SISTEMA PARA EL ESTUDIO DE CIBERATAQUES WEB

PROYECTO DE GRADO

BRAYAN ANDRÉS HENAO

JUAN SEBASTIÁN PRADA

Tutores del proyecto

CHRISTIAN CAMILO URCUQUI, MSC

ANDRÉS NAVARRO CADAVID, PhD

UNIVERSIDAD ICESI
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS

SANTIAGO DE CALI
MAYO, 2018

TABLA DE CONTENIDO

RESUMEN

La seguridad es un problema de una alta complejidad por lo tanto lo que se busca es reducir la brecha entre lo seguro y lo inseguro, para ello se han utilizado diferentes técnicas para aproximarse a una solución. Recientemente se ha involucrado el uso de *machine learning* para este tipo de tareas por lo cual nuestro proyecto busca generar un método de clasificación basado en éste. Nuestro proyecto propone un mecanismo para la generación y el estudio de las características que permiten el entrenamiento de modelos que cuentan con la capacidad de detectar ciberataques de tipo *Cross-site Scripting (XSS)* y *Structured Query Language Injection (SQLI)*, dos tipos de ciberataques asociados a diferentes riesgos informáticos basados en el top 10 del *Open Web Application Security Project 2017 (OWASP)*. Para la obtención de la información se realizó un script propio para la simulación de ataques informáticos en un ambiente vulnerable controlado el cual consta de un servidor con dos máquinas virtuales, una equipada con una aplicación vulnerable y una herramienta para la captura de tráfico; la otra consta de un navegador web y un script para la automatización del acceso a la aplicación. Cada ataque se realizó con una base de información de ataques conocidos de XSS y SQLI obtenidos de diferentes fuentes. Posteriormente se realizó un entrenamiento de cuatro algoritmos de *machine learning* y su evaluación, además, se probó la capacidad de generalización del mejor modelo contra un *dataset* externo. Finalmente, el proyecto presenta la propuesta de un sistema para el estudio de ciberataques web el cual tiene como objetivo facilitar la generación de la información necesaria para el desarrollo de métodos de detección.

Palabras clave

1. Aprendizaje de máquina.
2. Seguridad web.
3. Ataque de comandos en sitios cruzados.
4. Inyección SQL.
5. Ciencia de datos.

ABSTRACT

Security is a complex problem, therefore, the aim is to reduce the gap between the secure and the insecure, for this purpose different techniques have been used to approach a solution. Recently the use of machine learning has been involved for this type of tasks, which is why our project seeks to generate a classification method based on it. Our project proposes a mechanism for the generation and study of the characteristics that allow the training of models that could detect Cross-site Scripting (XSS) and Structured Query Language Injection (SQLI) cyber-attacks, two types of cyber-attacks associated with different computer risks based on the top 10 of the Open Web Application Security Project 2017 (OWASP). To obtain the information, a script was created for the simulation of cyber-attacks in a controlled vulnerable environment, which consists of a server with two virtual machines, one equipped with a vulnerable application and a tool for capturing traffic; the other consists of a web browser and a script for the automation of access to the application. Each attack was made with an information base of known XSS and SQLI attacks obtained from diverse sources. Subsequently, a training of four machine learning algorithms and their evaluation was carried out, as well as the ability to generalize the best model against an external dataset. Finally, the project presents the proposal of a system for the study of web cyber-attacks which aims to facilitate the generation of information necessary for the development of detection methods.

Keywords

1. Machine learning.
2. Web Security.
3. SQL injection.
4. XSS.
5. Data science.

LISTA DE ACRÓNIMOS

1. (HTTP) Protocolo de transferencia de hipertexto (del inglés *Hypertext Transfer Protocol).*
2. (OWASP) Proyecto abierto de seguridad de aplicaciones web (del inglés *Open Web Application Security Project*).
3. (URL) Localizador de recursos uniforme (del inglés *Uniform Resource Locator*).
4. (KNN) K vecinos más cercanos (del inglés *k-nearest neighbors*).
5. (XSS) Scripting de sitios cruzados (del inglés *Cross-site scripting*).
6. (SQL) Lenguaje de consulta estructurado (del inglés *Structured Query Languaje*)
7. (SQLI) Inyección SQL (del inglés *SQL injection*).

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

La seguridad informática se ocupa de diseñar las normas, procedimientos, métodos y técnicas destinadas a conseguir que un sistema de información sea seguro y confiable.

La seguridad web, en especial, involucra proteger la información mediante la prevención, detección y respuestas a las violaciones. Su objetivo principal es mantener las tres características primordiales de la información [1]: confidencialidad, la cual asegura el acceso a la información únicamente a las personas autorizadas; integridad, que es la protección contra la modificación no autorizada o la destrucción de información; y la disponibilidad, aquella que representa el acceso oportuno y confiable a los datos y servicios de información para [2] usuarios autorizados.

Las amenazas web se han venido incrementando a través de los años. Esto debido a diferentes problemas tales como las vulnerabilidades de los navegadores, defectos incluidos en el proceso de elaboración de una aplicación y ciertos problemas con algunos plugin.

A la par con las amenazas, también se han diseñado métodos de detección para prevenirlas, entre las más comunes se encuentran: el análisis estático, el análisis dinámico y el análisis híbrido.

El análisis estático es un método focalizado a encontrar anomalías o elementos maliciosos en el código fuente o en archivos de un elemento digital. Para la identificación de actividades maliciosas, este método puede ser incorporado a través de un enfoque de detección con firmas, es decir, cada elemento malicioso es plasmado en una base de datos (“lista de bloqueados”) que servirá como medio de verificación.

El análisis dinámico adopta el enfoque opuesto y se ejecuta mientras un programa está en funcionamiento, en este se supervisará la memoria del sistema, el comportamiento funcional, el tiempo de respuesta y el rendimiento general. Para este modelo, existe el modelo de detección basado en anomalías que consiste en generar un perfil de las peticiones que se hacen a un servidor, y a partir de este identificar cuáles peticiones difieren de este perfil marcándolas como posibles violaciones.

Finalmente, el análisis hibrido que es una combinación de los métodos anteriores en la que se busca sacar lo mejor de ambos métodos. Primero se realiza un análisis estático para identificar estructuras maliciosas y posteriormente un análisis dinámico para encontrar peticiones anómalas y posibles violaciones.

1. DESCRIPCIÓN DEL PROBLEMA

Actualmente los sistemas de detección de ciberataques web son vulnerables a nuevas metodologías de ataque. La existencia de diferentes tipos de violaciones web, la utilización de sistemas de detección dinámicos, estáticos o híbridos en los cuales las variables de detección no son cubiertas en su totalidad, y la introducción de nuevas tecnologías de desarrollo de aplicaciones web han generado una incertidumbre en la seguridad de la información de las aplicaciones o páginas web. Esto podría llevar a la pérdida de vidas humanas, pérdidas monetarias o sabotajes en las páginas o aplicaciones web.

2. OBJETIVOS DEL PROYECTO

2.1. Objetivo general

Diseñar un sistema para el estudio de ciberataques a aplicaciones web.

2.2. Objetivos específicos

- Generar un *dataset* que plasme al menos dos tipos de ciberataques asociados a tres riesgos informáticos del top 10 OWASP 2017.
- Entrenar un método de detección que sea capaz de clasificar una petición HTTP a un servidor web como segura o insegura.
- Diseñar un sistema para el estudio de ciberataques web que haga uso del método entrenado.
- Validar el método de detección.

3. METODOLOGÍA

3.1. Esquema de trabajo

Desde el 5 de agosto de 2017 que se inició el proyecto, se estableció la realización de reuniones semanales con el tutor, con el objetivo de revisar los avances obtenidos hasta entonces y resolver dudas.

Para el desarrollo del sistema para el estudio de violaciones web se utilizo el modelo de ciclo de vida incremental. Este modelo de ciclo de vida tiene ventajas como: facilidad en el control de riesgos; facilidad en el desarrollo y validación de requerimientos; capacidad para realizar proyectos con poco personal y permite conocer con mayor facilidad el estado de avance.

3.2. Fases de desarrollo del proyecto

A continuación, se describen las actividades que corresponden a cada fase identificada para el desarrollo del proyecto. En anexos podrá encontrar de manera detallada el cronograma del proyecto. Los objetivos específicos fueron mapeados con actividades:

Objetivo específico 1: Generar un *dataset* que plasme al menos dos tipos de ciberataques asociados a tres riesgos informáticos del top 10 OWASP 2017.

