UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

EVALUACIÓN DE RENDIMIENTO DE ALGORITMOS EN LA IDENTIFICACIÓN DE ATAQUES A SITIOS WEB UTILIZANDO LOGS DE SERVIDOR

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor(a) (es):

Bach. Chinguel Tineo Segundo Florentino

ORCID: https://orcid.org/0000-0002-9961-4744

Asesor(a):

Mg. Samillan Ayala Alberto Enrique

ORCID: https://orcid.org/0000-0002-0071-4367

Línea de Investigación:

Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú 2022

APROBACIÓN DEL JURADO

EVALUACIÓN DE RENDIMIENTO DE ALGORITMOS EN LA IDENTIFICACIÓN DE ATAQUES A SITIOS WEB UTILIZANDO LOGS DE SERVIDOR

_____

Bach. Chinguel Tineo Segundo Florentino
Autor

_____

Mg. Samillan Ayala Alberto Enrique
Asesor

_____ _____

Mg. Bances Saavedra David Enrique
Presidente de Jurado

Mg. Bravo Ruiz Jaime Arturo
Secretario de Jurado

_____

Mg. Tuesta Monteza Victor Alexci
Vocal de Jurado

Dedicatorias

Esta investigación la dedico a toda mi familia, quienes de muchas formas han contribuido a que pueda sacarlo adelante. Me refiero a mi familia en conjunto y no resaltaré nombres, pues al ser mi familia numerosa para mencionar a todos, y considerando que sería quitarle valor al apoyo recibido por quienes podría olvidar de mencionar.

Agradecimientos

Mi agradecimiento por haber logrado culminar exitosamente esta investigación es muy extenso. He recibido apoyo constante y muy valioso por distintas personas, iniciando desde mis docentes de Investigación I y II, docente de taller de investigación, docentes de la escuela de ingeniería de sistemas, compañeros y colegas de la escuela de ingeniería de sistemas, mi pareja, mis hermanos y padres.

Resumen

El 65.6% de la población del mundo tiene acceso y uso activo a internet (Internetworldstats, 2021). Existe un aproximado de 1,2 billones de sitios web activos (Netcraft, 2021) y cada uno almacena las solicitudes recibidas en un archivo log. A nivel de aplicaciones web, se conoce que los ataques de Broken Access control (top 1) e Inyección (top 3) según la clasificación realizada por OWASP en su Top 10 Web Application Security Risks publicada en 2021. Basado en estos datos estadísticos se propuso el proyecto de investigación denominado “Evaluación de rendimiento de algoritmos en la identificación de ataques a sitios web utilizando logs de servidor.” Debido a que los archivos log consisten en una fuente que está obteniendo mucha relevancia en la identificación de posibles ataques a sitios web pero que no estaba siendo aprovechada. Se hizo una selección de ataques a investigar basados en el top 10 liberado por OWASP en su informe de 2021, donde las vulnerabilidades de Broken Access Control, Injection y Cross Site Scripting que se encuentran dentro del top 3 de vulnerabilidades fueron seleccionadas. Se analizó diversas bases de datos de vulnerabilidades de seguridad conocidas y reglas de seguridad recabadas del proyecto web PHP para la identificación de intrusos (PHPIDS), lo que permitió elaborar un total de 809 patrones de ataques, los cuales fueron procesados para permitir que sean clasificados por diversos algoritmos de aprendizaje automático. Basado en una decena de investigaciones relacionadas a la presente investigación, se seleccionó a los algoritmos que mejor resultado de desempeño ofrecieron en su respectiva investigación y de los cuales se eligió a 3 que fueron BayesNet, Random Forest y Support Vector Machine (SVM). Haciendo uso de WEKA, una aplicación utilizada para pruebas de minería de datos y aprendizaje automático, se realizó la evaluación del rendimiento de los algoritmos en la clasificación de los 809 patrones de ataques mediante tres pruebas distintas que fueron Percentage Split, Cross Validation y Training Set. Los resultados de la evaluación concluyen que el algoritmo Random Forest obtuvo un rendimiento optimo, con una exactitud de 100% en la prueba de Training Set, el algoritmo SVM obtuvo un 99.9% y el algoritmo Bayes Net obtuvo un 98.9%.

Palabras Clave: Inyeccion SQL, XSS, Path Traversal, Random Forest, Log Files

Abstract

65.6% of the world's population has access to and active use of the Internet (Internetworldstats, 2021). There are approximately 1.2 billion active websites (Netcraft, 2021) and each one stores the requests received in a log file. At the web application level, Broken Access control (top 1) and Injection (top 3) attacks are known according to the classification made by OWASP in its Top 10 Web Application Security Risks published in 2021. Based on these statistical data, it was proposed the research project called "Performance evaluation of algorithms in the identification of attacks on websites using server logs." Because the log files consist of a source that is getting a lot of relevance in the identification of possible attacks on websites but that was not being taken advantage of. A selection of attacks to investigate was made based on the top 10 released by OWASP in its 2021 report, where the Broken Access Control, Injection and Cross Site Scripting vulnerabilities found within the top 3 vulnerabilities were selected. Various databases of known security vulnerabilities and security rules collected from the PHP web project for intrusion identification (PHPIDS) were analyzed, which allowed the elaboration of a total of 809 attack patterns, which were processed to allow them to be classified. by various machine learning algorithms. Based on a dozen investigations related to the present investigation, the algorithms that offered the best performance result in their respective investigation were selected and 3 were chosen, which were BayesNet, Support Vector Machine (SVM) and Random Forest. Using WEKA, an application used for data mining and machine learning tests, the performance evaluation of the algorithms in the classification of the 809 attack patterns was carried out through three different tests that were Percentage Split, Cross Validation and Training Set. The results of the evaluation conclude that Random Forest algorithm obtained optimal performance, with an accuracy of 100% in the Training Set test, the SVM algorithm obtained 99.9% and the Bayes Net algorithm obtained 98.9%.

Keywords: SQL Injection, XSS, Path Traversal, Random Forest, Log Files

Índice

I. INTRODUCCIÓN

1.1. Realidad Problemática

El rápido crecimiento de las tecnologías de la información (TI) ha hecho que la divulgación de información se vuelva crítica, y la web juega un papel muy importante pues se visualiza como el principal medio de difusión de información.

Según Netcraft, hasta diciembre de 2021, existe un aproximado de 1.2 billones de sitios web activos y si comparamos con la población mundial que a la misma fecha es de 7.8 billones según el Banco Mundial, tendríamos un aproximado de 1 sitio web por cada 7 personas.

Los sitios web se encuentran alojados en computadoras con características superiores a la de una computadora de uso personal a las cuales se les denomina servidores web. Estos servidores son habilitados y gestionados en su gran mayoría por empresas dedicadas a brindar este servicio, las cuales cuentan con estándares y protocolos de seguridad cuya finalidad es proteger la confidencialidad de la información que generan los visitantes de los sitios web.

Según Netcraft, hasta diciembre de 2021, Apache es el servidor web más utilizado pues representa el 24%, seguido de Ngnix (20%), Google (10%) y Cloudflare (9%). Esta investigación también indica que el 26% de sitios web activos son servidos por proveedores pequeños categorizados como “otros”.

Tanto para sitios web cuyos proveedores los tengan bajo Apache, Ngnix u otras tecnologías populares, como para proveedores que empleen tecnologías emergentes o de menor popularidad brindando el servicio de servidores web, es muy importante tener siempre un control recurrente de la seguridad de sus servidores, debido a la existencia de personas que por diversos motivos pueden estar en todo momento realizando pruebas de seguridad en los sitios web y servidores que los alojan para lograr identificar vulnerabilidades que puedan explotar y así romper la seguridad del servidor y robar su información.

Los servidores cuentan con un registro automático de cada consulta realizada a un sitio web, este registro contiene información del visitante que, al ser analizada

minuciosamente, puede brindar pistas para identificar comportamientos sospechosos o ataques para prevenir en la posteridad o incluso poder dar con el paradero del atacante. Estos registros del servidor se conocen como Archivos Log o Archivos de Registro.

Este documento presentó el proceso de análisis de archivos log de un servidor web Apache, que posee más sitios webs activos. Utilizando su tecnología y cuyo formato de registro de la actividad de un visitante es diferente al de un servidor Nginx u otro popular.

Un atacante puede ser una persona o un software diseñado para atacar servidores, por ello para generalizar se les denominará “agente de ataques”.

Los agentes de ataques utilizan diversos métodos dirigidos a aplicaciones web de los cuales se seleccionó los ataques de Broken Access control (top 1) e Inyección (top 3) según la clasificación realizada por OWASP en su Top 10 Web Application Security Risks publicada en (OWASP, 2021).

1.2. Trabajos previos

Arumugam et al., (2019), en su investigación "Prediction of SQL Injection Attacks in Web Applications" refiere que, a medida que las aplicaciones web se vuelven cada vez más complejas y conectadas, se vuelve imperativo reducir sus vulnerabilidades. La investigación se enfocó en los Ataques de SQL Inyección (SQLIA) y se propuso un sistema cuyo objetivo fue predecir la ocurrencia de SQLIA en un servidor determinado, con aplicaciones implementadas en él y con la ayuda de la herramienta Apache JMeter para simular archivos LOG. Por lo tanto, el modelo se construyó y entrenó de acuerdo con las palabras clave encontradas en el parámetro REQUEST aplicando la regresión logística que consiste en encontrar una relación entre las características y la probabilidad de un resultado particular, donde el evento ocurre (1) o el evento no ocurre (0). De un conjunto de datos de aproximadamente 1,00,000, el 70% se usó para entrenamiento y el resto para pruebas. El modelo ofreció una precisión del 72% para predecir la inyección de SQL.

AZIZI et al., (2019), en su investigación "Log files Analysis Using MapReduce to Improve Security", refiere que los archivos de registro representan una fuente de

información muy valiosa para diagnosticar la seguridad del sistema y detectar problemas que ocurren en el mismo, estos archivos de registro suelen ser son muy grandes y pueden tener una estructura compleja. Por ello, proporcionaron una metodología de análisis de seguridad que tiene como objetivo aplicar técnicas de Big Data, como MapReduce, sobre los archivos de registro del sistema para localizar y extraer datos probablemente relacionados con ataques realizados por usuarios malintencionados que pretenden comprometer el sistema. Estos datos conducirán, mediante un proceso de aprendizaje, a identificar, predecir ataques o detectar intrusiones de tipo SQLI y DDOS mediante el análisis de archivos de registro de acceso de servidores web apache. Los resultados obtenidos fueron capaces de extraer indicadores y eventos maliciosos que, de acuerdo a los indicadores ingresados previamente, lograron ser clasificados como SQLI o DDOS.

Zhang (2019), realizó la investigación, A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities, en Estados Unidos refiere que, Con el creciente número de aplicaciones web en los últimos años, La inyección SQL (SQLI) ha aparecido constantemente en la lista de los 10 principales riesgos de seguridad del Open Web Application Security Project (OWASP). Por lo tanto, los esfuerzos de investigación para identificar y prevenir SQLI son cruciales. El objetivo principal de la presente investigación fue desarrollar un clasificador basado en aprendizaje automático (ML) para identificar archivos que contienen vulnerabilidades SQLI. Además, los enfoques basados en el aprendizaje profundo recientemente han mostrado resultados mejorados en las tareas de clasificación del código fuente, como la predicción de errores, la localización de errores y la identificación de clonación de código. Por lo tanto, el objetivo secundario de la presente investigación es evaluar la eficacia de los enfoques basados en el aprendizaje profundo sobre los algoritmos clásicos de aprendizaje automático para identificar SQLI en código PHP. Se utilizaron algoritmos de aprendizaje automático clásicos y basados en aprendizaje profundo para entrenar y evaluar modelos de clasificadores utilizando funciones de validación de entrada y desinfección extraídas de archivos de código fuente. En validaciones cruzadas de diez veces, un modelo entrenado con SVM (98.6%), seguido de Regresión Logística (98.5%) y la red neuronal convolucional (CNN) con (95,4%).

Latib et al., (2018), refiere en su investigación "Analysing Log Files For Web Intrusion Investigation Using Hadoop", que el proceso de analizar una gran cantidad de datos del archivo de registro ayuda a la organización a identificar la actividad de los intrusos, así como las vulnerabilidades del sitio web. Sin embargo, analizarlos es un gran desafío, ya que el proceso requiere mucho tiempo y, a veces, puede resultar ineficaz pues es posible que los analizadores de registros tradicionales no puedan analizar una gran cantidad de datos. Por tanto, el objetivo para esta investigación fue analizar en función de los ataques que se captaron haciendo uso de los archivos de registro del servidor web. Se limpió y perfeccionó los archivos de registro, mediante un programa de preprocesamiento, se realizó una simulación experimental utilizando el entorno de Hadoop para producir los resultados de análisis requeridos. Los resultados de esta simulación experimental indican que la aplicación Hadoop es capaz de producir resultados de análisis a partir de archivos de registro web de gran tamaño para ayudar en la investigación de intrusiones web. Además de eso, el análisis de rendimiento del tiempo de ejecución muestra que el tiempo total de ejecución no aumentará linealmente con el tamaño de los datos. Este estudio también proporciona una solución para visualizar el resultado del análisis mediante Power View y Hive.

Gao et al., (2017), refiere en su investigación "Anomaly Detection of Malicious Users' Behaviors for Web Applications Based on Web Logs", que con más y más servicios en línea desarrollados en aplicaciones web, problemas de seguridad basados en aplicaciones web se están volviendo más serios. Generalmente los sistemas de detección de intrusiones se basan en cada solicitud para encontrar el ciberataque, en lugar de los comportamientos de los usuarios, y estos sistemas solo pueden proteger la aplicación web de vulnerabilidades conocidas en lugar de algunos ataques de día cero. Para detectar ataques recientemente desarrollados, analizaron los registros de los servidores web y definieron los comportamientos de los usuarios para dividirlos en normales y maliciosos. El resultado mostró que al utilizar la información de los archivos de registro web para definir los comportamientos de los usuarios, se puede obtener una mayor tasa de precisión y una menor tasa de falsas alarmas de detección de intrusos.

Baş Seyyar et al., (2017), en su investigación "Detection of attack-targeted scans from the Apache HTTP Server access logs", propuso un método para detectar exploraciones orientadas a ataques y para distinguirlas de otros tipos de visitas. En tal contexto, usaron archivos log de servidores web IIS y Apache e intentaron determinar las situaciones de ataque mediante el análisis y evaluación de los datos pasados. Además de las detecciones de exploración web, insertaron un conjunto de reglas para identificar ataques de inyección de SQL y XSS. Los resultados de esta investigación mostraron una probabilidad de detección muy alta y una probabilidad baja de falsa alarma, donde, la exactitud y las tasas de precisión del modelo fueron 99,38%, 100,00% respectivamente.

Cao et al., (2017), en su investigación "Machine Learning to Detect Anomalies in Web Log Analysis", refiere que, mientras la tecnología de la información se desarrolla rápidamente, los servidores web pueden ser atacados fácilmente debido a su alto valor. Por tanto, la seguridad web ha despertado una gran preocupación tanto en el ambiente académico como en la industria. La detección de anomalías juega un papel importante en el campo de la seguridad web, y los archivos de registro que guardan información detallada del tiempo de ejecución del sistema se han convertido en un objeto de análisis de datos importante en consecuencia. La detección de anomalías de registro tradicional depende de que los programadores inspeccionen manualmente realizando búsqueda de palabras clave y la coincidencia de expresiones regulares. Aunque los programadores pueden usar el sistema de detección de intrusiones para reducir su carga de trabajo, los datos del sistema de registro son enormes, los tipos de ataques son diversos y las habilidades de piratería están mejorando, lo que hace que la detección tradicional no sea lo suficientemente eficiente. En esta investigación propuso un sistema de detección de anomalías para archivos de registro web, que adopta un algoritmo de aprendizaje de dos niveles. El modelo de Decision Tree clasifica conjuntos de datos normales y anómalos. El conjunto de datos normal se verifica manualmente para el establecimiento de múltiples HMM. Después de comparar con los algoritmos de aprendizaje automático Logistic Regression, Decision Tree, y SVM, utilizados en la detección de anomalías, los resultados experimentales de este conjunto de datos mostraron al algoritmo Decision Tree con un 74.19%, HMM un 90.32%, SVM cuya

precisión es del 90,32%, este resultado es mejor y con menor tasa de falsos positivos. Finalmente, el sistema propuesto, Anomaly Detection System obtuvo una clasificación de 93.54%.

Ma et al., (2017), en su investigación "Neural Network Based Web Log Analysis for Web Intrusion Detection" refiere que, con el aumento de ataques a servidores web y aplicaciones web, es urgente desarrollar un sistema para detectar intrusiones web. Los archivos de registro web son datos de transmisión que registran el comportamiento de los clics de los usuarios durante la navegación por Internet. Al analizar cuidadosamente estos archivos de registro, podemos revelar algunas anomalías o ataques potenciales para reducir la pérdida de propiedad. Esta investigación propuso un método que aplica el algoritmo de red neuronal a la detección de intrusiones web basado en registros de acceso al servidor web. Antes de introducir los archivos de registro sin procesar en algoritmos de redes neuronales, se realizó el preprocesamiento de estos archivos de texto para que los registros procesados sean de buena calidad, con menos ruido y errores. Sus resultados demostraron que la precisión del método propuesto es superior al clasificador de Decision Tree (93%), lo que muestra que el método de red neuronal (97%) se puede aplicar a la detección de intrusiones web de manera efectiva.

Han & Phyu, (2016), en su investigación "Classification of SQL injection, XSS and Path Traversal for Web Application Attack Detection", refiere que, la detección de ataques a aplicaciones web representa una de las ramas de la investigación con más popularidad durante estos años. Los ataques de inyección SQL, XSS y Path Traversal son los tipos de ataques de aplicaciones web más comunes. El sistema propuesto clasificó eficazmente tres ataques mediante el algoritmo Random Forest para garantizar una precisión razonable. El módulo de longitud de la solicitud se calcula en función de la longitud determinada de la URL para analizar cada registro como normal o ataque. El análisis regular de patrones se enfatiza en el contenido de la URL y otras características para analizar ciertos patrones de ataque. En este sistema se utilizó el conjunto de datos de ataques web estándar ECML / PKDD. Se propuso la combinación del algoritmo Random Forest con un módulo de longitud de solicitud y expresiones regulares para patrones de ataque según el conjunto de

datos estándar ECML / PKDD. Los resultados mostraron que la precisión de Random Forest fue de 86.62%, y la precisión de la aplicación del sistema propuesto fue de 91.12%.

Moh et al., (2016), en su investigación "Detecting Web Attacks Using Multi-Stage Log Analysis", refiere que las aplicaciones basadas en la web han ganado aceptación universal en todos los ámbitos de vida, incluidas las comunidades sociales, comerciales, gubernamentales y académicas. Incluso con la reciente aparición de la tecnología en la nube, la mayoría de las aplicaciones en la nube se acceden y controlan a través de interfaces web. Por lo tanto, la seguridad web ha seguido siendo fundamentalmente importante y extremadamente desafiante. Uno de los principales problemas en la seguridad de las aplicaciones web son los ataques de inyección de SQL. La mayoría de las soluciones existentes para detectar estos ataques utilizan análisis de logs y emplean métodos de coincidencia de patrones o de aprendizaje automático. Los métodos de coincidencia de patrones pueden ser eficaces, dinámicos; sin embargo, no pueden detectar nuevos tipos de ataques. Los métodos de aprendizaje automático supervisado pueden detectar nuevos ataques, pero deben depender de una fase de entrenamiento fuera de línea. Este trabajo propuso una arquitectura de análisis de logs de múltiples etapas, que combina métodos de coincidencia de patrones y de aprendizaje automático supervisado. Utilizó los logs generados por la aplicación durante los ataques para detectarlos de forma eficaz y ayudar a prevenir futuros ataques. La arquitectura se describe en detalle; Se implementa y aloja un prototipo de prueba de concepto en Amazon AWS, utilizando Kibana para la coincidencia de patrones y Bayes Net para el aprendizaje automático. Se evaluó en 10,000 logs para detectar ataques de inyección SQL. Los resultados del experimento donde el modelo Bayes Net primero clasifica los registros y luego Kibana detecta más inyecciones de SQL según las consultas y la clasificación (realizada por el modelo de Bayes Net) mostraron una alta precisión del 95,4%.

1.3. Teorías relacionadas al tema

1.3.1. Servidor Web

El servidor es un elemento que tiene conexión con la red de transporte y que se encarga de brindar uno o varios servicios a ciertos clientes. Según la arquitectura, software, funcionalidad, ubicación u otra, puede tener características muy diversas. Esto permite establecer múltiples clasificaciones de servidores. (Stevens, Fenner, & Rudolf, 2004).

Figura 1: Servidor web y cliente web.
Fuente: (Lenguaje I, 2017)

1.3.2. URI

Se denomina URI a una secuencia de caracteres que permiten identificar a un recurso web, la URI también se puede llamar URL, y es así como se llamará en el resto de la investigación. Una URL tiene un formato como el siguiente: <scheme>://<authority><path>?<query>, Donde <scheme> identifica al protocolo de acceso, el cual puede ser http, ftp, telnet u otros. Seguido de los dos puntos (:) y después puede contener un slash (/) en casos sea una URL relativa o doble slash (//) para indicar que la URL será absoluta. Se define como URL absoluta a la dirección web que contiene el formato de una URL completa y URL relativa a la dirección que solo posee parte del esquema original, dirección que parte desde la autoridad (<authority>) o desde el directorio (<path>). La autoridad (<authority>) hace referencia a la dirección de dominio del recurso, que puede ser una dirección

de dominio identificada por un nombre (dominio.com) o una dirección IP (50.57.50.11), siempre antecedida por el esquema, los dos puntos y el slash doble.

