronas neuronales y regresión logística.

3.4 ANALIZAR LOS RESULTADOS OBTENIDOS

Una vez entrenados los modelos de clasificación, se procedieron a realizar pruebas rápidas e individuales para generar una puntuación con cada uno. Nótese que para cada modelo se utilizaron los parámetros por defecto.

```
tree = DecisionTreeClassifier(random_state=4).fit(X_train_arr, y_train)
tree_score_test = tree.score(X_test_arr, y_test)
print('tree score: ', tree_score_test)

mlp = MLPClassifier(random_state=1).fit(X_train_arr, y_train)
mlp_score_test = mlp.score(X_test_arr, y_test)
print('multi-layer perceptron score: ', mlp_score_test)

log = LogisticRegression(random_state=4).fit(X_train_arr, y_train)
log_score_test = log.score(X_test_arr, y_test)
print('logistic regression score: ', log_score_test)
```

Figura 8 Código de los modelos con parámetros por defecto

```
tree score:  0.9419488597097443
multi-layer perceptron score:  0.9447131997235666
logistic regression score:  0.9378023496890118
```

Figura 9 Puntajes de los modelos con parámetros por defecto

Como se aprecia en la figura, los tres modelos de clasificación lograron un puntaje muy bueno, por encima del 90%, pero sin llegar al 100% porque desembocaría en un sobreentrenamiento. Una vez hecho esto, se comenzó a realizar una validación cruzada de cada uno de los modelos, buscando los mejores parámetros y con esto, las mejores puntuaciones.

```python
depths = np.arange(1, 21)
num_leafs = [1, 5, 10, 20, 50, 100]
param_grid = [ { 'max_depth': depths, 'min_samples_leaf': num_leafs } ]
tree = DecisionTreeClassifier(random_state=4)
tree = GridSearchCV(tree, param_grid, cv=10)
tree = tree.fit(X_arr, y)
print("tree best score: ", tree.best_score_)
print("tree best params: ", tree.best_params_)

param_grid = [ { "hidden_layer_sizes": [80, 100, 120] } ]
mlp = MLPClassifier(random_state=1)
mlp = GridSearchCV(mlp, param_grid, cv=10)
mlp = mlp.fit(X_arr, y)
print("multi-layer perceptron best score: ", mlp.best_score_)
print("multi-layer perceptron best params: ", mlp.best_params_)

param_grid = [ { "C": [0.001, 0.01, 0.1, 1, 10, 100, 1000] } ]
log = LogisticRegression(random_state=4)
log = GridSearchCV(log, param_grid, cv=10)
log = log.fit(X_arr, y)
print("logistic regression best score: ", log.best_score_)
print("logistic regression best params: ", log.best_params_)
```

Figura 10 Código de los modelos con validación cruzada

```
tree best score:  0.946358978473671
tree best params:  {'max_depth': 15, 'min_samples_leaf': 1}
multi-layer perceptron best score:  0.9485704515409245
multi-layer perceptron best params:  {'hidden_layer_sizes': 120}
logistic regression best score:  0.9366793898963038
logistic regression best params:  {'C': 10}
```

Figura 11 Mejores puntajes de los modelos y mejores parámetros utilizando validación cruzada

Como se observa en los puntajes de la validación cruzada, tanto el árbol de decisiones, como el perceptrón multi capa obtuvieron una ligera mejora en los resultados dados los parámetros generados por la validación cruzada, aunque la regresión logística bajó levemente su puntuación, suponemos que por la manera en la que la librería utilizada divide los datos de entrenamiento de los datos de pruebas. Sin embargo, no se evidencia una diferencia notable entre un método y otro, concluyendo así que los parámetros por defecto de los tres modelos funcionan bastante bien para el caso estudiando en el presente trabajo.

Para demostrar el buen rendimiento de los modelos utilizados de una manera más gráfica se diseñaron matrices de confusión, las cuales pueden observarse a continuación.