Actividades:

- Escoger los dos ataques asociados a tres riesgos informáticos del top 10 OWASP 2017
- Realizar un análisis comparativo de los Honeypots web existentes.
- Realizar un análisis comparativo de los Honeypots server existentes.
- Investigar acerca de *payloads* de los tipos de ataques.
- Realizar un análisis comparativo de aplicaciones web vulnerables existentes.
- Desplegar el entorno para la generación del *dataset.*
- Generar el *dataset* por medio del ambiente y los *payloads* investigados.

Objetivo específico 2: Entrenar un método de detección que sea capaz de clasificar una petición HTTP a un servidor web como segura o insegura.

Actividades:

- Investigar acerca de las características necesarias para la detección de ciberataques web
- Implementar un algoritmo que permita extraer las características investigadas para los ciberataques *SQL Injection* y *XSS*.
- Análisis exploratorio de los datos obtenidos en el *dataset*.
- Selección de los algoritmos de *machine learning* a usar.
- Entrenar y validar de cada uno de los algoritmos previamente seleccionados usando librerías de *machine learning* proporcionadas por RStudio.

Objetivo específico 3: Diseñar un sistema de detección de violaciones web que haga uso del método entrenado.

Actividades:

- Realizar la elicitación de requerimientos.
- Realizar el análisis y especificación de los requerimientos.
- Definir la prioridad de los requerimientos.
- Realizar el diseño global.
- Realizar el diseño detallado.

Objetivo específico 4: Validar el método de detección.

Actividades:

- Investigar acerca de *datasets* existentes que contengan información acerca de ciberataques web relevantes para ser utilizados como prueba de generalización.
- Realizar las pruebas con los modelos entrenados en el objetivo específico 2 utilizando el *dataset* previamente investigado.
- Interpretación de los resultados.
- Conclusión acerca de los resultados.

4. MARCO TEÓRICO Y ESTADO DEL ARTE

4.1. Marco teórico

Para el marco teórico se abordarán las definiciones de los análisis: estático, dinámico e hibrido. También, se cubrirán algunos elementos conceptuales sobre aplicaciones web y una introducción a *Machine learning*.

Los temas abordados en el marco teórico son de utilidad para conocer cómo funcionan y como fue abordado el problema de la ciberseguridad en los siguientes trabajos presentados en el estado del arte. El enfoque del proyecto presentado hará uso *machine learning* porque es una corriente que buscar abordar el problema desde otra perspectiva.

4.1.1. Análisis estático

Este análisis se encarga de detectar ciberataques web con base en el uso de patrones o reglas sin llegar a ejecutar el código [6]. La efectividad de este método se basa en que los patrones que utilizan las violaciones que son previamente conocidos y, por lo tanto, pueden ser detectados fácilmente. Sin embargo, si no se tiene conocimiento del ciberataque o es camuflado de alguna forma no podrá ser detectado por este análisis; por esto, para este tipo de análisis es importante tener una fuente de datos de patrones de ciberataques actualizada.

La principal ventaja de este método es su rapidez en procesamiento [3]. Al ser un método basado en detección de patrones y comparaciones hace que sea rápido al momento de procesar e identificar un posible ciberataque a la página o aplicación web.

La principal debilidad de esta aproximación es la ofuscación de código malicioso [6]. La ofuscación consiste en camuflar el código para ocultar su verdadera forma, esto se consigue cambiando el *encode,* este es un mecanismo utilizado para hacer una traducción de los caracteres no imprimibles o especiales a caracteres aceptados universalmente por servidores web y buscadores*,* dificultando así su identificación. Para solventar esta debilidad existen diferentes aproximaciones para realizar la traducción del código, pero no siempre son efectivas. Adicionalmente este método puede producir falsos positivos al encontrar una estructura que puede parecer maliciosa en peticiones benignas lo cual es molesto al momento de utilizar una aplicación.

4.1.2. Análisis dinámico

El análisis dinámico, a diferencia del estático, se encarga de detectar las violaciones durante la ejecución del código. Este método se basa en el uso de detección por anomalías, es decir establece un comportamiento normal de los usuarios de la aplicación o de la página web para luego hacer una clasificación de las peticiones como normales o anómalas [3].

Para el análisis mediante anomalías, se necesitan definir una serie de factores para garantizar una cobertura de todos los posibles factores de ciberataque [9]. Estos pueden ser, entre otros, longitud de cadenas, distribución de caracteres o gramática de la cadena. Una vez establecidos estos factores, se realiza una fase de adecuación para definir un rango de aceptación. Todas aquellas peticiones que pasen de estos rangos serán clasificadas como anómalas indicando un posible ciberataque.

Su principal ventaja es el alto porcentaje de detección, al tener diferentes análisis puede cubrir un rango más amplio que simplemente la estructura o contenido de la petición, lo que le permite ser un método más efectivo [9].

Su principal desventaja es el tiempo que toma para realizar el análisis de una petición. Al ser dinámico, cada petición es diferente y requiere su propio tiempo de análisis por lo cual incrementa considerablemente el tiempo de respuesta. Y al igual que el análisis estático existe la probabilidad de que este método generé falsos positivos al encontrar un comportamientos anómalos en peticiones benignas.

4.1.3. Análisis hibrido

El análisis hibrido como su nombre lo indica es la combinación de los métodos estáticos y dinámicos para el análisis de violaciones [3]. Con esto se busca obtener lo mejor del análisis estático, que es su rapidez de detección, y del análisis dinámico, que es su efectividad de detección. Para lograr esto se hace una descomposición en dos: primero se aplica un análisis estático para determinar si es o no una petición maliciosa, en caso de ser negativo se aplica un análisis dinámico para verificar y validar el primer diagnóstico.

El siguiente diagrama **Ilustración 1** ilustra cómo se realiza el proceso de detección en un análisis híbrido.

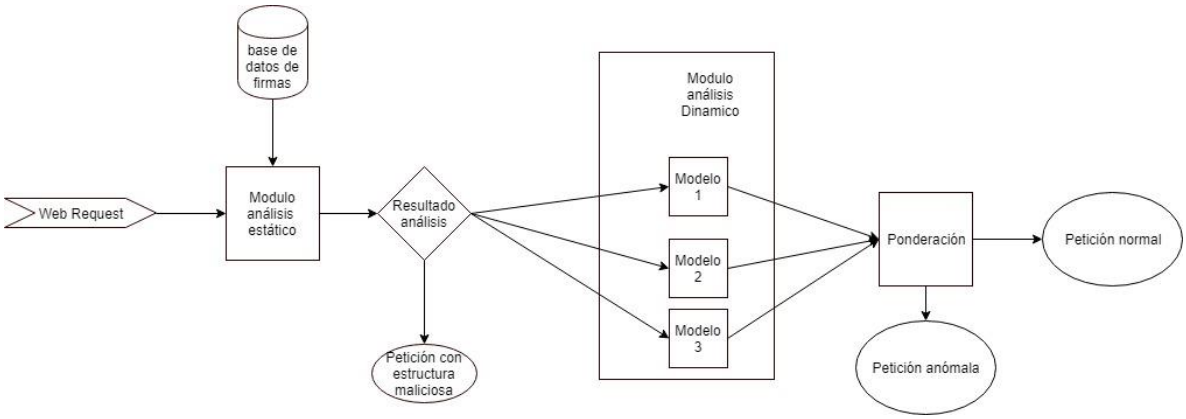


Ilustración 1: Esquema general de un análisis hibrido.

4.1.4. Machine learning

Machine learning es un campo de la Inteligencia artificial que está enfocado en algoritmos que le enseñan a las maquinas a aprender. Para lograr este aprendizaje, se usan algoritmos a los que se proveen una serie de datos para que sean capaces de encontrar o reconocer patrones en la información dada [10].

Sin embargo, el *machine learning* tiene sus limitaciones en cuanto a lo que puede lograr. Si los datos dados no son lo suficientemente variados puede generar un sesgo en el algoritmo, lo que generará predicciones o reconocimientos incorrectos; en caso de que los datos sean incompletos se puede llegar a malinterpretaciones de la información [10].

Los algoritmos cuentan principalmente con dos fases, entrenamiento y clasificación [10]. Durante el entrenamiento se busca que el algoritmo encuentre patrones o los reconozca dependiendo del tipo de aprendizaje que se esté usando y los resultados que se quieran lograr. Existen cuatro enfoques de aprendizaje que se pueden utilizar para el entrenamiento de un algoritmo de *machine learning:* Aprendizaje supervisado, no supervisado, por refuerzo y semi-supervisado; De estos cuatro se analizarán los dos primeros y el último.

El aprendizaje supervisado consiste en aprender basado en etiquetas o datos bien definidos, es decir una serie de entradas etiquetadas con una clase o valor, llamados predictores, con los cuales el algoritmo aprende. Tiene como objetivo predecir una clase o valor con base en los predictores antes mencionados.