El camino (<path>) hace referencia a la ubicación de un directorio o archivo que pertenece al dominio de autoridad. Adicionalmente una URL puede recibir parámetros que serán pueden alimentar o modificar el recurso al que se está accediendo, estos parámetros se deben especificar después del signo de interrogación (?) y se envuelven como el componente de la URL llamado consulta (<query>). La consulta tiene algunos caracteres reservados (;, /, ?, :, @, &, =, +, , y $) que serán interpretados por el servidor. Estas definiciones están basadas en la RFC 2396. (Berners-Lee, 1998).

1.3.3. Protocolo HTTP

El protocolo HTTP (HyperText Transfer Protocol, protocolo de transferencia de hipertexto) se ubica en la capa de aplicación descrita en el modelo TCP/IP según el RFC 1945 (Berners-Lee, MIT/LCS, Fielding, University of California Irvine, & Nielsen, 1996).

Romero Laguillo (1997). En su libro “Publicar en Internet: guía práctica para la creación de documentos HTML” describe al protocolo HTTP de la siguiente manera: un proceso denominado servidor se encarga de escuchar en un puerto de comunicaciones TCP que puede ser el 80, y espera que los clientes web envíen solicitudes de conexión. Luego que la conexión es establecida, TCP se asegura de mantener la comunicación y garantizar el intercambio de datos sin la presencia de errores, basado en operaciones simples de tipo solicitud/respuesta.

1.3.4. Archivo LOG

Un archivo log es representado por uno o más documentos de texto que son creados y administrados de manera independiente por un servidor, en estos archivos se registra toda la actividad que se realiza en el servidor. Al existir diversos tipos y versiones de servidores, estos pueden registrar o no determinados tipos, formatos y contenidos en sus archivos LOG.

Los archivos de LOG más importantes son dos:

El Error Log (Registro de errores del servidor), el cual se encarga de registrar la identificación de cualquier problema durante su operatividad. El Access Log (Registro de acceso), el cual registra todas las solicitudes que el servidor ha recibido y procesado. Entre los datos presentes en este documento se puede identificar la IP de origen, el navegador, la versión del protocolo HTTP, fecha de la consulta y URL consultada.

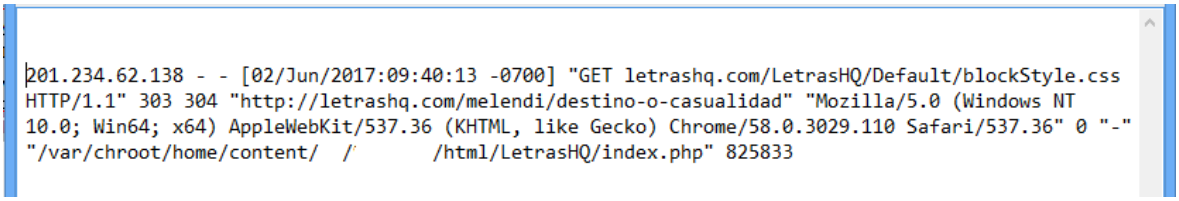


Figura 2: Registro de una solicitud en un Archivo Log
Fuente: Elaboración propia.

1.3.5. Ataques informáticos

El informe de OWASP (2021) indica que hay 10 categorías de vulnerabilidades de seguridad principales en los métodos de ataque a aplicaciones web, de los cuales se destacan las vulnerabilidades de Broken Access Control y Injection como top 1 y top 3 respectivamente. Dentro de estas categorías existen métodos que pueden ser detectados mediante el análisis de archivos LOG, de los cuales se describen los principales.

1.3.5.1. Injection

La fundación OWASP (2007) nos dice que:

Existen diversos tipos de inyección como SQL, HTML, XML, PHP, de comandos del sistema operativo, etc. La inyección se lleva a cabo porque los datos enviados por el usuario se registran como un comando o consulta, burlando al intérprete que ejecute la sentencia. Los fallos de inyección posibilitan al atacante el leer, crear, modificar o eliminar de manera arbitraria, los datos disponibles en una aplicación.

Escenario #1: la aplicación utiliza datos no confiables en la construcción del siguiente comando SQL vulnerable:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") + "'";
```

Escenario #2: la confianza total de una aplicación en su *framework* puede resultar en consultas que aún son vulnerables a inyección, por ejemplo, *Hibernate Query Language (HQL)*:

```
Query HQLQuery = session.createQuery("FROM accounts WHERE
custID='" + request.getParameter("id") + "'");
```

En ambos casos, al atacante puede modificar el parámetro *"id"* en su navegador para enviar: ' or '1'='1. Por ejemplo:

```
http://example.com/app/accountView?id=' or '1'='1
```

Esto cambia el significado de ambas consultas, devolviendo todos los registros de la tabla *"accounts"*. Ataques más peligrosos podrían modificar los datos o incluso invocar procedimientos almacenados.

Figura 3: Ejemplos de ataques de inyección.
Fuente: Elaboración propia.

1.3.5.2. Cross Site Scripting

La fundación OWASP (2007) nos dice que:

Cross Site Scripting (XSS), representa la amenaza de seguridad de aplicaciones web más frecuente. Los fallos de XSS ocurren cuando una aplicación vulnerable recopila datos de un usuario y los transmite a un navegador web sin haber realizado una validación y/o codificación. La ejecución exitosa de un ataque de XSS puede permitirle secuestrar las sesiones, modificar sitios web, posibilitar la realización de ataques de phishing o instalación de malware de secuencias de comandos.

Ejemplo de escenario de ataque (Cross site cripting XSS)

La aplicación utiliza datos no confiables en la construcción del siguiente fragmento de HTML sin validación o escape:

(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";

El atacante modifica el parámetro 'CC' en su navegador para:

'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.

Este ataque hace que el ID de sesión de la víctima sea enviado al sitio web del atacante, lo que permite al atacante secuestrar la sesión actual del usuario.

Tenga en cuenta que los atacantes también pueden usar XSS para derrotar cualquier defensa CSRF automatizada que pueda utilizar la aplicación. Ver 2017-A8 para información sobre CSRF.

Figura 4: Ejemplo de Cross Site Scripting.
Fuente: Elaboración propia.

1.3.5.3. Path Traversal o Directory Traversal

Un ataque muy común es el denominado Directory traversal que consiste en acceder a cualquier página que no esté disponible al público pero que no cuenta con algún nivel de protección que prohíba su visualización como en el ejemplo mostrado en el escenario 2 de la siguiente figura.

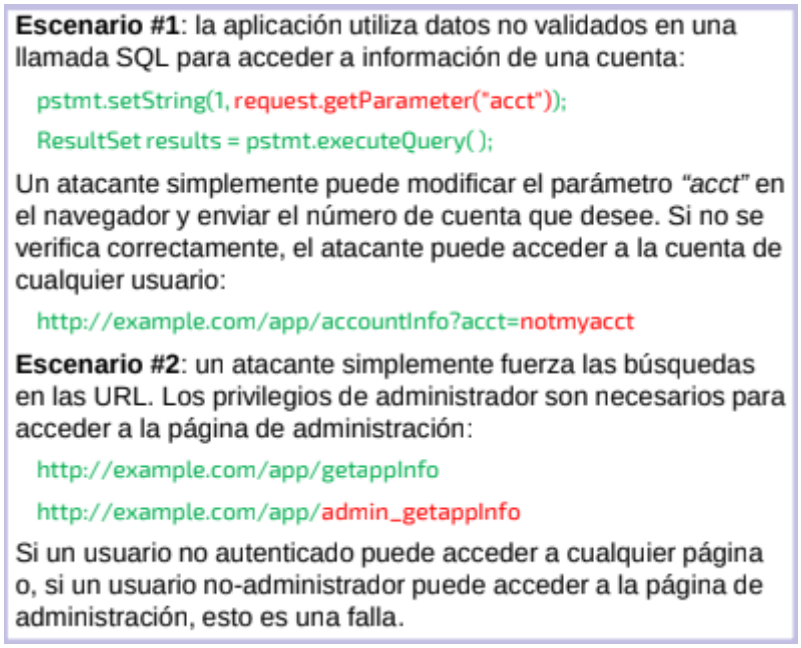


Figura 5: Ejemplo de ataque de Broken Access Control sobre datos no validados y sobre Path Traversal.
Fuente: (OWASP, 2017)

1.3.6. Algoritmos de aprendizaje supervisado

1.3.6.1. Bayes Net

El algoritmo Bayes Net (BN) está basado en el teorema de Bayes. Lo que significa que la probabilidad condicional de cada nodo debe ser calculada para formar una red bayesiana, la cual se representa como un grafo acíclico dirigido. En Bayes Net se debe asumir que todos sus atributos son nominales y no hacen faltan valores. Para estimar la probabilidad condicional se hace uso de diversos algoritmos como, K2, la búsqueda genética y Hill Climbing (V.Vaithiyanathan, K. Rajeswari, Kapil Tajane, Rahul Pitale., 2013)

Las Redes Bayesianas se utilizan con frecuencia como un clasificador que permita predecir la probabilidad de una etiqueta perteneciente a la clase de destino según características dadas. (T. Roos, H. Wettig, P. Gr¨unwald, P. Myllym¨aki, H. Tirri., 2005)

Notación Matemática

Un modelo Bayes Net "B" para un set "n" de variables "X={□1, □2, … , □□}", cada uno tiene un conjunto finito de estados mutuamente excluyentes consta de dos componentes principales, B=(G,Q) donde "G" es un grafo a cíclico dirigido y los nodos de "G" corresponden a las variables de "X". El segundo componente es un set de parámetros "Q" que especifican todas las probabilidades de distribución □(□□|□□□□, □) de condicionales (o densidades) que cuantifican los arcos del grafo. Según (Boaz y Malka, 2011) describe la notación matemática del algoritmo Bayes Net, como la siguiente.

$$P(\boldsymbol{X}|\mathcal{G}) = \prod_{i-1}^{n} P(X_i|\mathbf{Pa}_i, \mathcal{G})$$

Figura 6: Notación matemática del algoritmo Bayes Net.
Fuente: (Boaz y Malka, 2011).

De la figura anterior se define que, P(X|G) = representa la red bayesiana, la cual se define como EL producto de □(□□|□□□□, □) en donde, □□ = Cada variable de X. □□□□ representa los padres de □□ y □ es el grafo formado por □□□□ .

Una red bayesiana es utilizada mayormente para inferencia a mediante el cálculo de sus probabilidades condicionales, para la obtención de la inferencia Bayes Net hace uso de diferentes algoritmos de búsqueda y en esta investigación se hizo uso del algoritmo K2.

1.3.6.2. SVM (SVM)

El algoritmo Support Vector Machine (SVM) nos brinda una manera ágil para el entrenamiento de las Máquinas de Soporte Vectorial a través de la implementación de un algoritmo de programación dinámica secuencial. Por lo que, SVM es un algoritmo que trata de identificar el Hiperplano con el máximo margen que permita separar las instancias de clases de un determinado dato (Platt, 1998).

Para Tribak (2012), SVM:

SVM busca siempre la posibilidad de elegir el menor problema cuadrático, el cual debe ser optimizado en cada interacción. Al implicar este problema a dos multiplicadores de Lagrange, cada iteración buscará a estos dos multiplicadores, llegando así a optimizar y ajustar los valores de la SVM.

Notación Matemática

SVM se basa en diferentes fórmulas para el desarrollo de su algoritmo, pero según (Carmona Suárez, 2014) la más importante es la de la Figura 11:

$$\begin{aligned} \text{máx} \quad & L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < \boldsymbol{x}_i, \boldsymbol{x}_j > \\ \text{s.a.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \geq 0,\ i = 1, \ldots, n \end{aligned}$$

Figura 7: Notación matemática del algoritmo SVM.
Fuente: (Carmona Suárez, 2014)

Referente a la Figura 11, describe un problema de clasificación binaria con un conjunto de datos (□□ , □□) ,..., (□□ , □□) , donde x i es un vector de entrada *e* y i ∈ {-1, +1} es una etiqueta binaria correspondiente a ella.

Donde C es un hiperplano SVM y K(□□ , □□) es la función del núcleo, ambas suministradas por el usuario; y las variables $\alpha_{□}$ son multiplicadores de Lagrange.

Para mejor orientación se anexó la Figura 8 que representa el proceso que realiza el algoritmo.

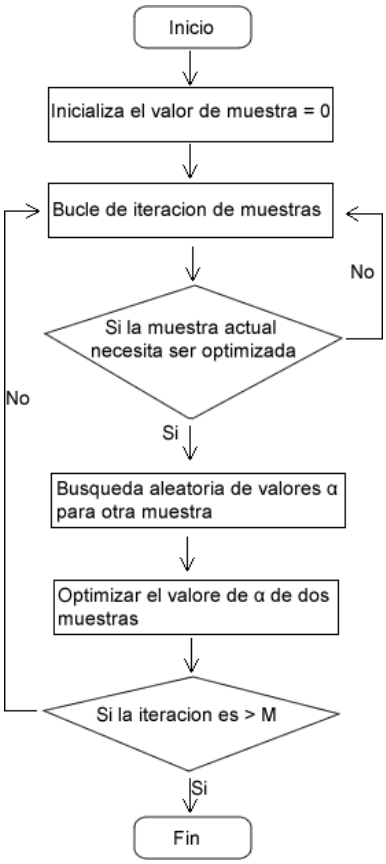


Figura 8: Diagrama de flujo del algoritmo SVM. Fuente: elaboración propia.

1.3.6.3. Random Forest

Según (Han & Phyu, 2016), este algoritmo consiste en la recopilación de árboles de decisión y el voto de la mayoría en estos árboles se utiliza como resultado final. Funciona de manera eficiente con datos de gran tamaño y proporciona una alta precisión. El algoritmo puede proporcionar métodos efectivos para estimar los datos faltantes. El analista no necesita realizar ninguna selección de variables o reducción de datos. No es necesario cambiar la escala, transformar o modificar los datos. Cultivar una gran cantidad de árboles forestales al azar no crea un riesgo de ajuste excesivo.

Los pasos de procesamiento en el algoritmo de bosque aleatorio son los siguientes:

1. Elija T- número de árboles para crecer.

2. Elija m- número de variables utilizadas para dividir cada nodo m << M, donde M es el número de variables de entrada. Me mantengo constante mientras crece el bosque.

3. Cultive árboles T. al cultivar cada árbol, haga lo siguiente:

    a) Construya una muestra bootstrap de tamaño n extraída de Sn con reemplazo y haga crecer un árbol a partir de esta muestra bootstrap.

    b) Al cultivar un árbol en cada nodo, seleccione m variables al azar y utilícelas para encontrar la mejor división.

    c) Haga crecer el árbol al máximo. No hay poda.

4. Para clasificar el punto X, recolecte votos de todos los árboles del bosque y luego use la votación por mayoría para decidir sobre la etiqueta de la clase.

1.4. Formulación del Problema.

¿Qué algoritmo aprendizaje automático presentará el mejor rendimiento en la identificación de ataques a sitios web?

1.5. Justificación e importancia del estudio.

Los archivos LOG representan una extensa fuente datos y que se encuentra accesible en cualquier momento, lo que hace más factible su empleo en la detección de ataques a aplicaciones web en comparación con otros métodos tradicionales como IDS basados en red o en host, firewalls o bloqueo de los puertos HTTP y HTTPS, los cuales se están volviendo menos eficientes en los intentos de controlar esta problemática.

Esta investigación describió una alternativa para las empresas que hacen uso frecuente de tecnologías de información (TI) y que están expuestas a riesgos de seguridad de la información en sus servidores web, contribuyendo en la identificación de patrones comunes de ataques propuestos (Injection, Cross Site Scripting y Directory traversal) mediante sus archivos log, además de realizar una evaluación de rendimiento de los algoritmos Bayes Net, Random Forest y SVM en la clasificación correcta de estos tipos de ataques.

1.6. Hipótesis.

Mediante el análisis de archivos logs será posible identificar accesos maliciosos a los servidores web Apache usando algoritmos de aprendizaje automático.

1.7. Objetivos.

1.7.1. Objetivo general.

Evaluar el rendimiento de algoritmos de aprendizaje automático en la identificación de ataques a sitios web utilizando logs de servidor.

1.7.2. Objetivos específicos.

A. Obtener los archivos log de un servidor web Apache.
B. Pre procesar los archivos log.
C. Extraer patrones de identificación de accesos maliciosos.
D. Elegir los algoritmos a evaluar.
E. Aplicar herramienta de análisis de rendimiento de los algoritmos en la detección de accesos maliciosos.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación.

2.1.1. Tipo de investigación

El presente trabajo representa una investigación Cuantitativa debido a que se analiza diversos elementos, los cuales se pueden medir y cuantificar. Toda la información se obtendrá basado en muestras de la población, y sus resultados son extrapolables a toda la población, con un determinado nivel de error y nivel de confianza.

2.1.2. Diseño de la investigación

El tipo de diseño es de tipo experimental, de sub tipo cuasi experimental con post prueba y se aplicará a un solo grupo debido a que se analizará los Logs y se manipulará de forma deliberada la variable independiente, que viene a ser Algoritmos de aprendizaje automático, esto con la finalidad de poder observar sus efectos en la variable independiente planteada.

2.2. Población y muestra.

2.2.1. Población

De acuerdo a la revisión literaria realizada en la sección de trabajos previos, en donde se presentó 10 papers de investigación relacionados a la presente investigación, se ha evidenciado que entre todos los papers fueron 10 diferentes algoritmos los utilizados para el mismo fin que persigue nuestra investigación, es por ello que hemos considerado esos 10 algoritmos como nuestra población.

Los algoritmos identificados fueron Logistic regression, Decision Tree, Random Forest, Support Vector Machine, Multilayer Perceptron, Long short-term memory, Convolutional Neural Network, Recurrent Neural Networks, Bayes Net y Naive Bayes.

2.2.2. Muestra

La muestra poblacional se estableció por conveniencia decidiendo elegir a los algoritmos Random Forest, Bayes Net y Support Vector Machine. Los detalles y explicación de esta elección se describen en la fase de elección de algoritmos ubicada en la sección 3.3.1.4. de la presente investigación.

2.3. Variables y Operacionalización.

2.3.1. Variable independiente

Algoritmos de Aprendizaje automático.

2.3.2. Variable dependiente

Identificación de accesos maliciosos.

2.3.3. Operacionalización

Tabla 1
Operacionalización de variables dependiente e independiente del proyecto de investigación.

VARIABLES	DIMENSIONES	INDICADORES	DESCRIPCIÓN	FÓRMULA
INDEPENDIENTE Algoritmos de clasificación.	Rendimiento (Matriz de confusión de Kohavi y Provost.)	Exactitud	Fracción de sesiones que se detectan correctamente.	Exactitud = VN + VP / VN + FP + FN + VP
		Precisión	Fracción de sesiones correctamente clasificadas como ataques de todas las sesiones clasificadas como ataques.	Precisión = VP / FP + VP
		Recall	Proporción de ataques detectados a todos los ataques.	Recall = VP / FN + VP
DEPENDIENTE Identificación de accesos maliciosos.	Identificación de ataques	Inyección	Fracción de instancias clasificadas como ataque de Inyección.	Inyección = Total de instancias – (XSS + DT)
		XSS	Fracción de instancias clasificadas como ataque de Cross Site Scripting.	XSS = Total de instancias – (Inyección + DT)
		DT	Fracción de instancias clasificadas como ataque de Directory Traversal.	DT= Total de instancias – (Inyección + XSS)

Fuente: Elaboración propia.

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.

Para esta investigación se eligió la metodología OWASP que es una metodología de auditoría de seguridad web, de acceso libre y de colaboración.

Esta metodología fue creada por la fundación OWASP (Open Web Application Security Project) y es tomada como referente en auditorías de seguridad. (Tarlogic, 2017)

OWASP se rige por 3 principios básicos:

a) Identificar los agentes de amenaza.
b) Identificar vulnerabilidades que puedan ser explotados por los agentes de amenaza.
c) Estimar el impacto de la materialización de la amenaza sobre el negocio.

Además, existen dos modalidades de ejecución de esta metodología, la primera es mediante la Auditoría OWASP Top 10 y la segunda la Auditoría OWASP completa. Para esta investigación se siguió los conceptos de la Auditoría Top 10.

A continuación, se detalla de qué manera la investigación se adecuo a los 3 principios básicos de OWASP.

a) Identificar agentes de amenaza.

   Para esta investigación se identificó a los agentes de amenaza como cualquier individuo, grupo u organización que puede tener conocimiento de los riesgos existentes y que con la aplicación de diversas herramientas poder encontrar vulnerabilidades y sacar provecho de las mismas.

b) Identificar vulnerabilidades.

   La identificación de vulnerabilidades en esta investigación se basó en la modalidad de Auditoria OWASP TOP 10, de la cual se eligió a las vulnerabilidades TOP 1 (Directory traversal) y, TOP 3 (Injection, XSS) a ser estudiadas, e identificadas por ser calificadas de mayor impacto y por ser las más adecuadas para proveer resultados en una investigación como la nuestra.

c) Estimar el impacto

La metodología OWASP clasifica el impacto mediante una escala numérica de 0 a 9 dividida en 3 grupos como se visualiza en la Tabla 2.

Tabla 2

Estimación del impacto de la materialización de una amenaza.

Probabilidad	Impacto
0 - 2	Bajo
3 - 5	Medio (Moderado)
6 - 9	Alto (Severo)

Fuente: (OWASP, 2017).

Basado en la siguiente clasificación de impacto, se concluye que el impacto de las vulnerabilidades a tomar en esta investigación es Alto (Severo) y Medio (Moderado) respectivamente.

2.4.1. Técnicas de recolección de datos

2.4.1.1. La observación

La observación nos va a ser de utilidad en el análisis del progreso de la investigación, pues se basa en la observación de determinados individuos en el momento que desempeñan una tarea.

En la presente investigación, la observación a realizar es para posterior análisis documental y será aplicada a los archivos Log que se obtendrán del servidor web.

2.4.1.2. Análisis documental

El documento de análisis para esta investigación viene a ser el archivo Log.

Además, para la correcta identificación de patrones de ataques, se consultó diversas bases de datos vulnerabilidades conocidas como: Archivos Log, Base de datos OWASP, SANS, CWE, CVE, CAPEC y CVE Details.

2.4.1.3. Diagramas de Flujo

Es un esquema que describe como se ejecutan las tareas de un proceso, sistema o algoritmo informático para finalmente definir como se ejecutaran estas tareas en un computador. Los diagramas de flujo son muy utilizados en diversos campos y

permiten documentar, estudiar, planificar, mejorar y comunicar procesos complejos mediante estos diagramas claros y fáciles de comprender. (LucidChar, 2017)

En la presente investigación se empleó varios diagramas de flujo para representar secuencia de procesos largos para poder saber rápidamente cómo se relacionan y ejemplos de eso tenemos en las Figuras 8, 13 y 21.

2.4.2. Instrumentos de recolección de datos

Según la técnica de observación se elaboraron fichas de observación para registrar la información observada. En la imagen siguiente se muestra la ficha de información elaborada para registrar los archivos log obtenidos del servidor web Apache. La ficha completa se encuentra en el Anexo 3.

Ficha de observacion de archivos logs descargados del sitio web LetrasHQ.com

Fecha de elaboracion :	04/11/2017
Numero de archivos :	60
Tamaño minimo :	26,77
Tamaño maximo :	42,87
Tamaño total (Mb) :	1981,25

Archivos Log del sitio LetrasHQ.com			
ID	**Archivo**	**Tamaño (Mb)**	**Fecha**
1	Sat, Aug 12, 2017.log	42,87	12/08/2017
2	Sun, Aug 13, 2017.log	35,75	13/08/2017
3	Mon, Aug 14, 2017.log	32,44	14/08/2017
4	Tue, Aug 15, 2017.log	33,41	15/08/2017
5	Wed, Aug 16, 2017.log	34,96	16/08/2017
6	Thu, Aug 17, 2017.log	34,98	17/08/2017

Figura 9: Ficha de observación de archivos logs descargados del sitio LetrasHQ.com. Fuente: Elaboración propia.

También se elaboró una ficha de observación para registrar el análisis de rendimiento de los algoritmos de aprendizaje automático y su formato es como se muestra en la siguiente figura:

Ficha de observación de Análisis de rendimiento del algoritmo Random Forest

Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Training set
Numero de Instancias	809
# Instancias procesadas correctamente	809
Tiempo de Construcción	0,26
Tiempo de entrenamiento	0,08
Exactitud Total	1,000
Precisión Total	1,000
Recall Total	1,000

Matriz de confusión

Clasificación	a	b	c
a = XSS	374	0	0
b = Iniection	0	345	0
c = DT	0	0	90

Evaluación de rendimiento

Ataque	Exactitud	Precisión	Recall
XSS	1,000	1,000	1,000
Iniection	1,000	1,000	1,000
DT	1,000	1,000	1,000

Figura 12: Ficha de observación de Análisis de rendimiento de un algoritmo. Fuente: Elaboración propia.

Fueron varias fichas de observación elaboradas las que así mismo se ubicaron como anexo 5 de esta investigación.

2.5. Procedimiento de análisis de datos.

El proceso de recolección de información de la presente investigación inicia con la descarga de los archivos Log desde el servidor web Apache.

El proceso de cómo se hizo la recolección y transformación de los datos para que tengan utilidad en la investigación y permitan brindar una solución al problema planteado se detalla en la Figura 13.

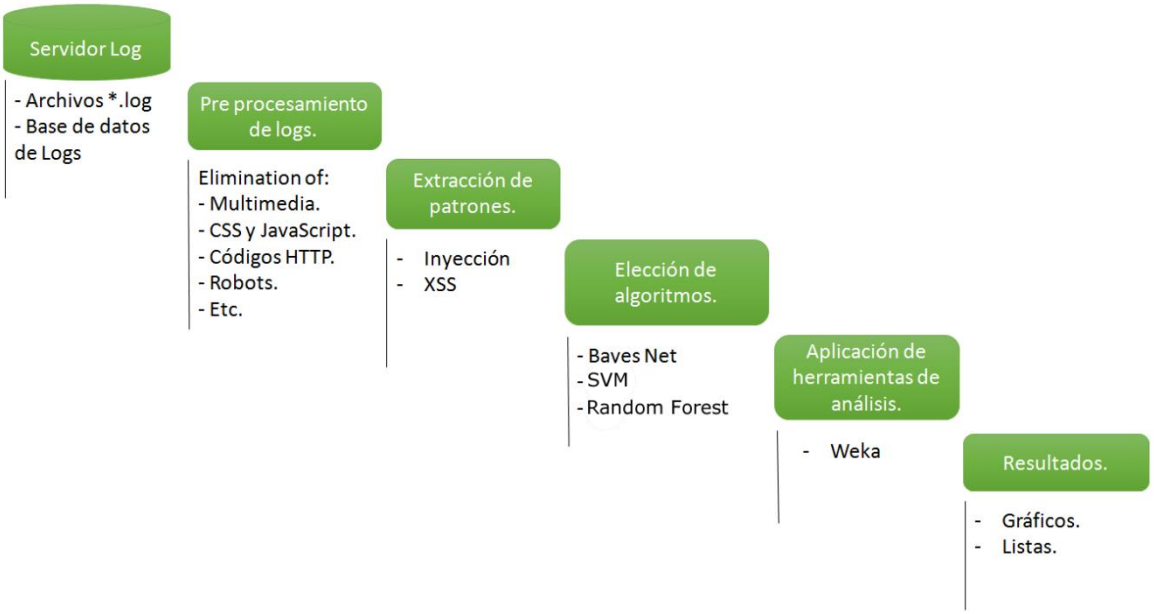


Figura 13: Enfoque propuesto para el desarrollo del proyecto de investigación. Fuente: Elaboración propia.