Árbol de decisión	Petición Normal	SQL Injection	XML Injection	HTML Injection	XPath Injection	CSV Injection	Command Injection
Petición Normal	948	0	0	0	38	0	13
SQL Injection	0	999	0	0	0	0	0
XML Injection	0	0	1062	0	0	0	0
HTML Injection	0	0	8	1086	4	0	0
XPath Injection	41	0	0	2	998	0	36
CSV Injection	0	0	0	0	0	999	0
Command Injection	46	0	0	2	102	0	849

Figura 12 Matriz de confusión para el árbol de decisión

Perceptrón multicapa	Petición Normal	SQL Injection	XML Injection	HTML Injection	XPath Injection	CSV Injection	Command Injection
Petición Normal	961	0	0	0	24	0	14
SQL Injection	0	999	0	0	0	0	0
XML Injection	0	0	1062	0	0	0	0
HTML Injection	0	0	10	1082	6	0	0
XPath Injection	43	0	0	0	970	0	64
CSV Injection	0	0	0	0	0	999	0
Command Injection	46	0	0	0	69	0	884

Figura 13 Matriz de confusión para el perceptrón multicapa

Regresión logística	Petición Normal	SQL Injection	XML Injection	HTML Injection	XPath Injection	CSV Injection	Command Injection
Petición Normal	929	0	0	0	37	0	33
SQL Injection	0	997	0	0	1	0	1
XML Injection	0	0	1056	6	0	0	0
HTML Injection	0	0	18	1067	6	0	7
XPath Injection	51	0	0	0	946	0	80
CSV Injection	0	0	0	0	0	999	0
Command Injection	95	0	0	3	80	0	821

Figura 14 Matriz de confusión para la regresión logística

Por último, se realizó una predicción utilizando los tres modelos, empleando el siguiente evento como entrada:

Tabla 3: *Evento de seguridad de prueba*

ip	datatime	status	bytes_sent	content_type	url_request	user_agent	referer
242.241.239.247	2020-11-15 17:10:53	302	7313	text/html	"GET /vulnerabilities/exec?input=;|/usr/bin/id|	"Mozilla/5.0 (iPhone; CPU iPhone OS 13_4_1 like Mac	"https://www.cyberseg.tk"

					HTTP/1.1"	OS X) AppleWebKit/605.1.15 (KHTML. like Gecko) Version/13.1 Mobile/15E148 Safari/604.1"	

Nótese que el evento contiene un ataque de *Command Injection* justo en el campo de *url_request*.

La clasificación entregada por cada uno de los modelos fue la siguiente:

```
Tree prediction:  ['Command Injection']
Multi-layer perceptron prediction:  ['Command Injection']
Logistic regression prediction:  ['Command Injection']
```

Figura 15 Resultados de la predicción de cada modelo para el evento de seguridad de prueba

Como se ilustra en la figura, el evento fue clasificado como el ataque *Command Injection* esperado, validando de esta manera la calidad de los tres modelos entrenados.

4. CONCLUSIONES Y CONSIDERACIONES FINALES

Principalmente, se demostró la viabilidad de clasificar un evento como malicioso utilizando modelos de inteligencia artificial, siempre y cuando se procesen de la manera correcta los datos de entrada para dichos modelos. Si es posible, se recomienda utilizar expresiones regulares específicas para generar los diferentes *tokens* de cada ataque, pues la información varía de una petición a otra y puede terminar por contaminar los datos de entrada para el modelo de clasificación.

Resulta interesante la posibilidad que ofrece *Apache HTTP Server*, junto a su módulo *mod_log_config*, de reenviar en tiempo real los eventos que generalmente se guardan en un archivo, a cualquier otro servicio que se desee —En este caso *MySQL*—, por medio de pequeñas modificaciones, sin necesidad de incluir ningún otro *script* que monitoree la información.

Queda abierta la oportunidad de integrar el clasificador de eventos, bien sea a la herramienta *Power BI*, o a cualquier otra plataforma donde se pueda obtener información de interés de los eventos generalizados, como también al mismo tiempo clasificar los eventos en tiempo real. Además de la implementación de un modelo de predicción, con el que el sistema podría determinar futuros ataques e informar a la organización del riesgo cibernético próximo, con el fin de mitigar el peligro según sea el caso.