En el aprendizaje no supervisado los datos no contienen etiqueta alguna, por lo que no se tiene una clase o valor objetivo, su enfoque es descubrir factores no observados, estructura, o una representación más simple de los datos.

El aprendizaje semi-supervisado es una combinación entre los dos aprendizajes previamente mencionados, incluye una fase de exploración y descubrimiento de factores y una fase de clasificación utilizando estos factores previamente descubiertos.

Una vez entrenado el algoritmo se procede con la clasificación. En el caso de este proyecto, el algoritmo será entrenado mediante un aprendizaje supervisado para el reconocimiento de patrones de violaciones web y así poder clasificarlos como seguros o inseguros.

4.2. Estado del arte

Las violaciones web pueden ser evaluadas a partir de distintos enfoques: análisis estático, análisis dinámico y análisis híbrido [16]. Dado el objetivo de este proyecto, definimos el uso de *machine learning* como método de clasificación principal debido a que este presenta un enfoque diferente para el manejo e interpretación de los datos y, a su vez mejores resultados, en general, comparado con los métodos tradicionales [17]. Cada una de las soluciones que se presentan en esta sección proponen distintos enfoques para la detección tanto de violaciones (que es el enfoque de este proyecto) como para la detección de páginas web maliciosas.

4.2.1. Detection of Server-side Web Attacks (DSSA)

En [9] se propone un sistema de detección de intrusos basado en anomalías con reconocimiento de patrones para la detección de ciberataques hacia servicios web. Se utilizan modelos probabilísticos, en los cuales se le asigna una probabilidad (de normalidad) a cada petición realizada hacia el servidor. Se utiliza un *dataset*, generado por ellos mismos, en el cual plasmaron cerca de 450.000 peticiones hacia un servidor web de su institución académica, en un intervalo de 1 semana. Como medida adicional, crearon una serie de entradas (507) para probar el sistema, en el cual registraron violaciones diseñadas por ellos mismos. El *dataset* fue pre procesado utilizando un algoritmo de detección de *outliers* para reducir su ruido.

Se utilizaron 9 características principales, las cuales fueron divididas en 3 modelos estadísticos: el primero, denotado por la letra A, basado en la secuencia de símbolos; el segundo, denotado por la letra B, la distribución estadística de un valor entero no negativo; el tercero denotado por la letra C, la distribución estadística de símbolos.

Se obtuvieron los siguientes resultados: tasa de detección, 232/232 violaciones (100%) para el *dataset* generado y 505/507 violaciones (99,6%) para el *dataset* adicional que contenía las violaciones; y tasa de falsa alarma, 1.252 alertas/447.178 peticiones (0,28%). Se concluyó que el sistema diseñado es aceptable, con tasas altas de detección y muy bajas de falsa alarma. Se pueden mejorar los resultados si se refina el *dataset*, es decir si se mejora el algoritmo de detección de *outliers*, el cual puede mejorar las tasas obtenidas.

Este sistema da una soluciona aproximada al problema de detección de ciberataques, aunque difiere en la forma en que se aborda, este sistema lo hace desde una perspectiva probabilística, diferente a la que se propone en este proyecto.

4.2.2. Anomaly Detection of Web-based Attacks (ADWA)

Por otra parte, en [3] el autor presenta un sistema de detección de violaciones web basado en anomalías, en el cual se toma ventaja de la estructura de una petición HTTP que contiene parámetros. Para la evaluación de este sistema, se utilizaron 3 *dataset* diferentes: registros de un servidor web Apache en producción de Google; los dos restantes fueron obtenidos de servidores del Departamento de Ciencias de la Computación de la Universidad de California, Santa Barbara (UCSB) y la Universidad Técnica de Vienna (TU Vienna). Se utilizaron 11 *exploits reales* para la experimentación obtenidos de sitios de seguridad populares.

Este sistema utiliza 6 modelos para la detección: el primero, llamado modelo de tamaño de atributos; el segundo, la distribución de caracteres; el tercero, modelo de inferencia estructural; el cuarto, el *token finder*; el quinto, modelo de presencia o ausencia de atributos; y el último, orden de atributos.

Como resultados se obtuvieron 11/11 violaciones (100%) detectadas por el sistema. En cuanto a los falsos positivos obtenidos: el *dataset* de Google presentó 206 alertas/490.704 peticiones (0,04%), UCSB 3 alertas/4.617 peticiones (0,06%) y TU Vienna 151 alertas/713.500 peticiones (0,02%). Se concluyó que las violaciones web deben abordarse mediante herramientas y técnicas que mezclen la precisión de la detección basada en firmas con la flexibilidad de un sistema de detección basado en anomalías. Las tasas de falsos positivos podrían reducirse con el refinamiento de los algoritmos utilizados.

Este sistema, al igual que al anterior, soluciona el problema de detección de violaciones, pero lo aborda desde una perspectiva basada en anomalías, diferente a la que se propone en este proyecto.

4.2.3. A hybrid approach for detecting malicious web pages using decision tree and Naïve Bayes algorithms (HAMW)

El autor en [11] propone un modelo de detección híbrido de URL maliciosas utilizando dos algoritmos de clasificación: Decision Tree y Naïve Bayes. Con esto, se clasifican las páginas web como malignas o benignas. Para la evaluación de este sistema se utilizó un *dataset* que contenía 3.000 páginas web benignas y 355 páginas maliciosas obtenidas del sitio web de ranking Alexa [12] y PhishTank [13] respectivamente.

Este sistema utiliza un método de tres fases: la primera fase, el mecanismo innato para la validación basada en firmas de la página; la segunda fase, la extracción de características de manera dinámica; y la tercera fase, clasificación mediante algoritmo de árbol de decisión y Naïve Bayes.

Como resultados se obtuvo una tasa de detección del 93,1% y una tasa de falsos positivos de 6,9%. Adicionalmente se evaluó el desempeño por separado de cada uno de los algoritmos de clasificación utilizados, lo que da como resultado lo siguiente: el árbol de decisión utilizado presentó una tasa de detección del 83,1% y una tasa de falsos positivos de 16,9%; y Naïve Bayes obtuvo una tasa de detección del 66,1% y una tasa de falsos positivos de 33,9%. Se concluyó que el sistema híbrido propuesto, compuesto por dos algoritmos de clasificación (árbol de decisión y Naïve Bayes), presentó una tasa alta de detección comparado con la utilización por separado de cada uno de estos algoritmos. Como trabajo a futuro, el autor propone la aplicación de este método a un sistema de detección de intrusiones.

Este sistema, aunque cuenta con un método de *machine learning* (que es el método propuesto en este proyecto) no se enfoca el problema planteado ya que no se centra en la detección de violaciones, sino en la detección de páginas maliciosas.

4.2.4. Detection of malicious web pages based on hybrid analysis (DMWH)

El sistema propuesto en [14] aborda la utilización de un sistema híbrido, combinando los aspectos positivos tanto de un sistema estático como de uno dinámico. Para aquellas páginas desconocidas se utiliza un análisis dinámico, el cual analiza la ejecución de esta y decide si es o no maliciosa. Para validar este sistema, se utilizaron 1.469 páginas web de las cuales 787 fueron benignas (obtenidas rastreador web que permitía solicitar datos de las páginas web del sitio de ranking Alexa [12]) y 682 malignas (recolectadas de la lista de URL maliciosas de Malware Domain List [15]).

El sistema propuesto utiliza dos conjuntos de características, una estática y una dinámica. El proceso se divide en 3 métodos: el primero, la extracción de

características estáticas y dinámicas de la página (las características dinámicas fueron obtenidas a través de un método dinámico de análisis, el cual ejecuta directamente el código JavaScript de la página en un ambiente controlado para su posterior análisis); el segundo, la selección de las características más representativas mediante su correlación; y el tercero, la utilización de un algoritmo de clasificación (árbol de decisión).

Como resultados, se obtuvo que el sistema propuesto presenta una precisión del 95,2%, con una tasa de predicción del 88% y una tasa de falsos positivos de 0,048%. Se concluyó que, debido que los atacantes web actualizan sus técnicas para burlar los sistemas de detección existentes, no basta con solo un análisis estático, este debe estar acompañado por un análisis dinámico que comprueba en tiempo de ejecución el comportamiento de la página. Los resultados pueden mejorar en un futuro si se refinan los métodos de clasificación y se aumenta el tamaño del *dataset* utilizado.

Este sistema no soluciona el problema planteado, ya que no cuentan con un método de *machine learning* para la detección de amenazas; además se enfoca en el análisis de páginas web y no de peticiones a estas.