Se descargaron 60 días de archivos log del servidor web, lo que equivale a 60 archivos log, según el resultado obtenido en el cálculo de la muestra basado en la población de un año que representa 365 días debido a que se generaba un archivo log por día, haciendo un total de 365 archivos LOG.

Los archivos LOG pasaron por la fase de preprocesamiento, en la que se excluyeron líneas de registro que posean datos sin valor para la investigación.

Los archivos generados en el pre procesamiento fueron analizados para intentar identificar registros maliciosos que sean considerados en la etapa de extracción de patrones de ataques.

Después de haber generado un archivo con los patrones de ataques encontrados, se procedió a la selección de los algoritmos a los que se evaluará su rendimiento.

La herramienta empleada para evaluar el rendimiento de los algoritmos seleccionados fue Weka.

2.5.1. Análisis estadístico e interpretación de los datos

Los datos recolectados fueron formateados y sometidos a un análisis mediante diferentes algoritmos de aprendizaje automático haciendo uso de la herramienta de

análisis Weka, algoritmos a los que se evaluó la capacidad de rendimiento mediante los indicadores de exactitud, precisión, y recall se procedió a elaborar cuadros comparativos para identificar qué algoritmos obtienen mejores clasificaciones con respecto a otros.

2.5.1.1. **Variable Independiente:** Algoritmos de aprendizaje automático.

Rendimiento

Para evaluar el desempeño de rendimiento de los algoritmos se empleó la Matriz de Confusión que permite evaluar el desempeño de los algoritmos en un aprendizaje supervisado. (Kohavi & Provost, 1998)

Una matriz de confusión se representa según la Tabla 2 que se presenta a continuación.

Tabla 3

Representación de una matriz de confusión según Kohavi & Provost.

Matriz de confusión		VALOR DE PREDICCION NO	VALOR DE PREDICCION SI
VALOR REAL	NO	Verdadero Negativo (VN)	Falso Positivo (FP)
	SI	Falso Negativo (FN)	Verdadero Positivo (VP)

Fuente: (Kohavi & Provost, 1998)

Los datos de entrada de la matriz de confusión representan el siguiente significado para el contexto de esta investigación:

Tabla 4

Interpretación de los valores generado por la matriz de confusión.

Sigla	Descripción
VP	Cantidad de aciertos para la clase negativa.
FP	Cantidad de errores para la clase positiva.
FN	Cantidad de errores para la clase negativa.
VN	Cantidad de aciertos para la clase positiva.

Fuente: Elaboración propia.

Basándonos en esta tabla se evaluaron distintas medidas de rendimiento que pueden ser aplicadas a los algoritmos y que se explican a continuación:

Exactitud

Es la proporción del número total de predicciones correctas.

Exactitud = VN + VP / VN + FP + FN + VP

La tasa de recuerdo (Recall)

O de verdadero positivo (VP) es la proporción de casos positivos que se fueron identificados correctamente, mediante la ecuación:

Recall = VP / FN + VP

Precisión

La precisión es la proporción de casos positivos predichos que fueron correctos, mediante la ecuación:

Precisión = VP / FP + VP

2.5.1.2. Variable dependiente: Identificación de ataques.

Para medir la variable dependiente se definió como indicadores a Inyección, XSS y DT que representan la cantidad de ataques clasificados como Inyección, Cross Site Scripting y Directory Traversal respectivamente.

Inyección

Este indicador nos proporciona la fracción de instancias clasificadas como ataque de inyección.

Fórmula:

Inyección = Total de instancias – (XSS + DT)

XSS

Este indicador nos proporciona la fracción de instancias clasificadas como ataque de Cross Site Scripting.

Fórmula:

XSS = Total de instancias – (Inyección + DT)

DT

Este indicador nos proporciona la fracción de instancias clasificadas como ataque de Directory Traversal.

Fórmula:

DT= Total de instancias – (Inyección + XSS)

2.6. Criterios éticos.

2.6.1. El consentimiento informado

El consentimiento informado representa la garantía de que el informante ha expresado su intención voluntaria de participar en la investigación, habiendo antes comprendido la información que se le ha otorgado.

2.6.2. La confidencialidad

La confidencialidad se define como la garantía de que la información brindada sea protegida y no divulgada sin consentimiento del sujeto. Esta garantía se lleva a cabo mediante reglas que limitan el acceso a esta información.

2.6.3. Manejo de riesgos

El manejo de riesgos engloba dos aspectos principales que son: Primero, el investigador tiene que cumplir las responsabilidades y obligaciones adquiridas con los informantes, y segundo se refiere al manejo posterior de los datos que se han proporcionado.

2.7. Criterios de Rigor Científico.

2.7.1. Consistencia

El presente trabajo es de carácter científico y formal pues el análisis aplicado a los datos es de carácter profesional y aplicando habilidades, técnicas y conocimientos de ingeniería e investigación que permitan asegurar la consistencia y eficacia de los datos.

2.7.2. Validez

El resultado de la presente investigación será evaluado y analizado minuciosamente con la finalidad de asegurar un resultado válido que permita presentar una solución a la problemática planteada.

2.7.3. Fiabilidad

El usar diversas técnicas e instrumentos de medición en el proceso de recolección y transformación de los datos en información, nos permiten afirmar que la investigación cumple con este principio y asegura que los resultados se asemejen a lo proyectado en un inicio.

2.7.4. Transferibilidad

La presente investigación dejará a disposición mucha información y conocimiento que podrá ser transferida a investigadores que realicen investigaciones en contextos similares.

2.7.5. Neutralidad

La claridad y transparencia con la que se desarrolla la investigación nos permite garantizar la seguridad de que los resultados obtenidos, no serán alterados o desviados por motivaciones, intereses, y/o perspectivas del investigador.

III. RESULTADOS

Los resultados obtenidos en la investigación se presentaron mediante fichas de observación de rendimiento de los algoritmos Bayes Net, Random Forest y SVM, los cuales fueron sometidos a 3 pruebas cada uno, dividido en 1 prueba con los mismos datos de entrenamiento, 1 prueba mediante validación cruzada y 1 prueba mediante un test de porcentajes. Estos tipos de test, así como la descripción de lo que representan las fichas de resultados y cómo es que se calculan, se explican en los párrafos finales del capítulo III. Los resultados parciales de esta investigación fueron publicados en un paper de investigación para la revista Perspectivas de la universidad Inca Garcilaso de la Vega en 2018.

3.1. Resultados en Tablas y Figuras.

3.1.1. Algoritmo Bayes Net

3.1.1.1. Pruebas mediante Training Set.

La prueba de Training Set se basa en emplear para las pruebas de rendimiento, el total de elementos del archivo utilizado para generar el modelo.

Por lo tanto, la matriz de entrada fue de 8 columnas y 809 filas que esperaban una salida de 374 filas de ataques de XSS, 345 filas de ataques de Inyección y 90 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se plasmó el resultado de la prueba de Training set aplicadas al algoritmo Bayes Net, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 5

Prueba de Training set aplicada al algoritmo Bayes Net

Ficha de observación de Análisis de rendimiento del algoritmo Bayes Net	
Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Training set
Numero de Instancias	809
# Instancias procesadas correctamente	800
Tiempo de Construcción	0,05

	Tiempo de entrenamiento	0,01
	Exactitud Total	0,989
	Precisión Total	0,989
	Recall Total	0,989

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	370	3	1
b = Injection	4	341	0
c = DT	1	0	89

Evaluación de rendimiento

Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,989	0,987	0,989
Injection	0,988	0,991	0,988
DT	0,989	0,989	0,989

Fuente: Elaboración propia.

3.1.1.2. Pruebas mediante Cross Validation

La prueba de Cross Validation consiste en dividir los datos de muestra en subconjuntos, en este caso en 10 subconjuntos, donde un subconjunto se utilizó para el entrenamiento y los 9 restantes para armar el modelo. Este proceso es repetido durante 10 iteraciones, con cada uno de los posibles subconjuntos de datos de prueba.

La matriz de entrada fue de 8 columnas y 809 filas que esperaban una salida de 345 filas de ataques de Inyección, 374 filas de ataques de XSS y 90 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Cross Validation aplicadas al algoritmo Bayes Net, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 6

Prueba de Cross Validation aplicada al algoritmo BayesNet

Ficha de observación de Análisis de rendimiento del algoritmo Bayes Net	
Resumen de análisis	
Numero de Análisis	1

Tipo prueba	Cross-validation
Numero de Instancias	809
# Instancias procesadas correctamente	775
Tiempo de Construcción	0
Tiempo de entrenamiento	-
Exactitud Total	0,958
Precisión Total	0,958
Recall Total	0,958

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	356	13	5
b = Injection	13	331	1
c = DT	1	1	88

Evaluación de rendimiento

Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,952	0,962	0,952
Injection	0,959	0,959	0,959
DT	0,978	0,936	0,978

Fuente: Elaboración propia.

3.1.1.3. Pruebas mediante Percentage Split

La prueba de Percentage Split consiste en dividir el conjunto de datos y dividirlo en dos partes, la primera para construir el clasificador y la segunda para hacer la prueba. El porcentaje para entrenamiento por omisión, es de 66%.

Por lo tanto, la matriz de entrada fue de 8 columnas y 534 filas que esperaban una salida de 275 filas, que representan 114 filas de ataques de Inyección, 125 filas de ataques de XSS y 36 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Percentage Split aplicadas al algoritmo Bayes Net, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 7
Prueba de Percentage Split aplicada al algoritmo Bayes Net

Ficha de observación de Análisis de rendimiento del algoritmo Bayes Net

Resumen de análisis

Numero de Análisis	1
Tipo prueba	Percentage split
Numero de Instancias	275
# Instancias procesadas correctamente	259
Tiempo de Construcción	0,02
Tiempo de entrenamiento	0
Exactitud Total	0,942
Precisión Total	0,944
Recall Total	0,942

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	110	4	0
b = Injection	11	114	0
c = DT	1	0	35

Evaluación de rendimiento

Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,965	0,902	0,965
Injection	0,912	0,966	0,912
DT	0,972	1,000	0,972

Fuente: Elaboración propia.

3.1.2. Algoritmo Random Forest

3.1.2.1. Pruebas mediante Training Set.

La prueba de Training Set se basa en emplear para las pruebas de rendimiento, el total de elementos del archivo utilizado para generar el modelo.

Por lo tanto, la matriz de entrada fue de 8 columnas y 809 filas que esperaban una salida de 345 filas de ataques de Inyección, 374 filas de ataques de XSS y 90 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Training set aplicadas al algoritmo Random Forest, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 8

Prueba de Training Set aplicada al algoritmo Random Forest

Ficha de observación de Análisis de rendimiento del algoritmo Random Forest

Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Training set
Numero de Instancias	809
# Instancias procesadas correctamente	809
Tiempo de Construcción	0,26
Tiempo de entrenamiento	0,08
Exactitud Total	1,000
Precisión Total	1,000
Recall Total	1,000

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	374	0	0
b = Injection	0	345	0
c = DT	0	0	90

Evaluación de rendimiento

Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	1,000	1,000	1,000
Injection	1,000	1,000	1,000
DT	1,000	1,000	1,000

Fuente: Elaboración propia.

3.1.2.2. Pruebas mediante Cross Validation

La prueba de Cross validation consiste en dividir los datos de muestra en subconjuntos, en este caso en 10 subconjuntos, donde un subconjunto se utilizó para el entrenamiento y los 9 restantes para armar el modelo.

El proceso es repetido durante 10 iteraciones, con cada uno de los posibles subconjuntos de datos de prueba.

La matriz de entrada fue de 8 columnas y 809 filas que esperaban una salida de 345 filas de ataques de Inyección, 374 filas de ataques de XSS y 90 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Cross Validation aplicadas al algoritmo Random Forest, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 9

Prueba de Cross Validation aplicada al algoritmo Random Forest

Ficha de observación de Análisis de rendimiento del algoritmo Random Forest

Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Cross-validation
Numero de Instancias	809
# Instancias procesadas correctamente	792
Tiempo de Construcción	0,15
Tiempo de entrenamiento	-
Exactitud Total	0,979
Precisión Total	0,979
Recall Total	0,979

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	367	7	0
b = Injection	10	335	0
c = DT	0	0	90

Evaluación de rendimiento			
Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,981	0,973	0,981
Injection	0,971	0,980	0,971
DT	1,000	1,000	1,000

Fuente: Elaboración propia.

3.1.2.3. Pruebas mediante Percentage Split

La prueba de Percentage Split consiste en dividir el conjunto de datos y dividirlo en dos partes, la primera para construir el clasificador y la segunda para hacer la prueba. El porcentaje para entrenamiento por omisión, es de 66%.

Por lo tanto, la matriz de entrada fue de 8 columnas y 534 filas que esperaban una salida de 275 filas, que representan 114 filas de ataques de Inyección, 125 filas de ataques de XSS y 36 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Percentage Split aplicadas al algoritmo Random Forest, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 10

Prueba de Percentage Split aplicada al algoritmo Random Forest

Ficha de observación de Análisis de rendimiento del algoritmo Random Forest	
Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Percentage split
Numero de Instancias	809
# Instancias procesadas correctamente	259
Tiempo de Construcción	0,12
Tiempo de prueba	0,02
Exactitud Total	0,942
Precisión Total	0,943
Recall Total	0,942

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	110	4	0
b = Injection	8	107	0
c = DT	2	2	32

Evaluación de rendimiento			
Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,965	0,917	0,965
Injection	0,936	0,951	0,936
DT	0,889	1,000	0,889

Fuente: Elaboración propia.

3.1.3. Algoritmo SVM

3.1.3.1. Pruebas mediante Training Set.

La prueba de Training Set se basa en emplear para las pruebas de rendimiento, el total de elementos del archivo utilizado para generar el modelo.

Por lo tanto, la matriz de entrada fue de 8 columnas y 809 filas que esperaban una salida de 345 filas de ataques de Inyección, 374 filas de ataques de XSS y 90 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Training Set aplicadas al algoritmo SVM, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 11

Prueba de Training Set aplicada al algoritmo SVM

Ficha de observación de Análisis de rendimiento del algoritmo SVM	
Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Training set
Numero de Instancias	809
# Instancias procesadas correctamente	808
Tiempo de Construcción	1,04
Tiempo de entrenamiento	0,11
Exactitud Total	0,999
Precisión Total	0,999
Recall Total	0,999

Matriz de confusión			
Clasificación	**a**	**b**	**c**
a = XSS	373	1	0
b = Injection	0	345	0
c = DT	0	0	90

Evaluación de rendimiento			
Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,997	1,000	0,997
Injection	1,000	0,997	1,000
DT	1,000	1,000	1,000

Fuente: Elaboración propia.

3.1.3.2. Pruebas mediante Cross Validation

La prueba de Cross Validation consiste en dividir los datos de muestra en sub conjuntos, en este caso en 10 subconjuntos, donde un subconjunto se utilizó para el entrenamiento y los 9 restantes para armar el modelo. El proceso es repetido durante 10 iteraciones, con cada uno de los posibles subconjuntos de datos de prueba.

La matriz de entrada fue de 8 columnas y 809 filas que esperaban una salida de 345 filas de ataques de Inyección, 374 filas de ataques de XSS y 90 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Cross Validation aplicadas al algoritmo SVM, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 12

Prueba de Cross Validation aplicada al algoritmo SVM

Ficha de observación de Análisis de rendimiento del algoritmo SVM	
Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Cross-validation
Numero de Instancias	809
# Instancias procesadas correctamente	786
Tiempo de Construcción	0,37
Tiempo de entrenamiento	-

	Exactitud Total	0,972
	Precisión Total	0,972
	Recall Total	0,972

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	364	9	1
b = Injection	13	332	0
c = DT	0	0	90

Evaluación de rendimiento

Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,973	0,966	0,973
Injection	0,962	0,974	0,962
DT	1,000	0,989	1,000

Fuente: Elaboración propia.

3.1.3.3. Pruebas mediante Percentage Split

La prueba de Percentage Split consiste en dividir el conjunto de datos y dividirlo en dos partes, la primera para construir el clasificador y la segunda para hacer la prueba. El porcentaje para entrenamiento por omisión, es de 66%.

Por lo tanto, la matriz de entrada fue de 8 columnas y 534 filas que esperaban una salida de 275 filas, que representan 114 filas de ataques de Inyección, 125 filas de ataques de XSS y 36 filas de ataques de Directory Traversal.

En las siguientes fichas de observación se registró la prueba de Percentage Split aplicadas al algoritmo SVM, además de un resumen del análisis, la matriz de confusión y el resultado de la evaluación de rendimiento del algoritmo.

Tabla 13

Prueba de Percentage Split aplicada al algoritmo SVM

Ficha de observación de Análisis de rendimiento del algoritmo SVM

Resumen de análisis

Numero de Análisis	1
Tipo prueba	Percentage Split
Numero de Instancias	275
# Instancias procesadas correctamente	257

Tiempo de Construcción	0,36
Tiempo de entrenamiento	0
Exactitud Total	0,935
Precisión Total	0,937
Recall Total	0,935

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	109	5	0
b = Injection	9	116	0
c = DT	3	1	32

Evaluación de rendimiento

Ataque	Exactitud	Precisión	Recall
XSS	0,956	0,901	0,956
Injection	0,928	0,951	0,928
DT	0,889	1,000	0,889

Fuente: Elaboración propia.

3.1.4. Resumen de resultados

A continuación, se describe un resumen del rendimiento de los algoritmos BayesNet, SVM y Random Forest según el tipo de prueba Training Set, que es el tipo de prueba que más confiabilidad otorga a la investigación porque representa la capacidad del algoritmo de clasificar el mismo modelo construido.

3.1.4.1. Resumen para Bayes Net