5. REFERENCIAS

Abad, C., Taylor, J., Sengul, C., Yurcik, W., Zhou, Y., & Rowe, K. (2003). Log correlation for intrusion detection: A proof of concept. *Proceedings - Annual Computer Security Applications Conference, ACSAC, 2003-Janua*(Acsac), 255–264. https://doi.org/10.1109/CSAC.2003.1254330

Ambre, A., & Shekokar, N. (2015). Insider threat detection using log analysis and event correlation. *Procedia Computer Science*, *45*(C), 436–445. https://doi.org/10.1016/j.procs.2015.03.175

Bonatti, P., Dullaert, W., Fernandez, J. D., Kirrane, S., Milosevic, U., & Polleres, A. (2019). *The SPECIAL Policy Log Vocabulary*. Institute for Information Business.

BSA-The Software Aliance. (2018). 2018 BSA GLOBAL CLOUD COMPUTING SCORECARD. *Business Software Alliance*, 26.

Cano, J. A., & Baena, J. J. (2015). Trends in the use of information and communication technologies for international negotiation. *Estudios Gerenciales*, *31*(136), 335–346. https://doi.org/10.1016/j.estger.2015.03.003

Cloudflare. (2020). *What Is a Brute Force Attack?* Cloudflare.

Eduardo Díaz Rodríguez, H. (2017). Tecnologías de la información y comunicación y crecimiento económico. *Economía Informa*, *405*, 30–45. https://doi.org/10.1016/j.ecin.2017.07.002

ESDS. (2010). *Difference Between Server and Desktop Computer*. ESDS.

Ficco, M., & Romano, L. (2010). A correlation approach to intrusion detection. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, *45 LNICST*(September 2014), 203–215. https://doi.org/10.1007/978-3-

642-16644-0_19

Kaspersky. (2018). *CYBERTHREAT REAL-TIME MAP*. Kaspersky Lab.

Lavrova, D., Pechenkin, A., & Gluhov, V. (2015). Applying correlation analysis methods to control flow violation detection in the internet of things. *Automatic Control and Computer Sciences*, *49*(8), 735–740. https://doi.org/10.3103/S0146411615080283

Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, *58*(4), 431–440. https://doi.org/10.1016/j.bushor.2015.03.008

Meera, G., & Geethakumari, G. (2016). Event Correlation for Log Analysis in the Cloud. *Proceedings - 6th International Advanced Computing Conference, IACC 2016*, 158–162. https://doi.org/10.1109/IACC.2016.38

Moiseenko, I., Stapp, M., & Oran, D. (2014). Communication patterns for web interaction in Named Data Networking. *ICN 2014 - Proceedings of the 1st International Conference on Information-Centric Networking*, 87–95. https://doi.org/10.1145/2660129.2660152

Nelson, L. E., O'Donnell, G., Caldwell, J., & Lynch, D. (2018). *Adoption Profile: Public Cloud In Europe, Q2 2018*.

Oluwatosin, H. S. (2014). Client-Server Model. *IOSR Journal of Computer Engineering*, *16*(1), 57–71. https://doi.org/10.9790/0661-16195771

OWASP. (2017). *OWASP Top Ten Web Application Security Risks | OWASP*. https://owasp.org/www-project-top-ten/

Pérez Pérez, Y. (2016). Importancia De La Ciberseguridad En Colombia. *Universidad Piloto de Colombia*, 9.

Sobers, R. (2020). *110 Must-Know Cybersecurity Statistics for 2020*. Varonis.

Statista. (2020). *Number of low-power wide-area network (LPWAN) connected devices worldwide from 2016 to 2021*. Statista.

Tanenbaum, A. S. (2003). *Redes de computadoras* (G. Trujano Mendoza (ed.); 4ta ed.). Pearson Education Inc.

Vaarandi, R., Blumbergs, B., & Çallškan, E. (2015). Simple event correlator - Best practices for creating scalable configurations. *2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision, CogSIMA 2015*, 96–100. https://doi.org/10.1109/COGSIMA.2015.7108181

Valenzuela, A. (2018). El Internet de las Cosas causa un 63% más de ciberataques en las empresas - elEconomista.es. In *El Economista*.

Yusof, R., Selamat, S. R., & Sahib, S. (2008). Intrusion Alert Correlation Technique Analysis for Heterogeneous Log. *IJCSNS International Journal of Computer Science and Network Security*, *8*(9), 132–138.