4.2.5. Machine Learning Classifiers to Detect Malicious Websites (MLCMW)

En [18] se propone un sistema de detección que utiliza una serie de características que sirven para realizar la clasificación mediante 4 algoritmos de *machine learning*. Se utilizó un *dataset* generado por ellos mismos, en el cual se presentaron 967 registros (861 observaciones benignas y 106 malignas) obtenidos de distintos sitios web.

El sistema propone una división por capas de características a evaluar: en la de aplicación, se evalúan 10 variables; y en la de red, se evalúan 12 variables. Estas características son evaluadas mediante 4 algoritmos de clasificación: Support Vector Machine (SVM), Regresión logística (RL), Naïve Bayes (NB) y C 4.5 (en su versión de Weka llamado J48).

Como resultados, se obtuvo que el sistema basado en machine learning obtuvo una exactitud de: 94,71% para SVM, 90,58% para Regresión logística, 10,96% para Naïve Bayes y 98,76 para J48. Se concluyó que el método de obtención de los datos no está documentado, por lo que se propone una búsqueda exhaustiva para mejorar los resultados presentados; al igual, se puede afirmar que el algoritmo J48 presenta los mejores resultados al momento de identificar una página web maliciosa.

Este sistema, aunque cuenta con un método de *machine learning* (que es el método propuesto en este proyecto) no se enfoca en el problema planteado ya que no se centra en la detección de violaciones, sino en la detección de páginas maliciosas.

4.2.6. Defending malicious script attacks using machine learning classifiers (DMSML)

El autor en [19] propone un sistema que presenta un interceptor entre el navegador y el servidor que analiza el código JavaScript para clasificar la petición como benigna o maligna. Se utilizó un *dataset* que contenía 1.924 registros, de los cuales 1.515 eran benignos y 409 maliciosos.

El sistema propone la extracción de características del código JavaScript, extrayendo únicamente aquellas que son relevantes, para ello utiliza un método wrapper. Estas características son evaluadas mediante 4 algoritmos de clasificación: Naïve Bayes, C 4.5 (en su versión de Weka llamado J48), Support Vector Machine (SVM) y K Nearest Neighbors (K-NN). Para la evaluación de este sistema se utilizaron 3 pruebas distintas: 100% training, 80% training 20% test y 10-fold cross-validation.

Como resultados, se obtuvo una exactitud de: Naïve Bayes 97,25%, J48 97,09%, SVN 96,51% y K-NN 100% para 100% training; Naïve Bayes 97,99%, J48 98,64%, SVN 95,42% y K-NN 96,41% para 10-fold cross-validation; Naïve Bayes 95, 06%, J48 99,22%, SVN 94,55% y K-NN 97,14% para para 80% training y 20% test. Se concluyó que K-NN fue el algoritmo con mejores resultados; también se afirmó que el método de extracción de características wrapper jugó un papel importante en los buenos resultados presentados.

Este sistema hace uso de *machine learning* (enfoque propuesto por nosotros) pero soluciona el problema propuesto parcialmente, ya que se centra únicamente en la detección de un tipo de ciberataque (Cross Site Scripting – XSS).

Característica\Sistema	DSSA	ADWA	HAMW	DMWH	MLCMW	DMSML	Nuestro sistema
Método utilizando *machine learning*	✗	✗	✓	✓	✓	✓	✓
Detección ataques web	✓	✓	✗	✗	✗	✓	✓
Análisis híbrido	✗	✗	✓	✓	✗	✗	✓
Validación mediante un simulador de ataques	✗	✗	✗	✗	✗	✗	✓

Tabla 2: Cuadro comparativo estado del arte.

5. RESULTADOS Y RECOMENDACIONES

5.1. Investigación

5.1.1. Análisis de riesgos informáticos:

OWASP es una comunidad abierta dedicada a desarrollar, mantener y comprar aplicaciones y APIs que sean confiables. Adicionalmente ofrecen una serie de recursos para lograr incrementar la seguridad de las aplicaciones web. Entre estos recursos OWASP publica periódicamente un top 10 con los riesgos informáticos más críticos, información sobre que representan y sus posibles repercusiones y finalmente una serie de recomendaciones para evitar el riesgo.

Teniendo en cuenta este top decidimos enfocarnos en los siguientes riesgos informáticos:

- *Injection*: La inyección de datos consiste en el envío de datos no confiables y estos son interpretados como parte de la búsqueda. Los atacantes pueden acceder a información confidencial o ejecutar comandos. Aunque existen diferentes tipos de inyección nosotros nos enfocamos en la inyección SQL.
- *Broken Authentication*: Consiste en el manejo inseguro de control de sesiones o autenticaciones. Mediante este riesgo los atacantes pueden obtener acceso a las aplicaciones y realizar acciones en nombre de un usuario determinado.
- *Sensitive Data Exposure*: Muchas aplicaciones web no protegen bien la información sensitiva, esto permite a los atacantes a acceder información que puede ser confidencial y ser usada con fines maliciosos.
- *Cross-Site Scripting (XSS)*: El XSS consiste en la inyección de código HTML o JavaScript en aplicaciones de tal modo que le permitan al atacante ejecutar este código en el navegador de la víctima. Esto puede ser usado con diferentes intenciones entre ellas conseguir sesiones de usuarios, realizar *defacement* o redirigir al usuario a sitios maliciosos.

Una vez definidos estos riesgos decidimos enfocarnos en los ataques tipo XSS y SQLI. Estos ataques están directamente relacionados con dos riesgos informáticos e indirectamente relacionados con los otros dos debido a esto y a que representan las fallas de seguridad más graves existentes enfocamos nuestra investigación en la extracción de sus características.

5.1.2. Análisis para la selección de Honeypots

Un Honeypot es un ambiente computacional monitoreado que funciona como anzuelo. Existen diferentes tipos de Honeypots y cada uno de estos tienen diferentes funciones, en nuestro caso realizaremos un análisis acerca de los Honeypots server y web. Los Honeypots server tiene como función emular un servidor vulnerable para atraer a los atacantes y obtener información acerca de los ataques realizados, también pueden tener herramientas de monitoreo de red y sistemas de alertas para avisar de un intruso o un ataque. Los Honeypots web a diferencia de los servers se enfocan únicamente en aplicaciones web, estos ofrecen herramientas para monitorear las peticiones y determinar si son maliciosas o no.

Para nuestra investigación el uso de un Honeypot tiene como fin el poder monitorear y extraer información acerca de peticiones malignas realizadas al servidor y la aplicación web. Por lo tanto, las características deseables eran la capacidad de adaptabilidad, automatización de la captura de la información, clasificación de las peticiones y generación de una captura de tráfico.

Para realizar un análisis y comparación tuvimos en cuenta características como, última fecha de actualización, vulnerabilidades asociadas, tipo de interacción y una descripción sobre su funcionamiento.

5.1.3. Análisis comparativo Honeypots server

Para una primera iteración realizamos un análisis y clasificación de diferentes tipos de Honeypots server. Para esto hicimos un cuadro comparativo de 17 Honeypots que se puede ver en el anexo (Anexo 1). De estos Honeypots se seleccionaron tres para realizar una instalación y prueba de sus funcionalidades y determinar si cumplían con el objetivo de generación del *dataset*. Adicionalmente se encontró que la mayoría de estos Honeypots tienen un alcance muy específico que no se encuentra alineado con nuestro proyecto y se encuentran sin soporte actualmente.

Para la selección de los honeypots a analizar se tuvieron en cuenta los siguientes criterios: capacidad de detección, generación de capturas de tráfico, clasificación de peticiones y el tipo de enfoque de la herramienta.

Los Honeypots que fueron seleccionados para análisis fueron:

5.1.3.1. Beeswarm

Beeswarm es un Honeypot tipo server que tiene como objetivo determinar el comportamiento normal de una red y así poder clasificar el comportamiento anómalo. Actualmente no se encuentra documentación acerca del proyecto y se encuentra sin soporte. Se intento realizar una instalación y prueba del Honeypot pero no fue posible por la falta de guías y una serié de errores en el código fuente que no fueron posibles de solucionar.

5.1.3.2. Conpot

Conpot es un proyecto que aún se encuentra en desarrollo y tiene como objetivo simular una red empresarial grande incluso contando con diferentes tipos de dispositivos diferentes a servidores. Al ser un proyecto aún en desarrollo al realizar la instalación y configuración del ambiente se encontró que algunas de las herramientas tenían errores y no era posible realizar una configuración que supliera nuestras necesidades.