Ficha de observación

Tabla 14

Resumen de análisis de rendimiento del algoritmo Bayes Net

Ficha Resumen de análisis de rendimiento del algoritmo BayesNet

Según el tipo de prueba realizado

PRUEBA	Exactitud	Precisión	Recall
Training Set	0,989	0,989	0,989
Cross Validation	0,958	0,958	0,958
Precentage Split	0,942	0,944	0,942

Según el ataque identificado y el tipo de prueba Training Set

ATAQUE	Exactitud	Precisión	Recall
XSS	0,989	0,987	0,989

Injection	0,988	0,991	0,988
DT	0,989	0,989	0,989

Fuente: Elaboración propia.

Gráficos

Teniendo en referencia la Tabla 14, el rendimiento del algoritmo Bayes Net según el tipo de prueba al que fue sometido y según el ataque que debió identificar, se elaboró el gráfico para cada tipo de análisis con la finalidad de poder realizar la interpretación de los resultados.

La Figura 14 presenta el rendimiento del algoritmo Bayes Net con base en el tipo prueba aplicado, cuyo resultado representa el promedio general de las evaluaciones hechas en cada tipo de prueba al algoritmo Bayes Net, lo cual permite concluir que el algoritmo obtuvo el mejor rendimiento en las pruebas de Training Set, mostrando un 98,9% lo que representa 3,1% y 4,7% más que en las pruebas de Cross Validation y Percentage Split respectivamente.

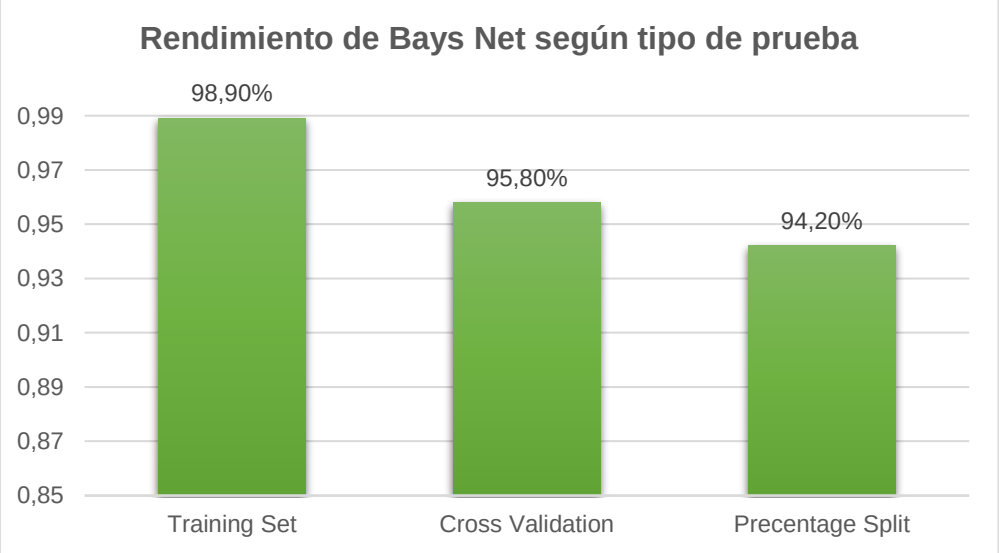


Figura 14: Rendimiento de Bayes Net según tipo de prueba. Fuente: Elaboración propia.

La Figura 15 representa el rendimiento del algoritmo Bayes Net según su capacidad para la identificación de ataques, el resultado representa el promedio de las evaluaciones realizadas según tipo de ataque mediante la prueba de Training Set al algoritmo Bayes Net, dejando como conclusión que el algoritmo obtuvo el mejor rendimiento en la identificación de ataques de XSS con un 98,9% de efectividad, al

igual que en la identificación de ataques de tipo Directory Traversal, y ambos a solo 0,1% por encima del rendimiento de identificación de ataques de Inyección.

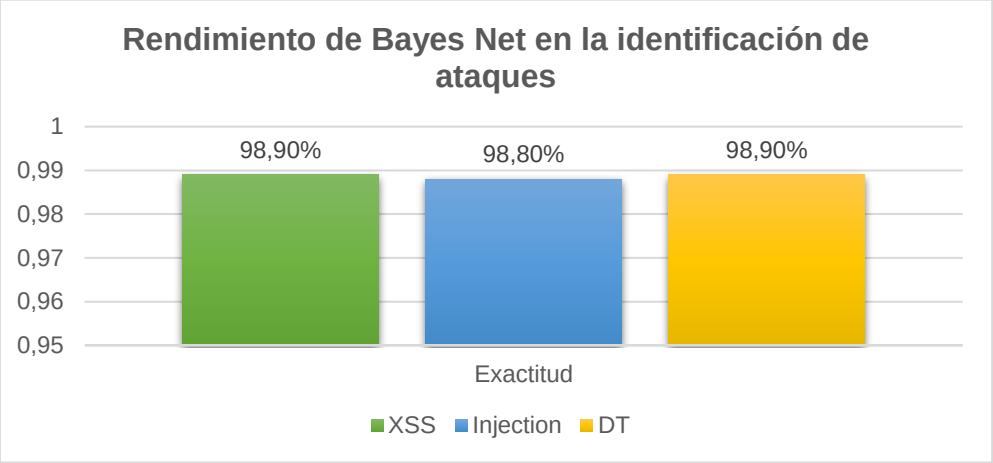


Figura 15: Rendimiento de Bayes Net en la identificación de ataques. Fuente: Elaboración propia.

3.1.4.2. Resumen para Random Forest

Ficha de observación

Tabla 15

Resumen de análisis de rendimiento del algoritmo Random Forest

Ficha Resumen de análisis de rendimiento del algoritmo Random Forest

Según el tipo de prueba realizado

PRUEBA	Exactitud	Precisión	Recall
Training Set	1.000	1.000	1.000
Cross Validation	0.979	0.979	0.979
Precentage Split	0.942	0.943	0.942

Según el ataque identificado y el tipo de prueba Training Set

ATAQUE	Exactitud	Precisión	Recall
XSS	1.000	1.000	1.000
Injection	1.000	1.000	1.000
DT	1.000	1.000	1.000

Fuente: Elaboración propia.

Teniendo en referencia la Tabla 15, el rendimiento del algoritmo Random Forest según el tipo de prueba al que fue sometido y según el ataque que debió identificar,

se elaboró el gráfico para cada tipo de análisis con la finalidad de poder realizar la interpretación de los resultados.

Gráficos

La Figura 16 presenta el rendimiento del algoritmo Random Forest con base en el tipo prueba aplicado, cuyo resultado representa el promedio general de las evaluaciones hechas en cada tipo de prueba al algoritmo Random Forest, lo cual permite concluir que el algoritmo obtuvo el mejor rendimiento en las pruebas de Training Set, mostrando un 100% lo que representa 2,1% y 5,8% más que en las pruebas de Cross Validation y Percentage Split respectivamente.

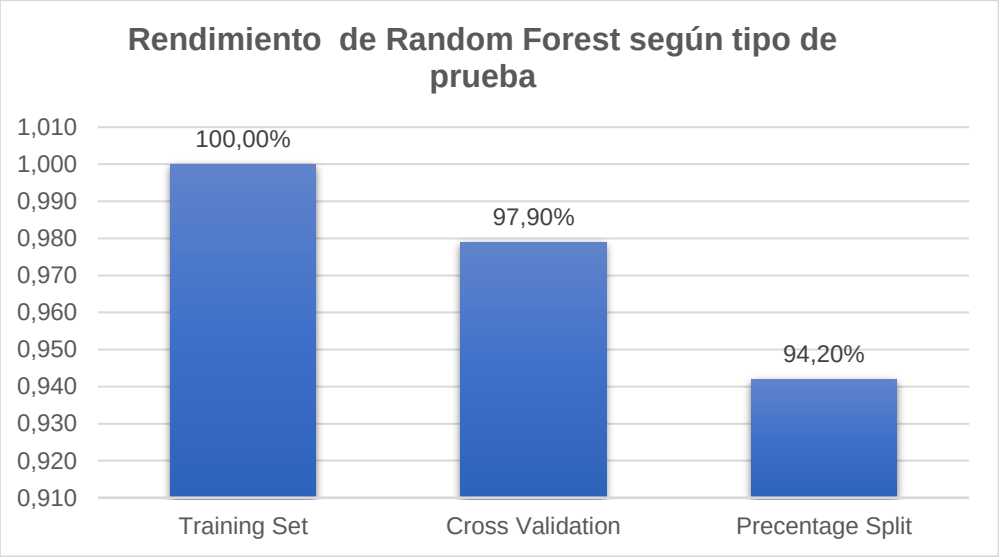


Figura 16: Rendimiento de Random Forest según tipo de prueba. Fuente: Elaboración propia.

La Figura 17 presenta el rendimiento del algoritmo Random Forest según su capacidad para la identificación de ataques, cuyo resultado representa el promedio general de las evaluaciones hechas por cada tipo de ataque mediante la prueba de Training Set al algoritmo Random Forest, lo cual concluye que el algoritmo presenta un rendimiento óptimo en la identificación de los 3 tipos de ataques en evaluación.

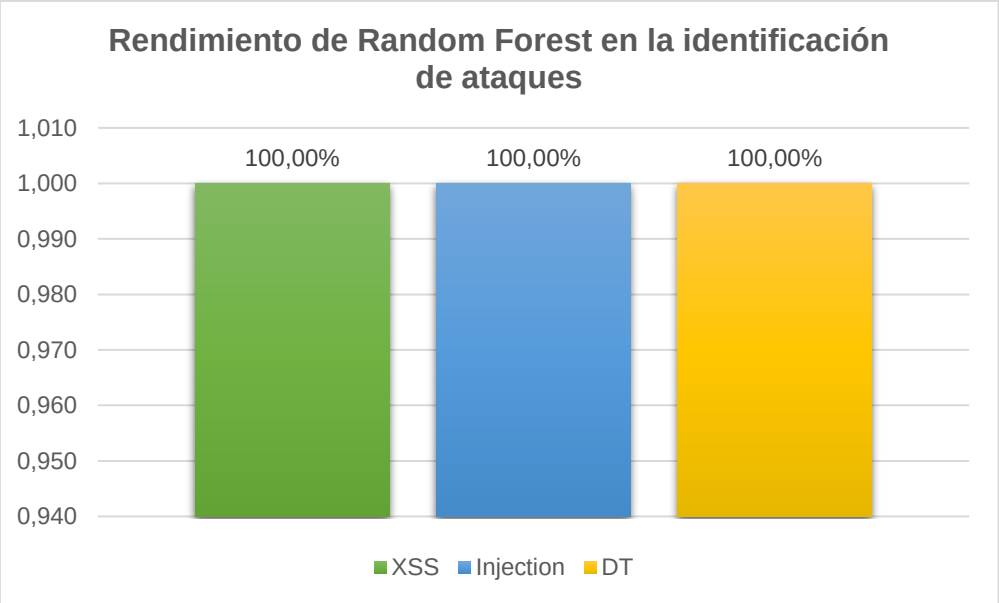


Figura 17: Rendimiento de Random Forest en la identificación de ataques. Fuente: Elaboración propia.

3.1.4.3. Resumen para SVM

Ficha de observación

Tabla 16

Resumen de análisis de rendimiento del algoritmo SVM

Ficha Resumen de análisis de rendimiento del algoritmo SVM

Según el tipo de prueba realizado

PRUEBA	Exactitud	Precisión	Recall
Training Set	0,999	0,999	0,999
Cross Validation	0,972	0,972	0,972
Precentage Split	0,935	0,937	0,935

Según el ataque identificado y el tipo de prueba Training Set

ATAQUE	Exactitud	Precisión	Recall
XSS	0,997	1,000	0,997
Injection	1,000	0,997	1,000
DT	1,000	1,000	1,000

Fuente: Elaboración propia.

Teniendo en referencia la Tabla 16, el rendimiento del algoritmo SVM según el tipo de prueba al que fue sometido y según el ataque que debió identificar, se elaboró el gráfico para cada tipo de análisis con la finalidad de poder realizar la interpretación de los resultados.

Gráfico

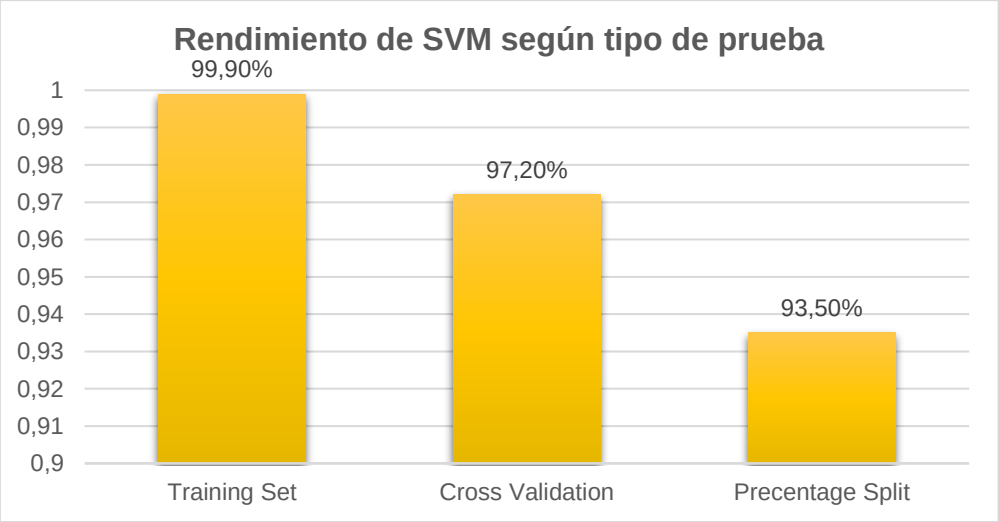


Figura 18: Rendimiento de SVM según tipo de prueba. Fuente: Elaboración propia.

La Figura 18 presenta el rendimiento del algoritmo SVM con base en el tipo prueba aplicado, cuyo resultado representa el promedio general de las evaluaciones hechas en cada tipo de prueba al algoritmo SVM, lo cual permite concluir que el algoritmo obtuvo el mejor rendimiento en las pruebas de Training Set, mostrando un 99,9% lo que representa 2,7% y 6,4% más que en las pruebas de Cross Validation y Percentage Split respectivamente.

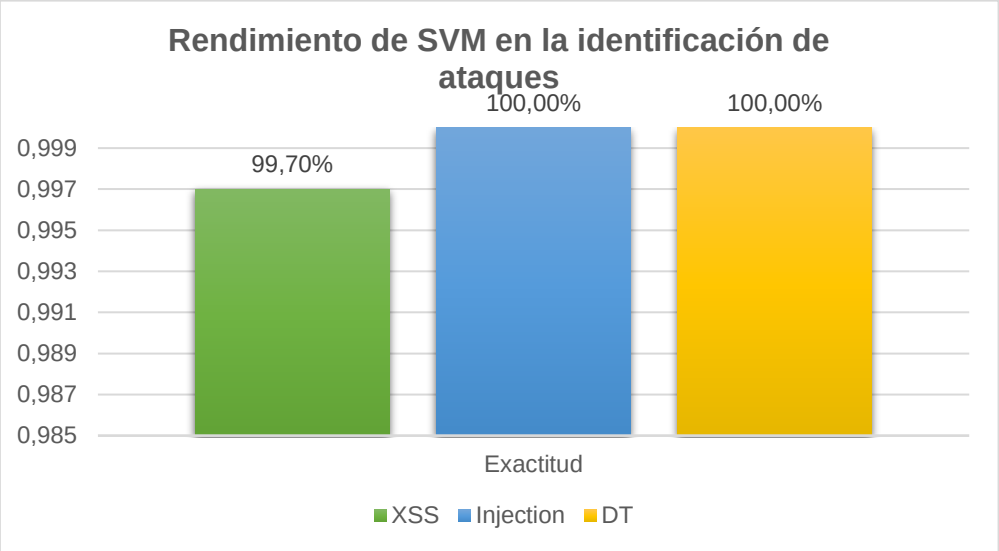


Figura 19: Rendimiento de SVM en la identificación de ataques. Fuente: Elaboración propia.

La Figura 19 representa el rendimiento del algoritmo SVM según su capacidad para la identificación de ataques, cuyo resultado representa el promedio general de las evaluaciones hechas por cada tipo de ataque mediante la prueba de Training Set al algoritmo SVM, lo cual permite concluir que el algoritmo presenta el mejor rendimiento en la identificación de ataques de XSS con un 100% de efectividad, al igual que en la identificación de ataques de tipo Directory Traversal, y ambos a 0,3% por encima del rendimiento de identificación de ataques de Inyección.

Finalmente se adjunta la ficha de Análisis comparativo de rendimiento de los algoritmos BayesNet, Random Forest y SVM para definir que algoritmo presentó el mejor rendimiento de entre los tres evaluados.

Tabla 17

Análisis comparativo de rendimiento de los algoritmos BayesNet, Random Forest y SVM mediante pruebas de Training Set

Ficha de observación de Análisis comparativo de rendimiento de los algoritmos Bayes Net, Random Forest y SVM mediante pruebas de Training Set

Algoritmo	Exactitud	Precisión	Recall
Bayes Net	0.989	0.989	0.989
Random Forest	1.000	1.000	1.000
SVM	0.999	0.999	0.999

Fuente: Elaboración propia.

Según la Tabla 17, se puede concluir que el algoritmo que presentó el mejor rendimiento en la identificación de ataques a sitios web es Random Forest, que presentó una exactitud de 1.000 que equivale a 100% en comparación con Bayes Net y SVM con un rendimiento de 99,9% y 98,9%, lo que permite obtener una diferencia de 0,1% y 1,1% respectivamente y que se grafica en la Figura 20.

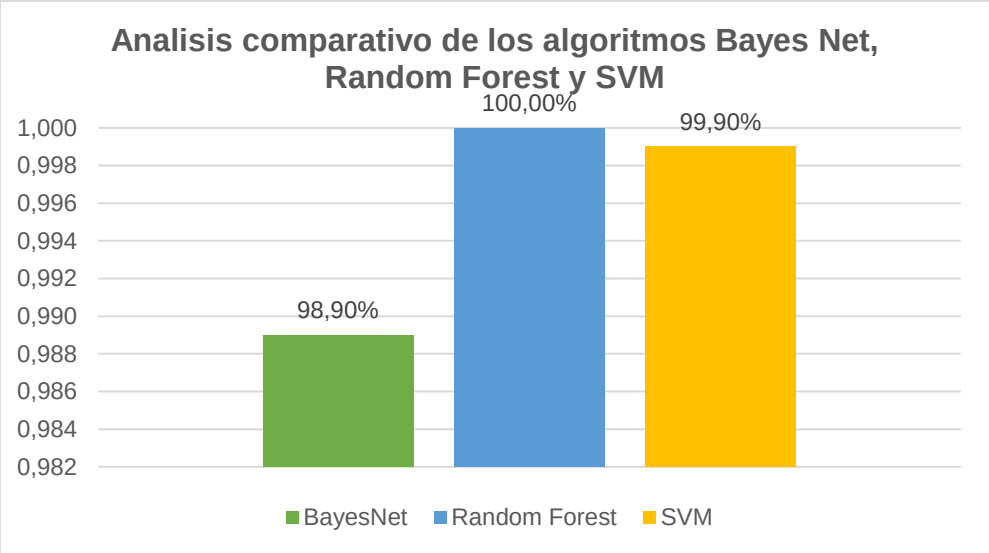


Figura 20: Análisis comparativo de rendimiento de los algoritmos Bayes Net, Random Forest y SVM. Fuente: Elaboración propia.

3.2. Discusión de resultados.

Random Forest fue el algoritmo de mejor rendimiento en las pruebas de Training Set, con una exactitud de 1.000 que equivale a 100%, seguido de SVM y Bayes Net con un rendimiento de 99,9% y 98,9%, lo que representa una diferencia de 0,1% y 1,1% respectivamente, tal como se visualiza en la Tabla 17 y en la Figura 20.

Random Forest fue el algoritmo que ofreció resultados más confiables en la correcta clasificación de ataques según las pruebas de Training Set, presentando un rendimiento de 100% para la identificación de XSS e Inyección y Directory Traversal.

Random Forest fue el algoritmo de mejor rendimiento en las pruebas de Cross Validation, con una exactitud de 0.979 que equivale a 97,9%, seguido de SVM y Bayes Net con un rendimiento de 97,2% y 95,8%, lo que representa una diferencia de 0,7% y 2,1% respectivamente.

SVM tuvo mayor dificultad en la identificación de ataques de XSS en las pruebas de Training Set, con un 99.7% de rendimiento.

Bayes Net tuvo mayor dificultad en la identificación de ataques de Injection en las pruebas de Training Set, con un 98.8% de rendimiento.

Bayes Net fue el algoritmo con el rendimiento general más bajo, pero evaluando la capacidad para interpretar la información suministrada, este algoritmo ofrece un mejor rendimiento en las pruebas de Percentage Split, al igual que Random Forest y superando a SVM, con un margen de 0,7% de diferencia.

Tomando en cuenta que la cantidad de patrones de ataques de tipo Directory Traversal fueron 90 a diferencia de los 374 y 345 patrones de ataques para XSS e Inyección respectivamente, se contempló la posibilidad de que si se hubiera aumentado la cantidad de patrones para ataques de Directory Traversal a una cantidad similar a la de los otros tipos de ataques posiblemente el rendimiento de clasificación de los algoritmos habría sido afectado negativamente.

3.3. Aporte práctico

La propuesta de investigación se dividió en 5 fases, como se muestra en la Figura 21, también registrada como Anexo 4:

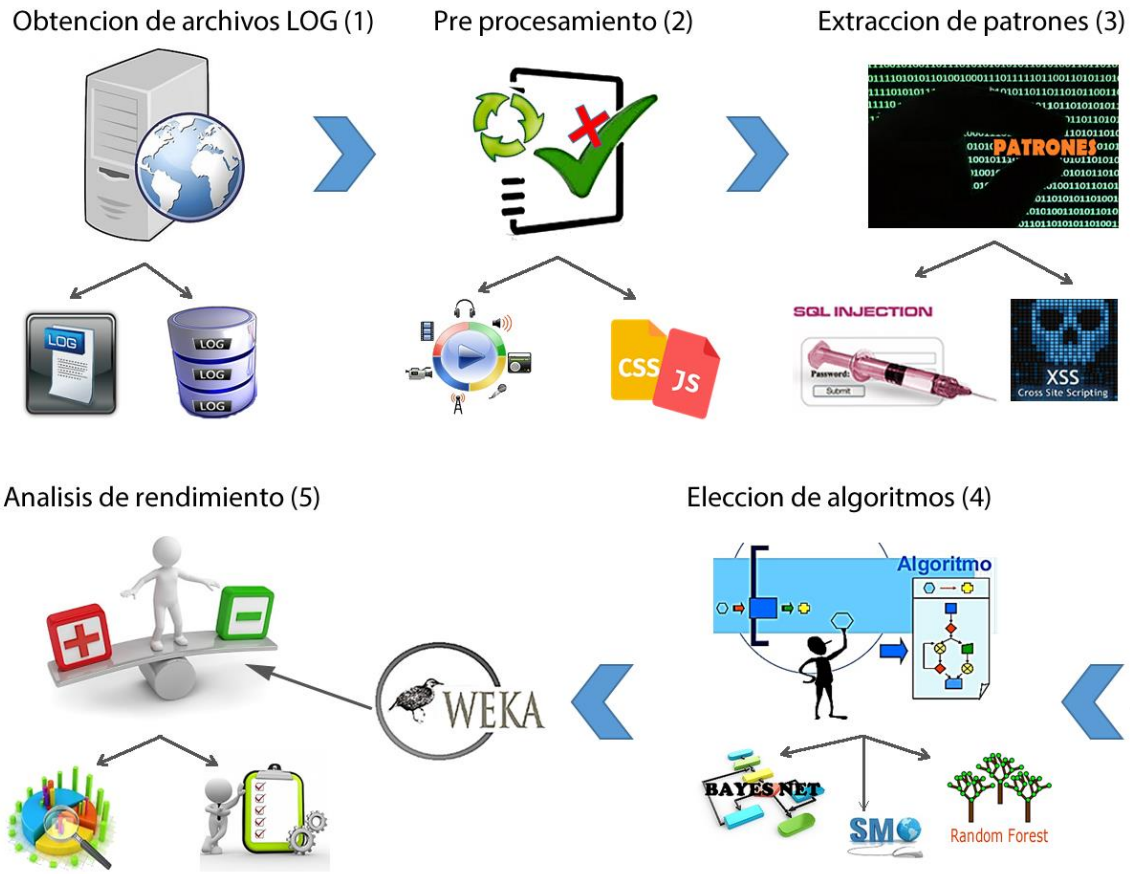


Figura 21: Método empleado en la investigación. Fuente: Elaboración propia

3.3.1. Desarrollo del Método propuesto

3.3.1.1. Fase de obtención de archivos LOG

Los datos se obtuvieron desde el servidor web letrashq.com desde 00:00:00 del sábado 12 de agosto de 2017 hasta el 23:59:59 del miércoles 11 de octubre de 2017. Fueron 60 archivos los descargados, cada archivo estaba nombrado con las tres primeras letras del día (en inglés), seguido de una coma y un espacio en blanco (,) luego las tres primeras letras del mes (en inglés) seguido de un espacio en blanco, el número del día respecto al mes, seguido de una coma (,) y finalmente el número del año como se muestra en la Figura 22.

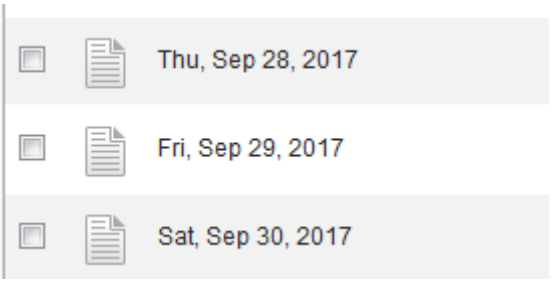


Figura 22: Vista del formato de los archivos log del servidor LetrasHQ.com. Fuente: Elaboración propia.

Los archivos log fueron descargados en una carpeta local, y se registró su nombre, tamaño y fecha de los datos contenidos en una ficha de observación que arrojó los resultados mostrados en la siguiente tabla, la cual también se registró como Anexo 3.

Tabla 18

Ficha de observación de Archivos Log descargados del servidor LetrasHQ.com

ID	**Archivo**	**Tamaño (Mb)**	**Fecha**
1	Sat, Aug 12, 2017.log	42,87	12/08/2017
2	Sun, Aug 13, 2017.log	35,75	13/08/2017
3	Mon, Aug 14, 2017.log	32,44	14/08/2017
4	Tue, Aug 15, 2017.log	33,41	15/08/2017
5	Wed, Aug 16, 2017.log	34,96	16/08/2017
6	Thu, Aug 17, 2017.log	34,98	17/08/2017
7	Fri, Aug 18, 2017.log	32,29	18/08/2017
8	Sat, Aug 19, 2017.log	34,58	19/08/2017
9	Sun, Aug 20, 2017.log	33,95	20/08/2017

10	Mon, Aug 21, 2017.log	27,28	21/08/2017
11	Tue, Aug 22, 2017.log	28,84	22/08/2017
12	Wed, Aug 23, 2017.log	30,43	23/08/2017
13	Thu, Aug 24, 2017.log	32,69	24/08/2017
14	Fri, Aug 25, 2017.log	31,63	25/08/2017
15	Sat, Aug 26, 2017.log	33,46	26/08/2017
16	Sun, Aug 27, 2017.log	30,04	27/08/2017
17	Mon, Aug 28, 2017.log	28,47	28/08/2017
18	Tue, Aug 29, 2017.log	26,77	29/08/2017
19	Wed, Aug 30, 2017.log	28,40	30/08/2017
20	Thu, Aug 31, 2017.log	28,51	31/08/2017
21	Fri, Sep 1, 2017.log	29,65	01/09/2017
22	Sat, Sep 2, 2017.log	33,03	02/09/2017
23	Sun, Sep 3, 2017.log	34,04	03/09/2017
24	Mon, Sep 4, 2017.log	29,61	04/09/2017
25	Tue, Sep 5, 2017.log	28,42	05/09/2017
26	Wed, Sep 6, 2017.log	34,66	06/09/2017
27	Thu, Sep 7, 2017.log	34,95	07/09/2017
28	Fri, Sep 8, 2017.log	34,90	08/09/2017
29	Sat, Sep 9, 2017.log	36,45	09/09/2017
58	Sun, Sep 10, 2017.log	32,00	10/09/2017
30	Mon, Sep 11, 2017.log	28,88	11/09/2017
31	Tue, Sep 12, 2017.log	29,82	12/09/2017
32	Wed, Sep 13, 2017.log	29,82	13/09/2017
33	Fri, Sep 15, 2017.log	30,66	15/09/2017
34	Sat, Sep 16, 2017.log	33,93	16/09/2017
35	Sun, Sep 17, 2017.log	30,29	17/09/2017
36	Mon, Sep 18, 2017.log	31,00	18/09/2017
37	Tue, Sep 19, 2017.log	31,31	19/09/2017
38	Wed, Sep 20, 2017.log	32,37	20/09/2017
39	Thu, Sep 21, 2017.log	34,03	21/09/2017
40	Fri, Sep 22, 2017.log	36,46	22/09/2017
41	Sat, Sep 23, 2017.log	37,02	23/09/2017
42	Sun, Sep 24, 2017.log	36,54	24/09/2017
43	Mon, Sep 25, 2017.log	31,84	25/09/2017
44	Tue, Sep 26, 2017.log	33,49	26/09/2017

45	Wed, Sep 27, 2017.log	35,08	27/09/2017
46	Thu, Sep 28, 2017.log	33,94	28/09/2017
47	Fri, Sep 29, 2017.log	36,11	29/09/2017
48	Sat, Sep 30, 2017.log	40,16	30/09/2017
49	Sun, Oct 1, 2017.log	37,19	01/10/2017
50	Mon, Oct 2, 2017.log	37,43	02/10/2017
51	Tue, Oct 3, 2017.log	36,15	03/10/2017
52	Wed, Oct 4, 2017.log	36,31	04/10/2017
53	Thu, Oct 5, 2017.log	33,77	05/10/2017
54	Fri, Oct 6, 2017.log	36,91	06/10/2017
55	Sat, Oct 7, 2017.log	36,77	07/10/2017
56	Sun, Oct 8, 2017.log	34,05	08/10/2017
57	Mon, Oct 9, 2017.log	31,52	09/10/2017
59	Tue, Oct 10, 2017.log	28,90	10/10/2017
60	Wed, Oct 11, 2017.log	30,04	11/10/2017

Fuente: Elaboración propia

Según la tabla anterior, se obtuvo 60 archivos con tamaños entre los 26,77 y 42,87 megabytes (Mb) y que arrojaron un total de 1981,25 Mb equivalente a 1,93 gigabytes (Gb).

Después de haber obtenido y registrado los 60 archivos, se procedió a la selección de registros útiles en la investigación, proceso que se explica a detalle en la fase de pre procesamiento de Logs.

3.3.1.2. Fase de pre procesamiento de Logs.

En esta fase se realizó la limpieza de Logs para descartar los LOGs de acceso que no son de utilidad para el presente proyecto. Cada Log contiene un aproximado de 160 mil líneas, donde cada línea es una consulta al servidor.

Aquí se realizó la clasificación de líneas de los archivos log para determinar qué líneas son útiles para la investigación y qué líneas deben ser descartadas.
La clasificación se basa en distinguir que la consulta contenga un grupo de cadenas de texto específicas y al mismo tiempo no debe contener otro grupo de cadenas, grupos que se elaboraron en base a las siguientes tablas:

Tabla 19

Solicitudes de un tipo de método específico.

Método	**Descripción**
GET	Solicitud de recurso.
POST	Envío de recurso.
OPTIONS	Describe las opciones de comunicación para el recurso de destino.
HEAD	Solicitud de recurso sin cuerpo en la respuesta.
CONNECT	Establece un túnel hacia el servidor identificado por el recurso.
DELETE	Borra un recurso en específico
PUT	Reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición

Fuente: Elaboración propia.

Tabla 20

Solicitudes a archivos de imagen.

Nombre	**Extensión**
Imagen JPEG	.jpeg
Imagen JPG	.jpg
Imagen GIF	.gif
Icono	.ico
Portable Network Graphic	.png

Fuente: Elaboración propia.

Tabla 21

Solicitudes a archivos de vídeo.

Nombre	**Extensión**
Audio y Video Intercalado	.avi
Flash Live Video	.flv
MPEG-4 parte 14.	.mp4
Grupo de Expertos de Películas	.mpg
Grupo de Expertos de Películas	.mpeg

Windows Media Video	.wmv

Fuente: Elaboración propia.

Tabla 22

Solicitudes a archivos de aplicación estáticos.

Nombre	**Extensión**
Archivo HTML	.html
Archivo HTM	.htm
Archivo XML	.xml

Fuente: Elaboración propia.

Tabla 23

Solicitudes a archivos de aplicación dinámica.

Nombre	**Extensión**
Archivo PHP	.php
Archivo ASP	.asp
Archivo ASPX	.aspx
Archivo Java Server Page	.jsp
Archivo Javascript	.js

Fuente: Elaboración propia.

Tabla 24

Solicitudes a archivos de texto.

Nombre	**Extensión**
Archivo TXT	txt
Archivo INI	.ini
Hoja de estilos	.css

Fuente: Elaboración propia.

Tabla 25

Solicitudes de archivos de fuentes.

Nombre	**Extensión**
Archivo EOT	.eot
Archivo WOOF	.woff

Fuente: Elaboración propia.

Tabla 26

Solicitudes con código de estado específico.

Código HTTP	**Descripción**
100	Informativo
200	Exitoso
206	Contenido parcial
300	Re direccionamiento
301	Movido permanentemente
400	Error del cliente
404	No encontrado
500	Error del servidor
505	Versión HTTP no soportada

Fuente: Elaboración propia.

Tomando estas tablas como condiciones de filtrado para el pre procesamiento de Logs se elaboró una ficha de observación en la que se describen los resultados del análisis realizado a cada archivo log.

Para obtener esta ficha se elaboró un mini sistema basado en lenguaje java que permitió la selección de un archivo log, el cual es sometido a un análisis línea por línea para determinar qué líneas son útiles y que líneas serán descartadas.

Como condiciones por defecto en el filtrado, se definió como líneas válidas, aquellas que contengan el método HTTP de tipo GET o HEAD, y se excluyó las líneas que no contengan tales métodos.

También, se elaboró una lista preliminar de cadenas de texto que no debe estar presente en una línea del archivo Log que va a ser útil para el proyecto, esta lista se expone en la siguiente figura.

Valor	Descripcion
200	Exitoso
.png	Imagen PNG
.jpg	Imagen JPG
.ico	Icono
.css	Hoja de estilos
.js	Archivo Javascript
206	Contenido parcial
.gif	Imagen GIF
.eot??	Fuente
.woof	Fuente

Figura 23: Lista de preliminar de cadenas de texto que no debe contener una línea del archivo Log**.** Fuente: Elaboración propia

La columna valor de la figura 23 representa la cadena de texto que no debe contener una línea útil para la investigación y la condición es que si o si, ninguna de las cadenas debe estar presente.

El funcionamiento del sistema se explica en la siguiente imagen y a continuación se explica el proceso paso por paso.

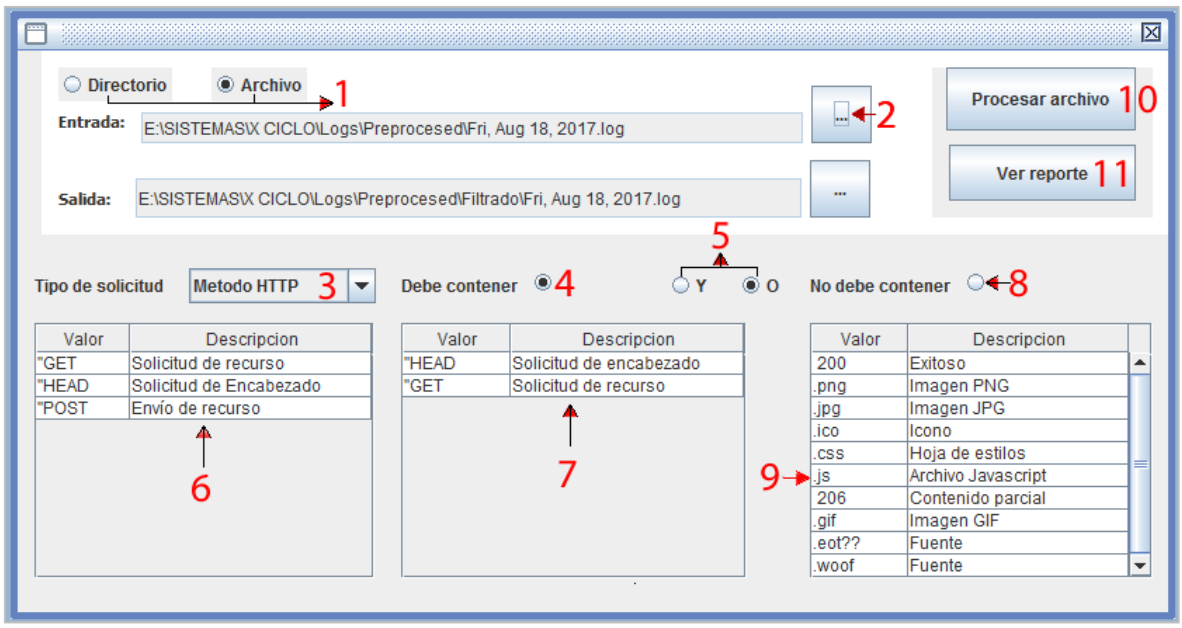


Figura 24: Formulario principal del sistema de pre procesamiento de archivos log. Fuente: Elaboración propia

PASO “1”: En este paso, se deberá elegir y seleccionar “Directorio o Archivo”, según lo que desea cargar. Se seleccionará Directorio cuando se desee procesar una carpeta completa de Logs, y Archivo, cuando se desee procesar un Log individual. En este caso se seleccionó la opción que permite procesar un solo archivo.

PASO “2”: El paso siguiente consiste en seleccionar la ruta del Directorio de Logs o el Archivo Log, la cual se mostrará en la línea de Entrada; además, en la línea de Salida se generará una nueva ruta donde se guardarán los archivos ya procesados.

PASO “3”: Se tiene una lista de Condiciones de Filtrado basado en las tablas descritas anteriormente, en este paso, se deberá elegir una de ellas, según los que se desee filtrar, y automáticamente sus valores y descripción se mostrará en el cuadro del paso Nº 6. En este caso se seleccionó la Condición de Filtrado: Método HTTP que contiene como valores a GET, HEAD y POST.

PASO “4”: Después del paso 3, se deberá seleccionar obligatoriamente la opción “Debe Contener”, ya que esto permitirá agregar la cadena de filtrado de la tabla del paso 6 (se verá a continuación detalladamente cómo funciona en los pasos 6 y 7).

PASO “5”: En este paso se deberá elegir, según la condición que desee seguir para el proceso de sus Logs, las opciones: Y u O. Se elegirá “Y” cuando su condición es el siguiente: Debe contener todos los valores al mismo tiempo; y se elegirá “O”, cuando desee que su condición sea: Debe contener cualquiera de los valores elegidos. En este caso se seleccionó “O”, esto quiere decir que según los valores de la tabla del Paso 7 y la condición dada: La línea del archivo debe Contener HEAD o GET.

PASO “6”: En la tabla del “Paso 6”, cargará todos los valores según lo que se haya seleccionado en el Paso 3, en este paso se podrá seleccionar un valor y con doble clic se podrá agregar a la tabla del Paso 7. En este caso se eligió el valor GET y HEAD.

PASO “7”: En la tabla de este paso, se cargará todos los valores que se hayan seleccionado de la tabla del Paso 6, con la ayuda de la selección del Paso 4; si quizá agrego un valor que no desea que contenga esa tabla, se selecciona el valor no deseado y con doble clic, desaparecerá de esta tabla. En este caso se agregó el valor HEAD y GET, es decir el archivo Log procesado deberá contener los métodos HEAD y GET.

PASO “8”: En este paso se selecciona la opción “No debe Contener”, y se regresa al Paso 3, se selecciona la Condición de Filtrado, y pasa al Paso 6, selecciona el valor y con doble clic se agregará a la tabla del Paso 9, esto quiere decir que seleccionara todas las líneas del Log a procesar que contengan los valores de la Tabla del Paso 7, así mismo esas líneas no deben contener los valores de la tabla del paso 9.

PASO “9”: Esta tabla cargará todos los valores que se han seleccionado de la tabla del Paso 6. Este paso es el penúltimo para el procesamiento del Log, ya que la tabla de este paso contendrá todos los valores que hará que eliminen las líneas que contengan los valores de la tabla del Paso 7. Si por algún caso, se agregó un valor que no desee que contenga, se ubica en el valor no deseado y con doble clic hará que se elimine de la tabla.

PASO “10”: Este paso es el último para que empiece el procesamiento de Logs, se dará clic en el botón “Procesar Archivo”, luego se debe confirmar si se desea grabar el análisis en la base de datos o no, y se esperará hasta que aparezca una ventana con un mensaje que confirme que el archivo ha sido procesado correctamente.

Seguido, se abrirá una nueva ventana que muestra los resultados del análisis efectuado, además de una tabla que contiene las líneas útiles resultantes del procesamiento del Log como se muestra en la Figura 25.

PASO “11”: (Opcional) En este paso, se hará clic en el botón Ver Reporte y este mostrará un nuevo formulario según se describe en el segundo párrafo del paso 10.

La ejecución de los pasos antes descritos da como resultado un archivo procesado el cual contendrá las direcciones URL resultantes del procesamiento. Además, si se eligió guardar el análisis en la base de datos, este resultado también estará disponible en una base de datos creada para guardar estos análisis.

A continuación, se presenta la imagen que representa el formulario resultante del paso 10.

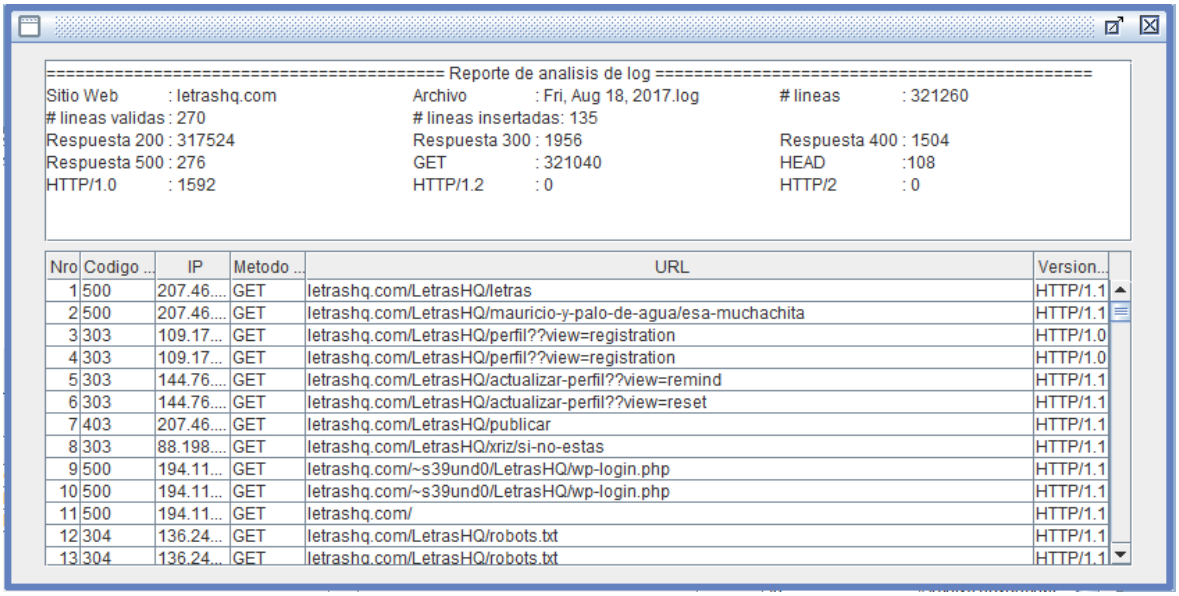


Figura 25: Formulario de reporte del análisis de pre procesamiento de un archivo log. Fuente: Elaboración propia.

La Figura 25 representa el formulario resultante al seguir el Paso 11 descrito anteriormente y su descripción es como sigue:

Sitio Web, representa el nombre del sitio web al que pertenece el archivo Log.

Archivo, representa el nombre del archivo log procesado.

 # líneas, representa el total de líneas que contiene el archivo log procesado.

 # líneas válidas, representa el número de líneas útiles encontradas en el archivo log procesado

 # líneas insertadas, representa el número de líneas válidas que fueron insertadas en la base de datos.

Respuesta 200, representa la cantidad de líneas que contienen un código de respuesta HTTP de clase 200.

Respuesta 300, representa la cantidad de líneas que contienen un código de respuesta HTTP de clase 300.

Respuesta 400, representa la cantidad de líneas que contienen un código de respuesta HTTP de clase 400.

Respuesta 500, representa la cantidad de líneas que contienen un código de respuesta HTTP de clase 500.

HTTP/1.0, representa la cantidad de líneas que contienen la versión 1.0 del protocolo HTTP.

HTTP/1.1, representa la cantidad de líneas que contienen la versión 1.1 del protocolo HTTP.

HTTP/2, representa la cantidad de líneas que contienen la versión 2 del protocolo HTTP.

Finalmente se visualiza una tabla que según cada columna contiene el número de línea útil, el código de respuesta HTTP, la dirección IP de procedencia de la solicitud, el método HTTP de la solicitud, la URL de la solicitud y la versión del protocolo utilizada en la solicitud.

Toda esta información es almacenada en la base de datos, siempre se haya elegido esa opción en el Paso 10 de la Figura 24.

En la Figura 26 se muestra parte de un archivo log procesado en donde se muestran las URLS que serán de utilidad para la Fase de extracción de patrones de identificación de ataques o accesos maliciosos descrita en la sección 3.3.1.3. Otra muestra similar a la siguiente figura se encuentra en el Anexo 2.

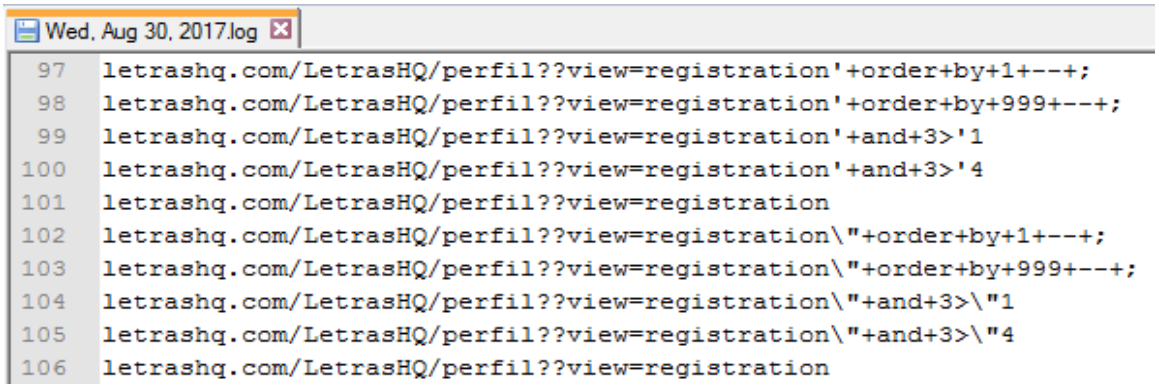


Figura 26: Fracción de un archivo log pre procesado. Fuente: Elaboración propia.

3.3.1.3. Fase de extracción de patrones de identificación de ataques o accesos maliciosos

En esta fase se realizó el análisis de direcciones URI para la identificación de patrones que representen un intento de vulneración al servidor.

3.3.1.3.1. Descripción de los tipos de URL encontrados

Para poder identificar estos patrones, se tomó en cuenta las URI obtenidas en la fase de preprocesamiento que como método de acceso sean GET o HEAD, pues estos dos métodos HTTP son similares y el envío de parámetros se realiza mediante la URL, lo que nos va a permitir identificar los valores que intentan ingresar los atacantes.

Las URI tienen un esquema definido como:

http[s]://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument

Tabla 27

Descripción del esquema de una URL

Sección	Descripción
http[s]://	Protocolo
www.example.com	Autoridad o Dominio
:80	Puerto
/path/to/myfile.html	Ruta de acceso
?key1=value1&key2=value2	Consulta
#SomewhereInTheDocument	Fragmento

Fuente: Elaboración propia.

La definición más exacta de URL se encuentra en la sección de sistemas teóricos conceptuales, como URI.

Los archivos log registran estas URI en el formato descrito anteriormente cuando las consultas son de tipo GET y HEAD, es por eso que, para la detección de inyección, XSS y Directory Traversal se analizara solo este tipo de consultas.

También hay registros log que almacenan URI relativas, donde se omite el protocolo y el puerto, y el dominio se representa con una barra invertida (/), o como en otros casos donde se omite también la ruta de acceso y la consulta parte directamente desde el dominio:

/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument

/?key1=value1&key2=value2#SomewhereInTheDocument

También existen URI que no incluyen en la ruta de acceso el tipo de archivo al que se está accediendo sino solo el nombre del directorio como, por ejemplo:

/path/to/myfolder?key1=value1&key2=value2#SomewhereInTheDocument

Además, hay URI que reemplazan el dominio por su dirección IP, la cual puede estar presente en todos los ejemplos antes mencionados.

3.3.1.3.2. Identificación de patrones de ataques.

Se ha identificado patrones que resaltan la diferencia de una URL normal a una URL que contiene consultas maliciosas, un ejemplo de entidad identificadora es la fundación Mitre, una organización estadounidense que provee entre otros, soporte en ingeniería de sistemas al gobierno de Estados Unidos de América.

Esta fundación posee una base de datos con documentación sobre distintas vulnerabilidades de seguridad cibernética que son conocidas públicamente. Cada vulnerabilidad aprobada posee su propia documentación y hasta la actualidad hay 112214 vulnerabilidades documentadas en esta base de datos denominada CVE.

Partiendo de esta base de datos como referencia se procedió a identificar los patrones de vulnerabilidades encontradas para ataques de Inyección, XSS y Directory Traversal.

Pero al ser más de 100 mil diferentes vulnerabilidades y que no presentan una categorización, se consultó a otras bases de datos que registren el mismo listado de vulnerabilidades y que hayan aplicado algún tipo de categorización, es así que se estudió otras bases de datos como CAPEC, CVE Details, CWE, SANS y OWASP, llegando a la conclusión que todas estas bases de datos hacen referencia a las mismas vulnerabilidades de CVE, mas no todas, sino solo las que son de

utilidad para el ámbito al que pertenecen las organizaciones dueñas de estas bases de datos.

Adicionalmente se investigó sobre un proyecto llamado PHPIDS el cual es un sistema de detección de intrusos para lenguaje PHP y de código abierto, el cual se basa una lista de expresiones regulares que al ser aplicadas al servidor web, actúan como cortafuegos evitando el acceso a las URLS detectadas como vulneradas. (PHPIDS, 2017)

Por lo tanto, se optó por la decodificación de las más importantes expresiones regulares de este proyecto, basado en los tipos de vulnerabilidades y los métodos de acceso que cubren.

Se logró decodificar 75 expresiones regulares obtenidas de este proyecto y se generaron 809 patrones de ataques en total, a lo cual se decidió denominar en adelante las reglas de ataques.

Para la obtención de reglas se utilizó la herramienta existente en el sitio web REGEXPER que permitió transforma una expresión regular en un gráfico.

Las imágenes generadas por REGEXPER se conocen comúnmente como "Diagramas Ferroviarios". Estos diagramas son una manera sencilla de ilustrar lo que a veces puede convertirse en un procesamiento muy complicado en una expresión regular, con bucle anidado y elementos opcionales. La forma más fácil de leer estos diagramas es comenzar por la izquierda y seguir las líneas a la derecha. Si encuentra una bifurcación, existe la opción de seguir una de las múltiples rutas (y esas rutas pueden regresar a partes anteriores del diagrama). Para que una cuerda coincida con éxito con la expresión regular en un diagrama, debe ser capaz de cumplir cada parte del diagrama a medida que se mueve de izquierda a derecha y avanzar por todo el diagrama hasta el final. (Regexper, 2017)

Según el proyecto PHPIDS, la expresión regular número 11 nos ayudará a detectar el acceso a un camino transversal o un directorio específico que generalmente contiene información confidencial.

A continuación, se muestra la Regla 11:

Regla 11

(?:%c0%ae\/)|(?:(?:\/|\\)(home|conf|usr|etc|proc|opt|s?bin|local|dev|tmp|kern|[br]oot|sys|system|windows|winnt|program|%[a-z_-]{3,}%)(?:\/|\\))|(?:(?:\/|\\)inetpub|localstart\.asp|boot\.ini)

En el análisis con REGEXPER se generó un diagrama, del cual en la Figura 27 se reprodujo una fracción.

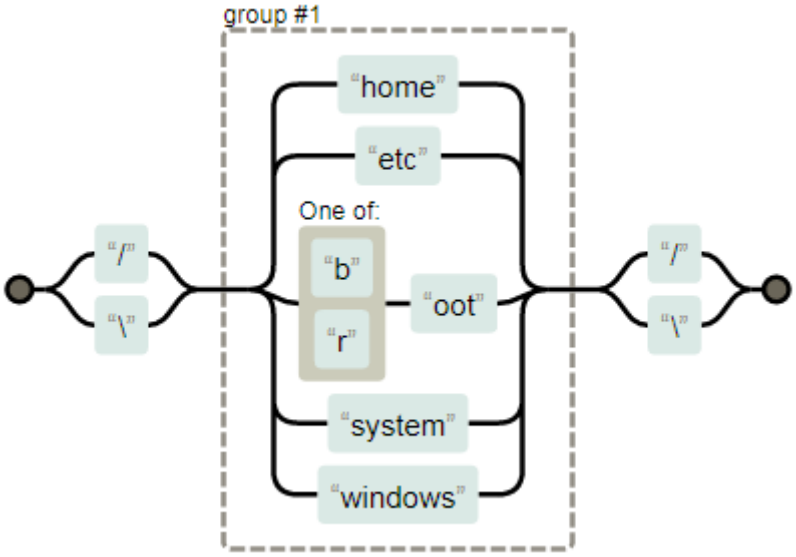


Figura 27: Expresión regular en un diagrama de REGEXPER. Fuente: (Regexper, 2021)

Basado en la Figura 27, se extrajo múltiples combinaciones de patrones de ataques los cuales fueron enviados a un archivo en formato ARFF el cual mediante la herramienta Weka se continuó con la Fase de aplicación de herramienta de análisis de rendimiento de algoritmos explicado posteriormente.

En situaciones donde las expresiones regulares mostraban diagramas confusos que abrían la posibilidad a ser interpretados erróneamente se hizo uso de la herramienta presente en el sitio web DEBUGGEX que nos permitió reforzar las funcionalidades de REGEXPER añadiendo la posibilidad de ingresar datos de prueba para poder comprobar que la regla creada abarcaba a los ejemplos de igual manera como haría con los datos de nuestros archivos log. (DEBUGGEX, 2021)

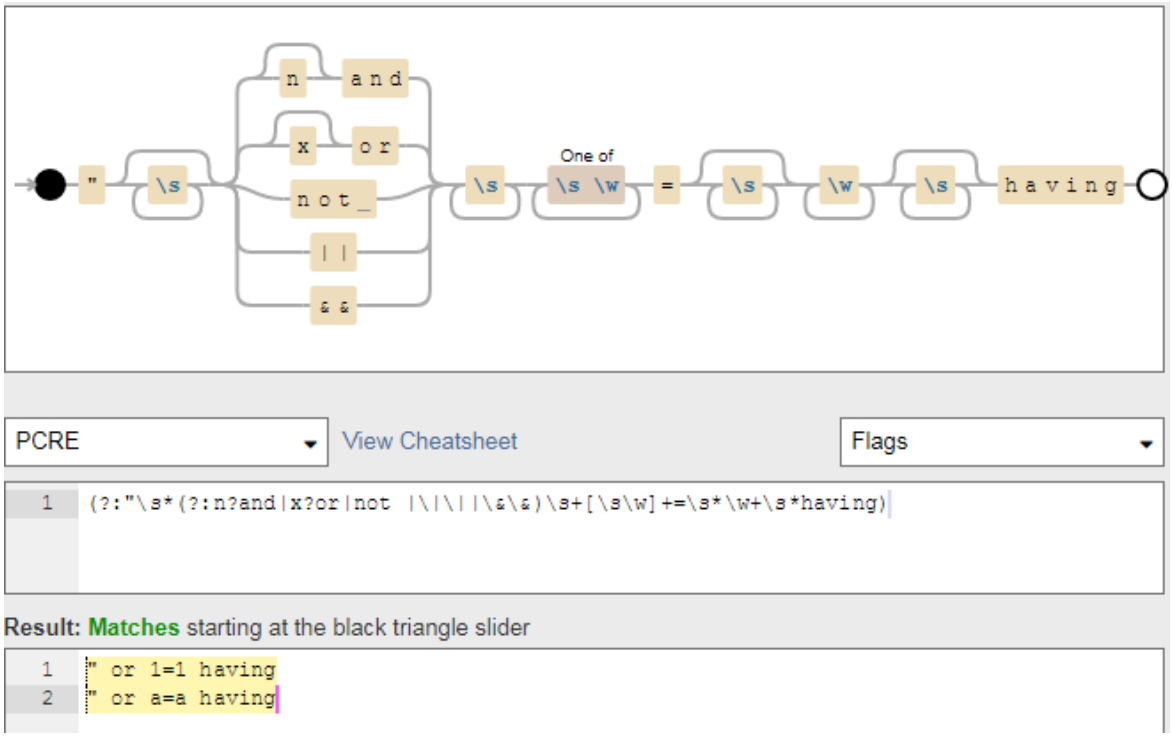


Figura 28: Expresión regular en un diagrama de DEBUGGEX. Fuente: Elaboración propia

Como ejemplo de las reglas extraídas de las expresiones regulares se anexó la Tabla 28 que representa una porción de los 809 patrones extraídos, además de la Figura 29 que muestra una porción del archivo ARFF que se mencionó en el párrafo anterior.

Tabla 28

Muestra de patrones de ataques identificados

Col. 1	Col. 2	Col. 3	Col. 4	Col. 5	Col. 6	Col. 7	Ataque
'\"'	'or'	'%20'	'@'	'='	'@'	'having'	'Injection'
'select'	'%20'	'@'	'from'	?	?	?	'Injection'
'union'	'all'	'%20'	'['	'%20'	'select'	?	'Injection'
'union'	'@'	'%20'	'('	'%20'	'select'	?	'Injection'
'@'	'%20'	'like'	'%20'	'\"'	?	?	'Injection'
'like'	'%20'	'"%'	?	?	?	?	'Injection'
'\"'	'like'	'\"'	?	?	?	?	'Injection'
'\"'	'x'	'or'	'%20'	'='	'@'	'having'	'Injection'
'\"'	'or'	'%20'	'@'	'='	'@'	'having'	'Injection'
'select'	'%20'	'from'	?	?	?	?	'Injection'
'select'	'@'	'from'	?	?	?	?	'Injection'
'update'	'char'	?	?	?	?	?	'Injection'
'create'	'function'	'@'	'returns'	?	?	?	'Injection'
'procedure'	'%20'	'analyse'	'%20'	'('	?	?	'Injection'
'exec'	'('	'%20'	'@'	?	?	?	'Injection'
'exec'	'xp_cmdshell'	?	?	?	?	?	'Injection'
'current_user'	'('	?	?	?	?	?	'Injection'
'database'	'('	?	?	?	?	?	'Injection'
'<'	'%'	'php'	?	?	?	?	'Injection'
'.pl?'	'@'	'='	'@'	'|'	'@'	';'	'Injection'
'@import'	'%%'	?	?	?	?	?	'XSS'
'java'	'.'	'lang'	?	?	?	?	'XSS'
'document.'	'@'	?	?	?	?	?	'XSS'

'%20'	'='	'%20'	'script'	?	?	?	'XSS'
'default'	'xml'	'namespace'	'='	?	?	?	'XSS'
'ip'	'='	'#'	'.'	'#'	?	?	'XSS'
'domain'	'='	'#'	'.'	'#'	?	?	'XSS'
'login'	'='	'#'	'.'	'#'	?	?	'XSS'
'my_addr'	'='	'#'	'.'	'#'	?	?	'XSS'
'\"'	'script'	'('	?	?	?	?	'XSS'
'vbscript'	':'	','	?	?	?	?	'XSS'
'data'	':'	'+'	?	?	?	?	'XSS'
'='	'%20'	'\"'	'%20'	'vbs'	'ript'	':'	'XSS'
'language'	'%20'	'='	'%20'	'vbs'	'ript'	?	'XSS'
'<'	'/'	'script'	?	?	?	?	'XSS'
'<'	'/'	'link'	?	?	?	?	'XSS'
'<'	'/'	'embed'	?	?	?	?	'XSS'
'<'	'/'	'im'	'port'	?	?	?	'XSS'
'<'	'i'	'frame'	?	?	?	?	'XSS'
'/'	'home'	'/'	?	?	?	?	'XSS'
'/'	'conf'	'/'	?	?	?	?	'DT'
'/'	'usr'	'/'	?	?	?	?	'DT'
'/'	'etc'	'/'	?	?	?	?	'DT'
'/'	'tmp'	'/'	?	?	?	?	'DT'
'/'	'root'	'/'	?	?	?	?	'DT'
'/'	'system'	'/'	?	?	?	?	'DT'
'\\'	's'	'bin'	'\\'	?	?	?	'DT'
'\\'	'bin'	'\\'	?	?	?	?	'DT'
'/'	'dev'	'\\'	?	?	?	?	'DT'
'\\'	'inetpub'	?	?	?	?	?	'DT'

Fuente: Elaboración propia.

La Tabla 28 muestra 50 de los 809 patrones de ataques identificados, los cuales para poder ser enviados a la herramienta Weka para la evaluación de rendimiento de los algoritmos seleccionados debieron ser ubicados en una matriz que aceptaba 8 elementos por fila.

Los elementos ubicados desde la columna 1 a la columna 7 fueron caracteres o cadenas de texto, los cuales todos juntos ubicados en una URL y en el mismo orden de aparición representaban un ataque. Y la columna 8 describía a qué tipo de ataque representaban los 7 elementos anteriores. Además, se elaboró una terminología para poder distinguir casos especiales donde algunos caracteres cumplían funciones diferentes a su valor gráfico y cuyos ejemplos se explicaron en los siguientes párrafos.

a) Los caracteres o palabras presentes en cada celda, están encerrados en comillas simples (‘’) para indicar a la herramienta que son cadenas de texto.

b) Algunas celdas contienen solo un signo de interrogación (?) sin comillas, que representa que el valor de esa celda no está definido y puede ser cualquier cadena o ninguna.

c) En alguna fila se puede visualizar dos signos de interrogación juntos (‘??’) esto se interpreta que el elemento de esa celda debe ser un signo de interrogación para diferenciarse de lo descrito en el punto b.

d) Para abreviar algunos patrones se decidió reemplazar cualquier cadena de solo texto por el signo de arroba (‘@’) y cualquier carácter numérico con el signo de numeral (‘#’).

e) En casos en que el elemento debe ser cualquiera de los caracteres descritos en el punto d, se mostró el carácter repetido para diferenciarlo, quedando algo como doble arroba (‘@@’) o doble numeral (‘##’). Similar al caso del punto c.

f) Para algunos caracteres se antepuso la barra invertida (\) debido a que sin esta modificación, la herramienta lo interpretaba como caracteres reservados que cumplen funciones especiales dentro de su lenguaje de programación. Algunos de estos caracteres fueron la comilla simple (‘), comilla doble (“) y la barra invertida (\)

La herramienta Weka admite varios tipos de archivos, pero el formato propio de la aplicación es Attribute-Relation File Format (.arff). Los archivos ARFF tienen formato de texto plano en ASCII, por lo que pueden ser visualizados y modificados desde cualquier editor de texto.

Un archivo ARFF se divide en 2 partes, la cabecera y los datos.
Para escribir comentarios, se debe empezar la línea con %.

Para declarar la relación, atributos y datos, se usa el signo de arroba (@) al comienzo.

En un archivo ARFF se diferencian dos partes: la cabecera y los datos.

Cabecera, contiene la declaración de relación de los atributos que tendrá un archivo, así como el nombre del atributo y el tipo de dato.
De la imagen anterior se tomó la línea siguiente: @relation 'Tipos de ataques', esto representa el nombre de la relación de datos que contiene el archivo.

Luego se define los atributos como se ha definido en la línea siguiente: @attribute Ataque {XSS, Injection, DT}.

El formato para definir un atributo consta de 3 partes que son:
La declaración de un atributo (@attribute), el nombre del atributo (Ataque) y el tipo de dato del atributo ({XSS, Injection, DT}. En este caso el tipo de dato se denomina Nominal pero también hay otros tipos de datos como String, Boolean, Numeric, Real, Integer y Date.

Datos, la declaración de datos comienza con la línea @data y en la siguiente línea se agregan los valores de cada atributo declarado y separado por comas (,).

Los datos separados por espacios deben ir entre comillas simples (‘’) y los caracteres especiales como la barra invertida o la comilla doble (“) deben ir antecedidos por una la barra invertida quedando del siguiente modo: (“) se declarará como (’\”’), (\) se declarará como (‘\\’).

En la Figura 29 se muestra una fracción del archivo patrones.arff elaborado en esta investigación.

patrones.arff