5.1.3.3. KFSensor

KFSensor es una herramienta comercial por lo cual tiene soporte y se encuentra en constante desarrollo, tiene herramientas de monitoreo, clasificación y generación de las capturas de tráfico además de facilitar la generación de escenarios y servidores de FTP, SMTP entre otros. Su instalación es sencilla pero limitada a Windows. A pesar de todas las ventajas que tiene este Honeypot no fue posible realizar una configuración de los servicios HTTP y HTTPS puesto que solo tiene como opción arrojar un banner acerca del tipo de servidor que se está corriendo.

5.1.4. Análisis comparativo Honeypots web

Al finalizar la revisión de los Honeypots server seleccionados anteriormente notamos que no satisfacen las necesidades planteadas para el proyecto porque no permitían una emulación completa del servicio web o realizar la captura de la información que son características necesarias para lograr nuestro objetivo de generación del *dataset*. Al no poder emular completamente el servicio web y sus vulnerabilidades no nos permite obtener un *dataset* con todas las características necesarias y sin poder tener un registro de la captura de información es imposible generar un el conjunto de datos necesario para el estudio. Al encontrar estas limitaciones iniciamos una búsqueda de un Honeypot web que cumpla con las características de emulación de vulnerabilidades de aplicaciones web y tenga la capacidad de caracterizarlas.

Para lograr esto realizamos un análisis comparativo de 19 Honeypots tipo web que se pueden encontrar en el anexo (Anexo 2). De estos 19 Honeypots encontramos limitaciones parecidas a las del tipo server, la mayoría están enfocados en un alcance muy específico, se encuentran sin soporte o se encuentran obsoletos. Al final de análisis seleccionamos tres para su instalación y prueba de funcionalidad.

Adicional a este análisis fue necesario realizar una comparación entre diferentes aplicaciones web vulnerables para determinar la mejor para nuestra investigación, además que se acoplara a las características de los Honeypots deseados. Para ello realizamos una comparación entre 5 aplicaciones vulnerables utilizadas para el estudio de test de penetración. Para ello se analizaron características como vulnerabilidades ofrecidas, fecha de actualización y lenguaje. La tabla se puede encontrar en el anexo (Anexo 3). La aplicación escogida para realizar el trabajo fue DVWA, una aplicación web bien conocida y utilizada para el ejercicio de test de penetración.

Los tres Honeypots tipo web seleccionados fueron:

5.1.4.1. Gastpof

Gastpof es un proyecto de Honeypot web para el estudio de vulnerabilidades en las aplicaciones, actualmente se encuentra obsoleto y tiene una segunda versión en dos proyectos llamados Snare y Tanner. No pudo ser probado debido a errores que tiene en la versión accesible en GitHub. Pero por su descripción tenía las herramientas necesarias para el estudio realizado en este proyecto.

5.1.4.2. Snare/Tanner

Aunque son dos proyectos separados están orientados para funcionar juntos, Snare se encarga de realizar una copia de una aplicación web insertando vulnerabilidades conocidas y desplegándola como un anzuelo. Por su parte Tanner se encarga de realizar un análisis de las peticiones HTTP que se realizan a Snare. Ambos proyectos aún se encuentran en desarrollo por lo tanto se encuentran errores en su instalación y clonación de las páginas.

5.1.4.3. Shadow-Daemon

Shadow-Daemon es un Honeypot tipo firewall de alta interacción que se enfoca en controlar las peticiones que se realizan a una aplicación. Tiene como limitaciones los lenguajes en los que pueden estar escritos las aplicaciones, PHP, Perl y Python. Su funcionamiento se basa en la generación de una *whitelist* para realizar un filtro

a partir de anomalías o una *blacklist* para identificar un conjunto de posibles formas de ataque. Adicionalmente permite clasificar las peticiones asociadas a un ataque realizado. Las desventajas encontradas fueron, la dificultad para generar un conjunto de reglas para la *blacklist* que funcionaran para la aplicación, la generación de *whitelist* era imposible puesto que requería un tiempo mínimo de uso normal por diferentes usuarios y no fue posible exportar la información generada por el Honeypot.

5.1.5. Análisis final del ambiente a utilizar

Al no poder utilizar estas herramientas para la obtención de la información debido a sus diferentes restricciones, se optó por utilizar un ambiente vulnerable que se puede observar en la ilustración 2:

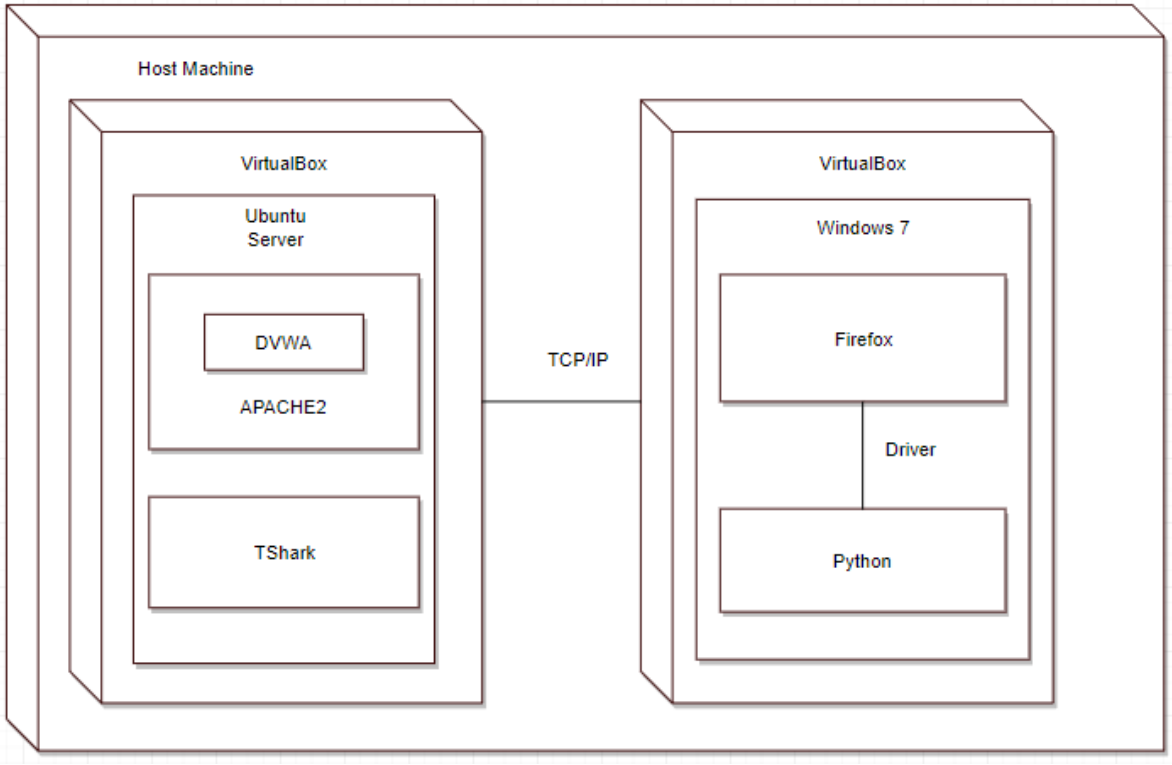


Ilustración 2: Arquitectura del sistema vulnerable planteado en este proyecto.

Se utiliza un servidor Ubuntu para exponer la aplicación vulnerable y como herramienta de captura se utiliza TShark que genera una captura de tráfico por tiempo y en formato PCAP. Para la parte del cliente se utiliza Firefox como navegador predeterminado y a través de un script de Python se realiza el control de este para acceso y realización de los ataques sistemáticos. Como fuente de información para los ataques se utilizaron listas de ataques tipo XSS y SQLI.

5.1.6. Análisis de las características de detección

La identificación de características es un paso muy importante en la detección de ciberataques, pues de la correcta identificación de estas depende el éxito que tendrá el clasificador utilizado. Estas características describen el comportamiento de un ciberataque, y plasman la interacción que el ciberataque tiene en sus distintas fases tanto en la capa de aplicación como en la capa de red.

Se analizaron distintas propuestas en las cuales se evalúan diferentes características para la identificación de ciberataques.

Se tomó como base las características identificadas en [18], después de un análisis profundo se descartaron por completo estas características debido a que estaban enfocadas en la comunicación con una página web, por lo que no se ajustan al enfoque del proyecto que son únicamente las peticiones HTTP hacia un servidor.

El autor en [20] propone una serie de características para el protocolo HTTP que pueden ser útiles al momento de detectar ciberataques tales como XSS, SQL Injection, DoS, entre otros y las divide en categorías (para ver en detalle cada categoría ir a Anexo 4):

- Información general de la petición.
- Contenido de la petición.
- Respuesta del servidor.
- Historial de solicitudes

En [21] el autor lista una serie de características identificadas en una petición HTTP y las subdivide en categorías, de acuerdo con el aspecto de la petición (para ver en detalle cada categoría ir a Anexo 5):

- URL.
- Método.
- Host.
- Parámetros.
- Cookies.
- Referer URL.
- Información delicada.
- Archivo sospechoso.
- Patrón sospechoso.

En [19] se propone 70 características de detección para ciberataques XSS obtenidas mediante un método de extracción de características (para verlas ir a

Anexo 6). El autor no especifica la descripción de cada característica, pero el nombre es bastante descriptivo.

En [22] el autor propone el análisis de algunas funciones JavaScript utilizadas frecuentemente en ciberataques, como lo son: eval(), attachEvent(), parseInt(), entre otras.

Para [23] el autor habla sobre la manipulación del DOM mediante código JavaScript, al igual analiza el comportamiento de ciertas funciones de JavaScript juntas como: eval(), unescape() y find().

El autor en [24] resalta una característica importante de los ciberataques mediante peticiones HTTP, la ofuscación del código. Este mecanismo consiste en cambiar la codificación del código enviado para ocultar su estructura maliciosa (ver ilustración 3). Basado en esta premisa, para la extracción de características se utilizaron funciones para convertir el código ofuscado. Debido a la ofuscación del código, la longitud de la URL puede crecer, debido a esto el autor presenta una característica importante, la longitud de la URL.

Además de esto, el autor analiza dos características nuevas, el redireccionamiento hacia otras páginas mediante el código y la presencia de tags HTML en este.

Ilustración 3: Ejemplo de código ofuscado. Tomado de [24]

Como se puede observar, los autores proponen clasificaciones diferentes para las características, pero estas no difieren mucho de una investigación a otra; en todas se tiene en cuenta el análisis de características muy comunes en todos los ciberataques como lo son el análisis de las cabeceras de la petición, al igual que el llamado a métodos o funciones JavaScript (en el caso preciso de los ciberataques XSS) y el ingreso de sentencias SQL (para el caso de SQL Injection).

Las características antes mencionadas fueron de apoyo para seleccionar las características finales de este proyecto. Se realizó una depuración de estas,

quedándose únicamente con aquellas que en mayor medida puedan identificar correctamente los ciberataques a trabajar.

5.1.6.1. Características finales

A continuación, se presentan las 13 características finales seleccionadas para este proyecto:

Nombre	**Tipo**	**Descripción**
req_uri_length	Entero positivo	Longitud de la URI en la petición.
req_method	Simbólico	Tipo de petición (POST/GET)
non_printchars	Entero positivo	Número de caracteres no imprimibles en la petición.
contains_sql_commands	Booleano	¿La petición contiene comandos SQL?
sensitive_files	Booleano	¿La petición contiene llamados a archivos especiales (../etc/passwd)?
directory_traversal	Booleano	¿La petición contiene recorridos entre directorios (../ , ../../)?
default_login_credentials	Booleano	¿La petición contiene credenciales de acceso predeterminadas (admin,guest,user)?
contains_html_tags	Booleano	¿La petición contiene tags HTML (<iframe>, <script>)?
dom_manipulation	Booleano	¿La petición contiene llamados para la manipulación del DOM?

url_chain	Booleano	¿La petición contiene redirecciones hacia otras páginas?
eval_func_presence	Booleano	¿La petición contiene llamados a la función de JavaScript eval()?
unescape_func_presence	Booleano	¿La petición contiene llamados a la función de JavaScript unescape()?
write_func_presence	Booleano	¿La petición contiene llamados a la función de JavaScript write()?
getelementsbytagname_func_presence	Booleano	¿La petición contiene llamados a la función de JavaScript getElementByTagName()?
alert_func_presence	Booleano	¿La petición contiene llamados a la función de JavaScript alert()?
fromcharcode_func_presence	Booleano	¿La petición contiene llamados a la función de JavaScript fromCharCode()?

Tabla 3: Características de detección finales.

5.1.7. Ambiente vulnerable y simulación ciberataques

Para la realización de este proyecto se simuló un ambiente vulnerable, en el cual se simularon ciberataques XSS y SQL Injection. La forma en que se realizó fue la siguiente:

1- Se recaudaron diferentes *payloads* asociados a ciberataques XSS y SQL Injection encontrados en la web (cabe resaltar que no fue fácil esta búsqueda, es información que comúnmente no está pública) como base de la simulación.

2- Se simuló un ambiente vulnerable, utilizando dos máquinas virtuales: la primera con Ubuntu 16.04 LTS tomando el rol de servidor atacado y DVWA [33] como aplicación vulnerable; la segunda con Windows 7 Ultimate, tomando el rol de máquina atacante. Ambas máquinas tuvieron como host dos equipos: la primera con Intel Core i7 3770K, 16GB RAM; la segunda con Intel Core i7 7700HQ, 16GB RAM.

3- Se creó un *bot* en Python y utilizando Selenium[34] en el cual simulaba el acceso al ambiente vulnerable e inyectaba los *payload* descritos anteriormente.

4- En la máquina atacada se utilizó TShark[35] para la captura del tráfico generado por la máquina atacante.

5- Cada captura tenía asociado un *payload* distinto. El tiempo total de cada captura fue de 3 minutos, basado en []

Se realizaron un total de 547 capturas de tráfico, de ellas 115 asociadas a ciberataques SQL Injection y 432 asociadas a ciberataques XSS.

```
req_uri_length   req_method non_printchars    contains_sql_commands
Min.   :  1.00   GET :585   Min.   :0.00000   Mode :logical
1st Qu.: 15.00   POST:544   1st Qu.:0.00000   FALSE:1048
Median : 28.00              Median :0.00000   TRUE :81
Mean   : 28.74              Mean   :0.06555
3rd Qu.: 28.00              3rd Qu.:0.00000
Max.   :235.00              Max.   :7.00000
sensitive_files directory_traversal default_login_credentils contains_html_tags
Mode :logical   Mode :logical       Mode :logical            Mode :logical
FALSE:1129      FALSE:1129          FALSE:657                FALSE:947
                                    TRUE :472                TRUE :182


dom_manipulation url_chain       eval_func_presence unescape_func_presence
Mode :logical    Mode :logical   Mode :logical      Mode :logical
FALSE:1113       FALSE:1127      FALSE:1129         FALSE:1129
TRUE :16         TRUE :2


write_func_presence getlementsbytagname_func_presence alert_func_presence
Mode :logical       Mode :logical                      Mode :logical
FALSE:1127          FALSE:1129                         FALSE:1063
TRUE :2                                                TRUE :66
```

```
fromcharcode_func_presence        attack_type       type
Mode :logical               NONE          :609   BENIGNA:609
FALSE:1129                  SQL INJECTION: 88   MALIGNA:520
                            XSS           :432
```

Ilustración 4: Estructura del *dataset* utilizado.

5.1.8. Análisis exploratorio

Para el desarrollo del proyecto se generó un *dataset* utilizando un algoritmo de extracción de características (con las características antes mencionadas).

Se puede ver más en detalle la estructura del *dataset* en la Ilustración 4.

El *dataset* cuenta con un total de 1129 peticiones HTTP, de las cuales 609 son benignas y 520 son malignas.

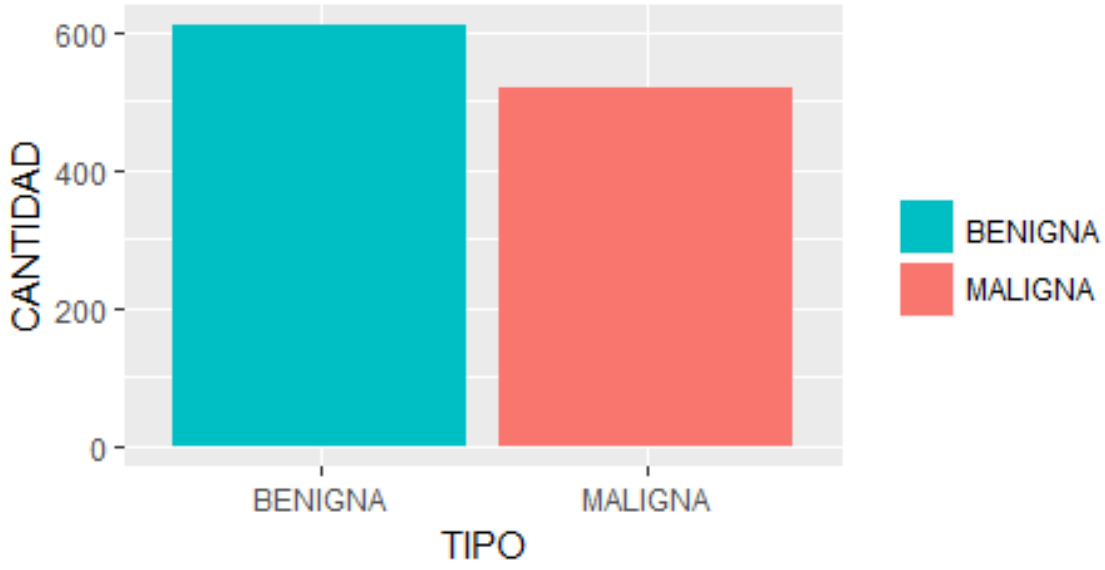


Ilustración 5: Distribución del tipo de peticiones en el *dataset*.