```
%Ataques
@relation 'Tipos de ataques'

%ATRIBUTOS POR ORDEN DE APARICIÓN EN LA DESCRIPCIÓN
@attribute Cadena1 {'\\x','%','&#','\\','&#x','union','@','like','\"','s
@attribute Cadena2 {'0','1','f','e','%20','all','distinct','(','!','@','
@attribute Cadena3 {'#','b','c','e','f','all','distinct','(','!','@','%2
@attribute Cadena4 {'%20','(','[','select','\"','#','=','@','from','retu
@attribute Cadena5 {'(','[','%20','\"','=','@','-','|',';','>','lang',')
@attribute Cadena6 {'%20','select','@','having',')',';','>','=',':','rip
@attribute Cadena7 {'select','having','-',';','=','/',':','vbs','ript','
@attribute Ataque {XSS,Injection,DT}

%Patrones de ataques
@data
'\"',   'x',    'or',   '%20',  '=',    '@',    'having',   'Injection'
'\"',   'or',   '%20',  '=',    '@',    'having',   ?,      'Injection'
'\"',   'or',   '%20',  '@',    '=',    '@',    'having',   'Injection'
'\"',   'not',  '%20',  '=',    '@',    'having',   ?,      'Injection'
```

Figura 29: Fracción de un archivo ".arff". Fuente: Elaboración propia

Con el archivo ARFF terminado se procedió a la ejecución de la fase de aplicación de herramienta de análisis de rendimiento de algoritmos (Sección 3.3.1.5), pero antes se explicó cómo se obtuvo los algoritmos a analizar, esto en la Fase de elección de algoritmos de aprendizaje automático (Sección 3.3.1.4).

3.3.1.4. Fase de elección de algoritmos de aprendizaje automático

Para la elección de los algoritmos de aprendizaje automático a los cuales se le enviará el archivo ARFF elaborado en la sección 3.3.1.3 se elaboró una tabla que representaba los principales papers descritos en los antecedentes de la investigación, seguido de un listado con los algoritmos que mejor rendimiento ofrecieron en sus respectivas investigaciones, así como se muestra en la Tabla 29.

Tabla 29

Clasificación de rendimiento de algoritmos empleados en los papers de antecedentes.

N°	Título	Año	Algoritmo	Precisión
3	A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities	2019	**SVM**	**98,60%**

10	Detecting Web Attacks Using Multi-Stage Log Analysis	2016	**Bayes Net**	**95,40%**
8	Neural Network Based Web Log Analysis for Web Intrusion Detection	2017	Decision Tree	93,00%
7	Machine Learning to Detect Anomalies in Web Log Analysis	2017	SVM	90,32%
9	Classification of SQL injection, XSS and Path Traversal for Web Application Attack Detection	2016	**Random Forest**	**86,62%**

Nota: La columna “N°” representa la ubicación del antecedente por orden de aparición en esta investigación. La tabla esta ordenada descendentemente según la columna “Precisión”. Fuente: Elaboración propia.

La Tabla 29 muestra un listado con los proyectos de investigación más resaltantes referenciados en la presente investigación, de los cuales se describe el nombre del proyecto, año de publicación, autor, algoritmo utilizado que presentó la mejor precisión y el porcentaje de precisión que obtuvo el algoritmo en orden descendente.

Los algoritmos identificados fueron los siguientes:

1. SVM
2. Bayes Net
3. Decision Tree
4. Random Forest

De los cuales se eligieron 3 algoritmos en base a las siguientes conclusiones.

SVM, se eligió como uno de los algoritmos a evaluar en base a sus resultados obtenidos en las investigaciones de Kevin Zhang y Qimin Cao en 2019 y 2017 respectivamente.

BayesNet, se eligió como uno de los algoritmos a evaluar con base a sus resultados obtenidos en la investigación de Melody Moh en 2016.

Random Forest, se eligió como uno de los algoritmos a evaluar con base a sus resultados obtenidos en la investigación de Ei Ei Han en 2016.

Decision Tree, se optó por descartar este algoritmo por elección personal para poder evaluar el desempeño de otro algoritmo de árboles de decisión, Random Forest.

En conclusión, se decidió elegir como algoritmos a evaluar en la investigación al algoritmo de Máquinas de vectores de soporte (SVM), el algoritmo de Árboles de decisión Random Forest y el algoritmo basado en el teorema de Bayes, Bayes Net.

3.3.1.5. Fase de aplicación de herramienta de análisis de rendimiento de algoritmos

En esta fase se realizaron las pruebas para evaluar el rendimiento de los algoritmos SVM, Random Forest y BayesNet con el archivo ARFF generado en la fase de identificación de patrones de ataque.

A los algoritmos se le evaluó el rendimiento según 3 indicadores que fueron la exactitud, precisión y recall.

Rendimiento

Para evaluar el rendimiento de los algoritmos se utilizó la herramienta llamada Matriz de Confusión que permite evaluar el desempeño de los algoritmos en un aprendizaje supervisado. (Kohavi & Provost, 1998)

Una matriz de confusión se representa según la Tabla 47 que se presenta a continuación.

Tabla 30

Representación de una matriz de confusión según Kohavi & Provost.

Matriz de confusión		VALOR DE PREDICCION	
		SI	NO
VALOR REAL	SI	Verdadero Positivo (VP)	Falso Negativo (FN)
	NO	Falso Positivo (FP)	Verdadero Negativo (VN)

Fuente: (Kohavi & Provost, 1998)

Las entradas en la matriz de confusión tienen el siguiente significado en el contexto de nuestro estudio:

Tabla 31

Interpretación de los valores generado por la matriz de confusión.

Sigla	Descripción
VP	Cantidad de aciertos para la clase negativa.
FP	Cantidad de errores para la clase positiva.
FN	Cantidad de errores para la clase positiva.
VN	Cantidad de aciertos para la clase negativa.

Fuente: Elaboración propia.

Basado en la tabla anterior se evaluó distintas medidas de rendimiento que pueden ser aplicadas a los algoritmos y que se explican a continuación.

Exactitud

Representa la proporción del número total de predicciones correctas.
En este análisis se obtiene dos tipos de exactitud, la exactitud total y la exactitud por clase elegida y las fórmulas son las siguientes:

Exactitud Total = VP + VN / VP + FP + FN + VN
Exactitud por Clase = VP / VP + FN
La exactitud por clase suele ser la misma que la Tasa de recuerdo

Precisión

Es la proporción de casos positivos predecidos que fueron correctos mediante la ecuación:

Precisión = VP / VP + FP

Tasa de recuerdo (Recall)

También llamada de verdadero positivo (VP) es la proporción de casos positivos que fueron identificados correctamente utilizando la ecuación:
Recall = VP / VP + FN

Para representar en un ejemplo cómo se consiguió los indicadores descritos anteriormente, se procedió a utilizar Weka como herramienta de análisis.

WEKA incluye varios algoritmos de aprendizaje automático. Los algoritmos pueden invocarse desde el código Java propio del usuario o aplicarse directamente al conjunto de datos listo. WEKA contiene herramientas de entorno de propósito general para pre procesamiento de datos, regresión, clasificación, reglas de asociación, agrupamiento, selección de características y visualización. (Dietterich, 2002)

Por lo tanto, se sometió a análisis el archivo Patrones.arff mediante la herramienta Weka, lo que reportó un resultado como se muestra en la Figura 30.

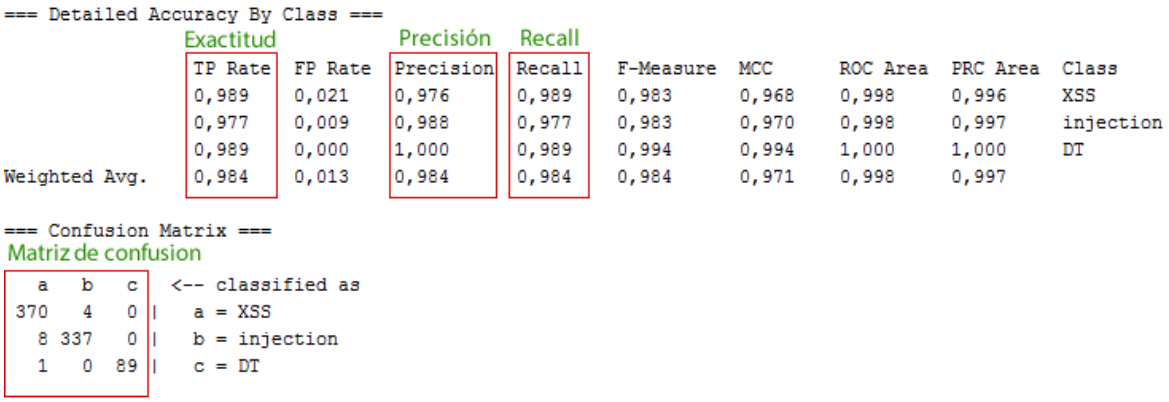


Figura 30: Resultado de evaluación de un algoritmo mediante Weka. Fuente: Elaboración propia

Entonces se procedió a desglosar el reporte para detallar la matriz de confusión para cada tipo de ataque clasificado.

Resultado

Tabla 32

Matriz de confusión según tipos de ataques.

Matriz de confusión		VALOR DE PREDICCIÓN			
		XSS	Injection	DT	Total
VALOR REAL	XSS (374)	370	4	0	374
	Injection (345)	8	337	0	345
	DT (90)	1	0	89	90
Total	809	379	341	89	809

Fuente: Elaboración propia

Matriz de confusión para XSS

Tabla 33

Matriz de confusión para XSS

XSS		VALOR DE PREDICCION	
		SI	NO
VALOR REAL	SI	370 (VP)	4 (FN)
	NO	9 (FP)	426 (VN)

Fuente: Elaboración propia

Exactitud Total

Fórmula = VP + VN / VP + FP + FN + VN

= (370 + 426) / (370 + 9 + 4 + 426)
= 0,98393

Precisión

Fórmula = VP / VP + FP

= 370 / (370 + 9)
= 0,97625

Recall

Fórmula = VP / VP + FN

= 370 / (370 + 4)
= 0,98930

Matriz de confusión para Injection

Tabla 34

Matriz de confusión para Injection

Injection		VALOR DE PREDICCION	
		SI	NO
VALOR REAL	SI	337 (VP)	8 (FN)
	NO	4 (FP)	459 (VN)

Fuente: Elaboración propia

Exactitud Total

Fórmula = VP + VN / VP + FP + FN + VN

= (370 + 426) / (370 + 9 + 4 + 426)
= 0,98393

Precisión

Fórmula = VP / VP + FP

= 337 / (337 + 4)

= 0,98827

Recall

Fórmula = VP / VP + FN

= 337 / (337 + 8)

= 0,97681

Matriz de confusión para Directory Traversal

Tabla 35

Matriz de confusión para Directory Traversal

DT		VALOR DE PREDICCION	
		SI	NO
VALOR REAL	SI	89 (VP)	1 (FN)
	NO	0 (FP)	719 (VN)

Exactitud Total

Fórmula = VP + VN / VP + FP + FN + VN

= (370 + 426) / (370 + 9 + 4 + 426)

= 0,98393

Precisión

Fórmula = VP / VP + FP

= 89 / (89 + 0)

= 1.0000

Recall

Fórmula = VP / VP + FN

= 89 / (89 + 1)

= 0,98888

De esta manera se ha comprobado cómo es que la herramienta Weka calcula el rendimiento de los algoritmos. Además de los indicadores ya calculados también se tomará en cuenta en los informes a otros indicadores proporcionados por Weka, los cuales son el número de instancias, el número y porcentaje de instancias clasificadas correcta e incorrectamente además del tiempo necesario para construir el modelo y el tiempo necesario para probar el modelo en los datos de entrenamiento, información que será guardada en una ficha de observación como la de la Figura 31.

Ficha de observación de Análisis de rendimiento del algoritmo J48

Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Training Test
Numero de Instancias	809
# Instancias procesadas correctamente	796
Tiempo de Construccion	0,03
Tiempo de entrenamiento	0,01
Exactitud Total	0,984
Precisión Total	0,984
Recall Total	0,984

Matriz de confusion			
Clasificación	**a**	**b**	**c**
a = XSS	370	4	0
b = Injection	8	337	0
c = DT	1	0	89

Evaluación de rendimiento			
Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	0,989	0,976	0,989
Injection	0,977	0,988	0,977
DT	0,989	1,000	0,989

Figura 31: Ficha de observación de análisis de rendimiento de un algoritmo. Fuente: Elaboración propia

La Figura 31 muestra el análisis realizado al algoritmo Random Forest, ese análisis se realizó con un tipo de prueba denominado Training Test, que consiste en hacer el test de prueba con el mismo conjunto de datos de entrenamiento.

Weka permite realizar 4 pruebas diferentes que son las siguientes:

Training Set, consiste en hacer el test de prueba con el mismo conjunto de datos de entrenamiento.

Supplied test set, Consiste en utilizar un archivo de datos distinto al de entrenamiento para hacer el test de prueba.

Cross Validation, En este tipo de prueba se calcula el porcentaje de aciertos esperado haciendo una validación cruzada de “n” hojas, el número predeterminado de hojas es 10.

Percentage split: Esta prueba consiste en dividir el conjunto de datos de entrenamiento y dividirlo en dos partes, la primera para construir el clasificador y la segunda para hacer la prueba. El porcentaje para entrenamiento por omisión, es de 66%.

Según el tipo de prueba elegido, el resultado del análisis del rendimiento de los algoritmos presentó variaciones considerables, variaciones que se explican con ejemplos en el capítulo de resultados.

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones.

a) Para su posterior análisis de archivos log en la búsqueda de patrones de ataques se obtuvieron 60 archivos log de un servidor web apache con tamaños entre los 26,77 y 42,87 megabytes (Mb) y que arrojaron un total de 1981,25 Mb equivalente a 1,93 gigabytes (Gb). Porque, para que, importancia?

b) Se realizó el preprocesamiento de los 60 archivos logs para eliminar la data sin valor y solo mantener la que sería útil a la investigación, obteniendo 60 nuevos archivos de entre 45 y 4152 URLs y un tamaño total 1.47 Mb.

c) Mediante el análisis de los archivos logs y de un conjunto de reglas de prevención de ataques basadas en expresiones regulares, se obtuvieron 809 patrones de ataques de los cuales 374 fueron de XSS, 345 de Inyección y 90 de Directory Traversal.

d) Se revisó 10 investigaciones previas, de las cuales se identificó que emplearon 10 algoritmos diferentes, de los cuales se eligió los algoritmos BayesNet, Random Forest y SVM para su evaluación de rendimiento.

e) Para la obtención de los resultados de la evaluación del rendimiento de los algoritmos seleccionados en esta investigación, la herramienta Weka mostró a Random Forest como el algoritmo de mejor rendimiento en las pruebas de Training Set, con una exactitud de 1.000 que equivale a 100%, superando a SVM y Bayes Net que mostraron un rendimiento de 99,9% y 98,9%, una diferencia de 0,1% y 1,1% respectivamente.

4.2. Recomendaciones.

a) Se recomienda fraccionar los archivos logs de servidores compartidos o que tengan tráfico superior a 500 mil solicitudes HTPP por archivo log, pues al desear visualizar el archivo con algún editor incorporado con el sistema operativo Windows, el editor podría dejar de funcionar.

b) Los archivos logs tienen un formato de registro de datos, pero agentes malintencionados pueden manipular las solicitudes induciendo al servidor a

omitir campos obligatorios como la versión del protocolo o el código HTTP lo cual produciría un error de procesamiento, por lo tanto, se recomienda, no asumir que todos los campos son obligatorios.

c) Se obtuvo 809 patrones de ataques, pero se puede obtener muchos más si se emplea reglas de otros proyectos o de IDS conocidos.

d) Diversos trabajos han evaluado el rendimiento de distintos algoritmos y los que mejor respondían para un trabajo no ofrecían los mismos resultados en otro, es por eso que se recomienda no asumir que un algoritmo es mejor a otro por los resultados que obtuvo en una investigación en particular pues cada investigación utilizo distintas técnicas y buscaba resultados diferentes.

e) Se recomienda que los atributos del archivo ARFF sean de tipo nominal o al menos que no sea String puesto que una mínima cantidad de algoritmos presentes en Weka se pueden utilizar para la evaluación y para la presente investigación solo Bayes Net ofreció resultados confiables en el análisis con datos de tipo String.

REFERENCIAS.

Apache. (2013). Servidor HTTP Apache. Obtenido de Archivos de Registro (Log Files): https://httpd.apache.org/docs/2.0/es/logs.html

Arumugam, C., Dwarakanathan, V. B., Gnanamary, S., Neyveli, V. N., Ramesh, R. K., Kandhavel, Y., & Balakrishnan, S. (2019). *Prediction of SQL Injection Attacks in Web Applications* (pp. 496–505). https://doi.org/10.1007/978-3-030-24305-0_37

AZIZI, Y., AZIZI, M., & ELBOUKHARI, M. (2019). Log files Analysis Using MapReduce to Improve Security. *Procedia Computer Science*, *148*, 37–44. https://doi.org/10.1016/j.procs.2019.01.006

Belshe, M., Thomson, M., & Peon, R. (2015). Hypertext transfer protocol version 2. https://tools.ietf.org/html/draft-ietf-httpbis-http2-01. Diakses 3 Desember 2015., 1-67.

Berners-Lee, T., Fielding, R., Irvine, U. C., & Masinter, L. (1998). Uniform Resource Identifiers (URI): Generic Syntax. Rfc, 2396, 40. https://doi.org/10.1017/CBO9781107415324.004

Berners-Lee, T., MIT/LCS, Fielding, R., University of California Irvine, & Nielsen, H. F. (1996). RFC 1945: Hypertext Transfer Protocol -- HTTP/1.0. Internet Engineering Task Force, 1-60. https://doi.org/10.1017/CBO9781107415324.004

Banco Mundial. (2021, December 13). *Población, total*. Https://Datos.Bancomundial.Org/Indicator/SP.POP.TOTL?End=2020&start=2020&view=bar.

Baş Seyyar, M., Çatak, F. Ö., & Gül, E. (2017). Detection of attack-targeted scans from the Apache HTTP Server access logs. *Applied Computing and Informatics*. https://doi.org/10.1016/j.aci.2017.04.002

Cao, Q., Qiao, Y., & Lyu, Z. (2017). Machine learning to detect anomalies in web log analysis. *2017 3rd IEEE International Conference on Computer and*

Communications (ICCC), 519–523. https://doi.org/10.1109/CompComm.2017.8322600

Carmona Suárez, E. (2014). Tutorial sobre Máquinas de Vectores Soporte (SVM). España.

DEBUGGEX. (octubre de 2017). DEBUGGEX. Obtenido de https://www.debuggex.com

Dietterich, T. (2002). Ensemble Learning (segunda ed.). USA: Cambridge.

Elhebir, M. H. A., & Abraham, A. (2015). A Novel Ensemble Approach to Enhance the Performance of Web Server Logs Classification, 7, 189-195.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). RFC 2616: Hypertext transfer protocol–HTTP/1.1, June 1999. Network Working Group Request for Comments, 2616(11). Recuperado a partir de http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:RFC+2616#0

Gao, Y., Ma, Y., & Li, D. (2017). Anomaly detection of malicious users' behaviors for web applications based on web logs. *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, 1352–1355. https://doi.org/10.1109/ICCT.2017.8359854

Han, E. E., & Phyu, T. N. (2016). Classification of SQL injection, XSS and Path Traversal for Web Application Attack Detection. In *Classification of SQL injection, XSS and Path Traversal for Web Application Attack Detection*. https://onlineresource.ucsy.edu.mm/bitstream/handle/123456789/309/201640.pdf?sequence=1&isAllowed=y

Internetworldstats. (2021, March 31). *World Internet Users and 2021 Population Stats*. Https://Www.Internetworldstats.Com/Stats.Htm.

Kohavi, R., & Provost, F. (1998). Confusion matrix. Machine learning (Vol. 30).

Kohavi, R., & Quinlan, R. (1999). Decision Tree Discovery. Recuperado el 6 de noviembre de 2017, de http://ai.stanford.edu/~ronnyk/treesHB.pdf

Latib, M. A., Ismail, S. A., Yusop, O. M., Magalingam, P., & Azmi, A. (2018). Analysing Log Files For Web Intrusion Investigation Using Hadoop. *Proceedings of the 7th International Conference on Software and Information Engineering - ICSIE '18*, 12–21. https://doi.org/10.1145/3220267.3220269

Lenguaje I. (2017). Recuperado el 7 de noviembre de 2017, de Arquitectura de las aplicaciones Web: https://sites.google.com/site/sitiodenystovar/arquitectura-de-las-aplicaciones-web

Ma, K., Jiang, R., Dong, M., Jia, Y., & Li, A. (2017). *Neural Network Based Web Log Analysis for Web Intrusion Detection* (pp. 194–204). https://doi.org/10.1007/978-3-319-72395-2_19

Mateos Ferré, R. (s.f.). SISTEMA DE ANÁLISIS DE DATOS MÉDICOS SEGÚN ITINERARIOS Y EVOLUCIONES. Universidad Rovira i Virgili, España. Recuperado el 6 de noviembre de 2017, de http://deim.urv.cat/~pfc/docs/pfc339/d1117431499.pdf

Moh, M., Pininti, S., Doddapaneni, S., & Moh, T.-S. (2016). Detecting Web Attacks Using Multi-stage Log Analysis. *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, 733–738. https://doi.org/10.1109/IACC.2016.141

Netcraft. (octubre de 2021). *October 2021 Web Server Survey*. Https://News.Netcraft.Com/Archives/2021/10/15/October-2021-Web-Server-Survey.Html.

OWASP. (2021). *OWASP Top 10:2021*. https://owasp.org/Top10/

OWASP (2017). OWASP Top 10 Application Security Risks - 2017.

PHPIDS. (octubre de 2017). PHPIDS. Obtenido de https://github.com/PHPIDS/PHPIDS

Platt, J. (1998). Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Microsoft.

Romero Laguillo, L. F. (1997). Publicar en Internet: guía práctica para la creación de documentos HTML. Universidad de Cantabria. Recuperado el 10 de Julio de 2017, de https://books.google.com/books?isbn=8481021792

Stevens, W., Fenner , B., & Rudoff , A. (2004). Unix Network Programming, Volume 1: The Sockets Networking API (3 ed., Vol. volume 1). Addison-Wesley Professional.

Tarlogic. (octubre de 2017). Tarlogic. Obtenido de Auditoria de seguridad web OWASP: https://www.tarlogic.com/servicios/auditoria-web-seguridad-owasp/

Tribak, H. (2012). Análisis Estadístico de Distintas Técnicas de Inteligencia Artificial en Detección de Intrusos.

Zhang, K. (2019). A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities. *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 1286–1288. https://doi.org/10.1109/ASE.2019.00164

ANEXOS.

Anexo 1. Resolución de aprobación / ampliación del proyecto de investigación

USS | UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N°1201-2021/FIAU-USS

Pimentel, 27 de diciembre 2021

VISTO:

El Acta de reunión N°1312 - 2021, remitida mediante oficio N° 0395-2021/FIAU-IS-USS de fecha 14 de diciembre de 2021 por la Escuela profesional de INGENIERÍA DE SISTEMAS, para la ejecución de la Tesis: **"EVALUACIÓN DE RENDIMIENTO DE ALGORITMOS EN LA IDENTIFICACIÓN DE ATAQUES A SITIOS WEB UTILIZANDO LOGS DE SERVIDOR"**, presentado por **CHINGUEL TINEO SEGUNDO FLORENTINO**, del Programa de estudios de **INGENIERÍA DE SISTEMAS**, y;

CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48º que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y *tesis* son *aprobados por el Comité de Investigación* y derivados a la facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El *periodo de vigencia de los mismos será de dos años*, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24º señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; *es individual o en pares para obtener un título profesional*. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, mediante documentos de vistos, el Comité de investigación de la referida Escuela profesional acordó aprobar la ampliación de la vigencia del Proyecto de tesis denominado **"EVALUACIÓN DE RENDIMIENTO DE ALGORITMOS EN LA IDENTIFICACIÓN DE ATAQUES A SITIOS WEB UTILIZANDO LOGS DE SERVIDOR"**, aprobada mediante Resolución de Facultad a cargo de **CHINGUEL TINEO SEGUNDO FLORENTINO**, en condición de egresado, del Programa de estudios de **INGENIERÍA DE SISTEMAS**.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO ÚNICO: AMPLIAR VIGENCIA, de la Tesis denominada **"EVALUACIÓN DE RENDIMIENTO DE ALGORITMOS EN LA IDENTIFICACIÓN DE ATAQUES A SITIOS WEB UTILIZANDO LOGS DE SERVIDOR"**, a cargo de **CHINGUEL TINEO SEGUNDO FLORENTINO**, del Programa de estudios **INGENIERÍA DE SISTEMAS,** hasta el **13 de diciembre de 2022.**

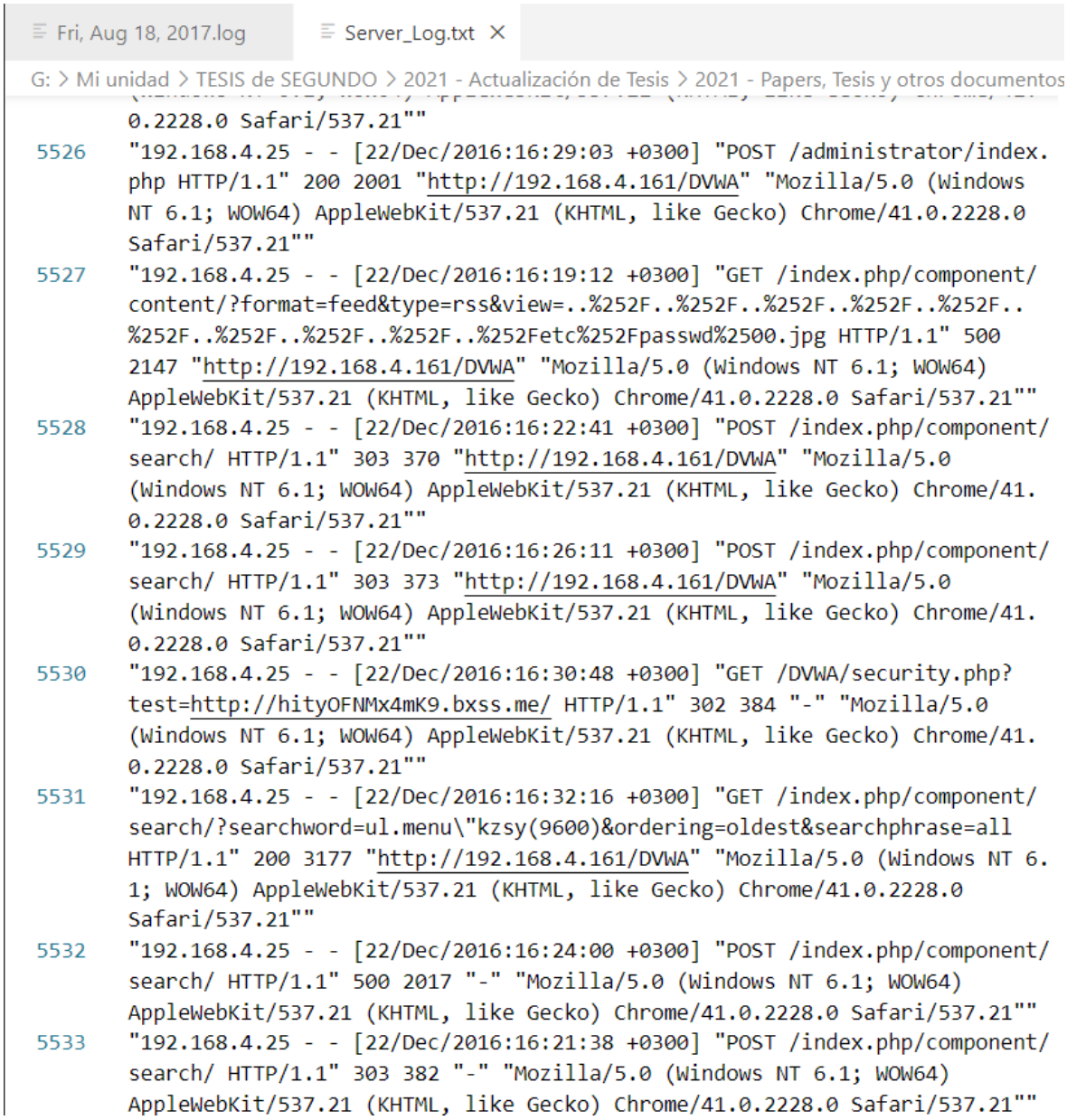
REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Mg. Victor Alexei Tuesta Montero
Decano (e) / Facultad De Ingeniería, Arquitectura Y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

MBA. María Noelia Sialer Rivera
Secretaria Académica / Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

cc: Interesado, Archivo

Anexo 2. Ficha de observación del contenido un archivo LOG de servidor web Apache



Fri, Aug 18, 2017.log | Server_Log.txt

G: > Mi unidad > TESIS de SEGUNDO > 2021 - Actualización de Tesis > 2021 - Papers, Tesis y otros documentos