De estas 520 peticiones malignas se etiquetaron 88 ciberataques SQL Injection y 432 ciberataques XSS.

Ilustración 6: Distribución de los tipos de ataque en el *dataset*.

5.1.9. Entrenamiento, validación y selección del método de detección

Para este proyecto se plantea la evaluación de 4 algoritmos de clasificación: C 4.5 (árbol de decisión) [25], Random Forest (árbol de decisión) [26], Naïve Bayes [27] y KNN (K-Nearest Neighbors) [27].

5.1.6.1 Entrenamiento

Para el entrenamiento se utilizó el método k-fold cross validation [28] en el cual se asegura que el modelo se entrene adecuadamente con el *dataset.* Se utilizó un K=10 para este algoritmo, sugerido en [28].

En el caso específico del entrenamiento del algoritmo KNN, para la selección del K óptimo normalmente se opta por una selección empírica basada en la experimentación, en este proyecto el K fue determinado automáticamente por la librería Caret [29].

5.1.6.2. Validación

Para la validación se utilizó un *dataset* (fusión de 2 *datases* tomados de [30]*)* que consta de 60.668 peticiones HTTP, de las cuales 36.000 corresponden a peticiones normales (benignas) y 24668 pertenecen a peticiones anómalas (incluye ciberataques como SQL Injection, XSS, desbordamiento de *buffer*, inyección CRLF, manipulación de parámetros, entre otros).

Las peticiones anómalas no se encuentran etiquetadas con ningún tipo de ciberataque, por lo que no se puede saber con exactitud cuántas peticiones corresponden a ciberataque.

Para cada clasificador se evaluaron tres indicadores principales: Kappa [31], Precisión [32] y F-Measure [32] mostrados al final de cada experimento en un cuadro comparativo.

Además de esto se utilizó la matriz de confusión de cada clasificador, una herramienta que permite la visualización del desempeño de un algoritmo de clasificación (Ilustración 5) la cual presenta otros indicadores importantes tales como FP (falsos positivos, cantidad de instancias benignas clasificadas como malignas, en la seguridad este tipo de errores son molestos pero no influyen un riesgo delicado), FN (falsos negativos, cantidad de instancias malignas clasificadas como benignas, en la seguridad este tipo de errores son muy graves, influyen un gran riesgo), *Recall* (proporción de verdaderos positivos clasificados correctamente) y Especificidad (proporción de verdaderos negativos clasificados correctamente). Estos últimos indicadores son mostrados en cada algoritmo de clasificación.

Valor real / Valor predicho	Uva	Fresa
Uva	10	3
Fresa	5	13

Ilustración 7 : Ejemplo matriz de confusión.

Para entender un poco mejor los indicadores previamente mencionados, se realizó la Ilustración 6. Las convenciones de esta son las siguientes:

- A = VP (Verdaderos positivos)
- B = FP (Falsos positivos)
- C = FN (Falsos negativos)
- D = VN (Verdaderos negativos)
- A+B = PP (Predicciones positivas)
- C+D = PN (Predicciones negativas)
- A+D = RP (Valores reales positivos)
- B+D = RN (Valores reales negativos

Valor real / Valor predicho	Uva	Fresa	Total
Uva	A	B	A+B
Fresa	C	D	C+D
Total	A+D	B+D	N

Ilustración 8: Explicación indicadores en clasificación binaria.

Se realizaron 2 experimentos para la validación: el primero, tomando todo el *dataset* antes mencionado como medio de prueba; el segundo, creado en base a los resultados del experimento 1 y con la voluntad de seguir la filosofía de la ciencia de datos en al cual los resultados de un experimento se pueden refinar y servir como entradas para el siguiente experimento, se adaptó una parte del *dataset* antes mencionado (solo el considerado anómalo por [30]) utilizando el algoritmo de extracción de características (enfocado como bien se ha comentado anteriormente en 2 ciberataques, XSS y SQL Injection) para generar un nuevo *dataset* en el cual se etiquetan como peticiones malignas aquellas que están asociadas a estos ciberataques, el resto de las peticiones se consideran como benignas (aunque no lo son) ya que el alcance del proyecto se definió en estos 2 ciberataques y no en se evaluaron los demás (desbordamiento de *buffer*, inyección CRLF, manipulación de parámetros, etc.).

Dado el enfoque de analítica y ciencia de datos del proyecto, se utilizó una metodología propuesta por IBM llamada ASUM-DM [35] en la cual, dados los resultados del experimento 1, se realiza una iteración al momento de resultados, es decir, se hace una retroalimentación de los resultados a los clasificadores y se hace un refinamiento el *dataset*, dando como resultado el experimento 2.

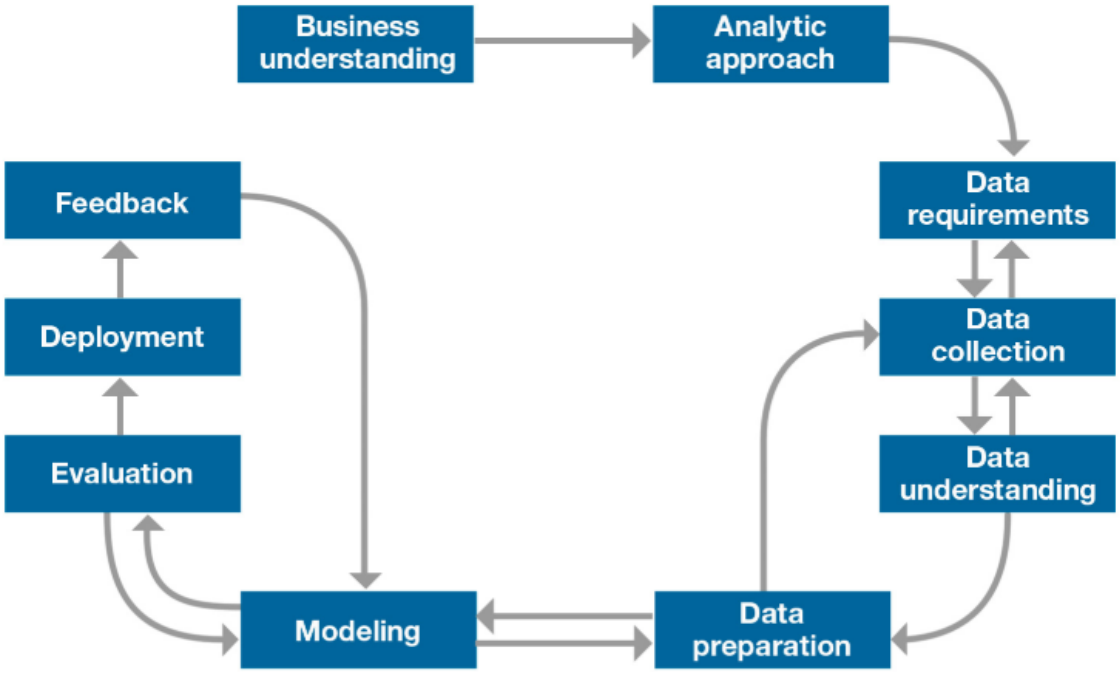


Ilustración 9 : Metodología ASUM-DM.

5.1.6.2.1. Experimento 1

A continuación, se presentan los indicadores antes mencionados para cada algoritmo de clasificación:

5.1.6.2.1.1. Random Forest

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	35994	24122
MALIGNA	6	546

Indicador	FP	FN	Recall	Especificidad
Valor	24122	6	0.999	0.022

Este clasificador presenta una gran cantidad de FP, que dicho previamente es un error muy grave en la seguridad ya que clasifica como BENIGNA una petición MALIGNA, es decir no detecta un ciberataque. Aunque se cuenta con buen Recall (cantidad de peticiones BENIGNAS clasificadas como BENIGNAS) la Especificidad es bastante mala (cantidad de peticiones MALIGNAS clasificadas como MALIGNAS). En conclusión no es un clasificador adecuado para este proyecto, ya que se tienen muchos FP.