```
        0.2228.0 Safari/537.21""
5526    "192.168.4.25 - - [22/Dec/2016:16:29:03 +0300] "POST /administrator/index.
        php HTTP/1.1" 200 2001 "http://192.168.4.161/DVWA" "Mozilla/5.0 (Windows
        NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.0.2228.0
        Safari/537.21""
5527    "192.168.4.25 - - [22/Dec/2016:16:19:12 +0300] "GET /index.php/component/
        content/?format=feed&type=rss&view=..%252F..%252F..%252F..%252F..%252F..
        %252F..%252F..%252F..%252F..%252Fetc%252Fpasswd%2500.jpg HTTP/1.1" 500
        2147 "http://192.168.4.161/DVWA" "Mozilla/5.0 (Windows NT 6.1; WOW64)
        AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.21""
5528    "192.168.4.25 - - [22/Dec/2016:16:22:41 +0300] "POST /index.php/component/
        search/ HTTP/1.1" 303 370 "http://192.168.4.161/DVWA" "Mozilla/5.0
        (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.
        0.2228.0 Safari/537.21""
5529    "192.168.4.25 - - [22/Dec/2016:16:26:11 +0300] "POST /index.php/component/
        search/ HTTP/1.1" 303 373 "http://192.168.4.161/DVWA" "Mozilla/5.0
        (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.
        0.2228.0 Safari/537.21""
5530    "192.168.4.25 - - [22/Dec/2016:16:30:48 +0300] "GET /DVWA/security.php?
        test=http://hityOFNMx4mK9.bxss.me/ HTTP/1.1" 302 384 "-" "Mozilla/5.0
        (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.
        0.2228.0 Safari/537.21""
5531    "192.168.4.25 - - [22/Dec/2016:16:32:16 +0300] "GET /index.php/component/
        search/?searchword=ul.menu\"kzsy(9600)&ordering=oldest&searchphrase=all
        HTTP/1.1" 200 3177 "http://192.168.4.161/DVWA" "Mozilla/5.0 (Windows NT 6.
        1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.0.2228.0
        Safari/537.21""
5532    "192.168.4.25 - - [22/Dec/2016:16:24:00 +0300] "POST /index.php/component/
        search/ HTTP/1.1" 500 2017 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64)
        AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.21""
5533    "192.168.4.25 - - [22/Dec/2016:16:21:38 +0300] "POST /index.php/component/
        search/ HTTP/1.1" 303 382 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64)
        AppleWebKit/537.21 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.21""
```

Anexo 3. Ficha de observación de los archivos logs obtenidos del servidor web apache.

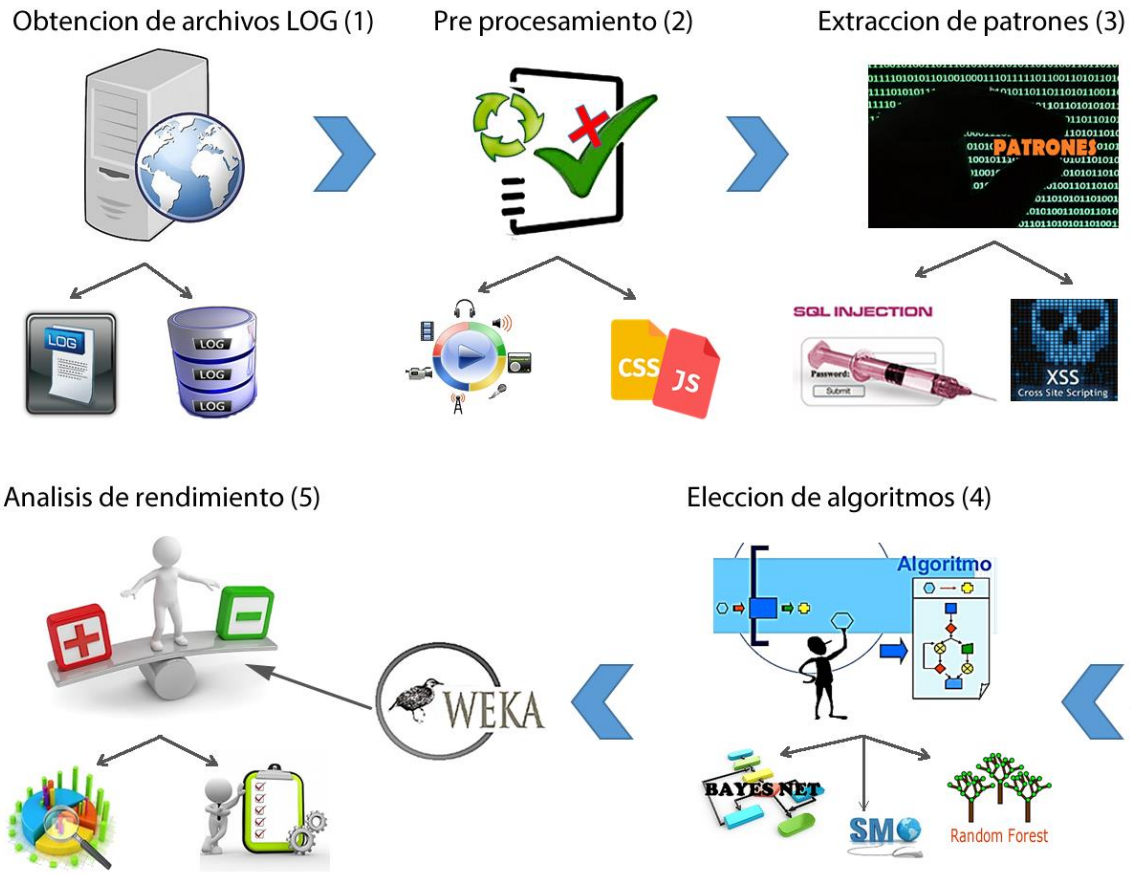
Ficha de observación de archivos logs descargados del sitio LetrasHQ.com

Fecha de elaboración:	03/02/2022
Numero de archivos:	60
Tamaño mínimo:	26.77
Tamaño máximo:	42.87
Tamaño total (Mb):	1981.25

Archivos Log del sitio LetrasHQ.com			
ID	**Archivo**	**Tamaño (Mb)**	**Fecha**
1	Sat, Aug 12, 2017.log	42.87	12/08/2017
2	Sun, Aug 13, 2017.log	35.75	13/08/2017
3	Mon, Aug 14, 2017.log	32.44	14/08/2017
4	Tue, Aug 15, 2017.log	33.41	15/08/2017
5	Wed, Aug 16, 2017.log	34.96	16/08/2017
6	Thu, Aug 17, 2017.log	34.98	17/08/2017
7	Fri, Aug 18, 2017.log	32.29	18/08/2017
8	Sat, Aug 19, 2017.log	34.58	19/08/2017
9	Sun, Aug 20, 2017.log	33.95	20/08/2017
10	Mon, Aug 21, 2017.log	27.28	21/08/2017
11	Tue, Aug 22, 2017.log	28.84	22/08/2017
12	Wed, Aug 23, 2017.log	30.43	23/08/2017
13	Thu, Aug 24, 2017.log	32.69	24/08/2017
14	Fri, Aug 25, 2017.log	31.63	25/08/2017
15	Sat, Aug 26, 2017.log	33.46	26/08/2017
16	Sun, Aug 27, 2017.log	30.04	27/08/2017
17	Mon, Aug 28, 2017.log	28.47	28/08/2017
18	Tue, Aug 29, 2017.log	26.77	29/08/2017
19	Wed, Aug 30, 2017.log	28.40	30/08/2017
20	Thu, Aug 31, 2017.log	28.51	31/08/2017
21	Fri, Sep 1, 2017.log	29.65	1/09/2017
22	Sat, Sep 2, 2017.log	33.03	2/09/2017
23	Sun, Sep 3, 2017.log	34.04	3/09/2017
24	Mon, Sep 4, 2017.log	29.61	4/09/2017
25	Tue, Sep 5, 2017.log	28.42	5/09/2017

26	Wed, Sep 6, 2017.log	34.66	6/09/2017
27	Thu, Sep 7, 2017.log	34.95	7/09/2017
28	Fri, Sep 8, 2017.log	34.90	8/09/2017
29	Sat, Sep 9, 2017.log	36.45	9/09/2017
58	Sun, Sep 10, 2017.log	32.00	10/09/2017
30	Mon, Sep 11, 2017.log	28.88	11/09/2017
31	Tue, Sep 12, 2017.log	29.82	12/09/2017
32	Wed, Sep 13, 2017.log	29.82	13/09/2017
33	Fri, Sep 15, 2017.log	30.66	15/09/2017
34	Sat, Sep 16, 2017.log	33.93	16/09/2017
35	Sun, Sep 17, 2017.log	30.29	17/09/2017
36	Mon, Sep 18, 2017.log	31.00	18/09/2017
37	Tue, Sep 19, 2017.log	31.31	19/09/2017
38	Wed, Sep 20, 2017.log	32.37	20/09/2017
39	Thu, Sep 21, 2017.log	34.03	21/09/2017
40	Fri, Sep 22, 2017.log	36.46	22/09/2017
41	Sat, Sep 23, 2017.log	37.02	23/09/2017
42	Sun, Sep 24, 2017.log	36.54	24/09/2017
43	Mon, Sep 25, 2017.log	31.84	25/09/2017
44	Tue, Sep 26, 2017.log	33.49	26/09/2017
45	Wed, Sep 27, 2017.log	35.08	27/09/2017
46	Thu, Sep 28, 2017.log	33.94	28/09/2017
47	Fri, Sep 29, 2017.log	36.11	29/09/2017
48	Sat, Sep 30, 2017.log	40.16	30/09/2017
49	Sun, Oct 1, 2017.log	37.19	1/10/2017
50	Mon, Oct 2, 2017.log	37.43	2/10/2017
51	Tue, Oct 3, 2017.log	36.15	3/10/2017
52	Wed, Oct 4, 2017.log	36.31	4/10/2017
53	Thu, Oct 5, 2017.log	33.77	5/10/2017
54	Fri, Oct 6, 2017.log	36.91	6/10/2017
55	Sat, Oct 7, 2017.log	36.77	7/10/2017
56	Sun, Oct 8, 2017.log	34.05	8/10/2017
57	Mon, Oct 9, 2017.log	31.52	9/10/2017
59	Tue, Oct 10, 2017.log	28.90	10/10/2017
60	Wed, Oct 11, 2017.log	30.04	11/10/2017

Anexo 4. Diagrama de las etapas que comprendió el desarrollo de la investigación.



Anexo 5. Ficha de observación de análisis de rendimiento de un algoritmo evaluado.

Ficha de observación de Análisis de rendimiento del algoritmo Random Forest

Resumen de análisis	
Numero de Análisis	1
Tipo prueba	Training set
Numero de Instancias	809
# Instancias procesadas correctamente	809
Tiempo de Construcción	0,26
Tiempo de entrenamiento	0,08
Exactitud Total	1,000
Precisión Total	1,000
Recall Total	1,000

Matriz de confusión

Clasificación	**a**	**b**	**c**
a = XSS	374	0	0
b = Iniection	0	345	0
c = DT	0	0	90

Evaluación de rendimiento

Ataque	**Exactitud**	**Precisión**	**Recall**
XSS	1,000	1,000	1,000
Iniection	1,000	1,000	1,000
DT	1,000	1,000	1,000