5.1.6.2.1.2. KNN

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	26995	17795
MALIGNA	9005	6874

Indicador	FP	FN	Recall	Especificidad
Valor	17795	9005	0.749	0.278

Similar al anterior clasificador, se tiene una gran cantidad de FP y se le suma una cantidad considerable de FN. El Recall y la Especificidad mejoraron, pero la cantidad de errores sigue siendo bastante considerable, en especial los FP. En conclusión no es un clasificador adecuado para este proyecto

5.1.6.2.1.3. C 4.5

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	35994	24122
MALIGNA	6	546

Indicador	FP	FN	Recall	Especificidad
Valor	24122	6	0.999	0.0221

Se tienen los mismos resultados que el clasificador Random Forest, esto se puede explicar debido a que ambos son árboles de decisión por lo que se comportan de manera similar. Se concluye al igual que el clasificador Random Forest que no es adecuado para este proyecto debido a la cantidad de FP presentes.

5.1.6.2.1.4. Näive Bayes

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	0	29
MALIGNA	36000	24639

Indicador	FP	FN	Recall	Especificidad
Valor	29	36000	0	0.998

Este clasificador presenta un número muy bajo de FP pero se tiene un gran número de FN. Se tiene un Recall de 0 (no se clasificó adecuadamente ninguna petición BENIGNA) y una especificidad buena. Se concluye con estos indicadores que aunque se presenta un alto número de FN este error es molesto en la seguridad pero no es muy grave, así que este clasificador podría servir para este proyecto, se espera la comparación final con los demás indicadores.

Los resultados finales del experimento 1 se presentan en el siguiente cuadro:

	Accuracy	*Kappa*	*F Score*
Baseline	0.539	-	-
Random Forest	0.602	0.025	0.748
KNN	0.558	0.030	0.668
C 4.5	0.602	0.025	0.748
Naïve Bayes	0.406	0	-

Tabla 4: Cuadro comparativo de resultados algoritmos de clasificación experimento 1.

Basados en los indicadores individuales de cada clasificador, se descartan como opción los clasificadores Random Forest, C 4.5 y KNN; esto se puede ratificar con los indicadores de la Tabla 4, en los cuales presentan una Precisión un poco más alta que el Baseline pero el Kappa demasiado bajo.

Quedando como última opción se tiene al clasificador Naïve Bayes, que en los indicadores individuales aunque se tuvo un gran número de FN se tenía una buena Especificidad, pero también es descartado debido a los indicadores de la tabla anterior, pues su Precisión es menor que el Baseline además de tener un Kappa de 0.

Se concluye que para el experimento 1 ningún clasificador será seleccionado, se tomará el Baseline como medida adecuada para la clasificación.

5.1.6.2.2. Experimento 2

A continuación, se presentan los indicadores mencionados en la sección previa para cada algoritmo de clasificación:

5.1.6.2.2.1. Random Forest

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	23242	542
MALIGNA	0	884

Indicador	FP	FN	Recall	Especificidad
Valor	542	0	1	0.619

Este clasificador presenta un número considerable de FP, que recordemos es algo muy grave en la seguridad, no presenta FN, tiene un Recall bastante bueno (de 1) y Especificidad alta. Puede ser considerado como clasificador evaluando los indicadores presentados en la tabla final.

5.1.6.2.2.2. KNN

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	16779	944
MALIGNA	6463	482

Indicador	FP	FN	Recall	Especificidad
Valor	944	6463	0.721	0.338

Este clasificador presenta un muy alto número de FP al igual que FN, cuenta con buen Recall y una Especificidad baja. Se concluye que no es un buen clasificador para este proyecto ya que presenta una gran cantidad de errores, en especial FP.

5.1.6.2.2.3. C 4.5

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	23242	542
MALIGNA	0	884

Indicador	FP	FN	Recall	Especificidad
Valor	542	0	1	0.619

Como en el experimento 1, el clasificador C 4.5 presenta resultados similares al clasificador Random Forest (recordemos que ambos son árboles de decisión). Se concluye al igual que el clasificador Random Forest que se tiene un número considerable de FP pero deberán evaluarse los indicadores en la tabla final.

5.1.6.2.2.4. Näive Bayes

Valores reales

Valores predichos	BENIGNA	MALIGNA
BENIGNA	29	0
MALIGNA	23213	1426

Indicador	FP	FN	Recall	Especificidad
Valor	0	23213	0.001	1

Este clasificador no presenta ningún FP (error grave en a seguridad) pero presenta una cantidad muy grande de FN (no tan grave como FP pero sigue siendo un error molesto). Se tiene una Especificidad excelente (de 1) pero un Recall bastante malo (0.001). Se concluye que deben ser evaluados los indicadores de la tabla final para tomar una decisión

Los indicadores finales del experimento 2 se presentan en el siguiente cuadro:

	Accuracy	*Kappa*	*F Score*
Baseline	0.539	-	-
Random Forest	0.978	0.754	0.988
KNN	0.699	0.021	0.819
C 4.5	0.978	0.754	0.988
Naïve Bayes	0.059	0.001	0.002

Tabla 5: Cuadro comparativo de resultados algoritmos de clasificación experimento 2.

Con base en los indicadores presentados para cada clasificador se descarta el clasificador KNN, puesto que presenta un gran número de FP, además se ratifica esto con los indicadores de la Tabla 5, en el cual aunque se tiene una Precisión un poco más alta que el Baseline se tiene un Kappa muy bajo (0.021).

El clasificador Naïve Bayes también se descarta, aunque no posee FP sí se tienen gran cantidad de FN, esto puede ser molesto pero con base en los indicadores de la tabla anterior se toma la decisión de descartarse como clasificador adecuado para este proyecto, puesto que la Precisión es muy mala (0.059, es decir, el Baseline es mejor) al igual que el Kappa (0.001).

Finalmente se analizan los clasificadores Random Forest y C 4.5, ambos presentan una cantidad considerable de FP (cerca del 38% del total de peticiones MALIGNAS fueron clasificadas como BENIGNAS) pero basado en los indicadores de la tabla anterior poseen una muy buena Precisión (0.978) al igual que un Kappa bueno (0.754).

Se concluye para el experimento 2 que cualquiera de estos dos clasificadores son los indicados para este proyecto, despreciando los FP son clasificadores que presentan muy buenos resultados.

5.2. Propuesta de sistema de software para el estudio de ciberataques web

Como un componente adicional al desarrollo de un método de detección de ataques web. Se tiene como objetivo realizar un sistema que permita el estudio de ataques web, para esto se realizó un proceso de ingeniería de software empezando por la licitación y finalizando en la etapa de diseño como trabajo a futuro se propone iniciar un desarrollo del software propuesto.

Para la licitación de los requerimientos se realizaron tres sesiones con nuestro tutor que para sentidos prácticos fue nuestro cliente. La primera sesión se realizó un esquema preliminar del sistema y las dos sesiones siguientes se utilizaron a modo de validación y retroalimentación para el proceso. De este proceso se tiene un documento que puede ser visto en el anexo (Anexo 7) y tiene como fin esclarecer los requerimientos iniciales del proyecto.

Posteriormente se realizó un proceso de análisis de requerimientos. Para esto se utilizó el método Dorfman para realizar una subdivisión y sub-especificación de estos. Este análisis tuvo como fruto una división en 9 categorías y un total de 25 requerimientos. El análisis completo puede ser visto en los documentos anexos (Anexo 8). Una vez finalizado este proceso se realizó a modo de complemento una serie de casos de uso que pueden ser vistos en el anexo (Anexo 9).

Como definición de requerimientos no funcionales se tuvieron en cuenta los siguientes: seguridad, mantenibilidad, modificabilidad, accesibilidad y usabilidad, al ser una propuesta inicial del proyecto no se tuvieron en cuenta requerimientos no funcionales organizacionales.

Para el diseño global se realizaron dos iteraciones, en una se desarrolló la propuesta del modelo de datos y en la segunda se realizó el diagrama de *deployment*. Ambos de estos artefactos pueden ser encontrado en los anexos (Anexo 10). Para el diagrama de *deployment* se tuvo en cuenta un patrón cliente-servidor para facilitar el acceso al software, además provee herramientas para la escalabilidad en casos de ser necesaria.

6. CONCLUSIONES Y TRABAJO A FUTURO

La dificultad para encontrar herramientas de libre acceso dificulta la creación de nuevos métodos de seguridad. Aunque esto se realiza para prevenir a los atacantes de obtener información sensible sobre el funcionamiento de los métodos, esto también imposibilita a los investigadores de seguir un camino concreto cuando se quieren trabajar sobre los trabajos futuros propuestos. En nuestro caso aún es necesario realizar una mejora en la generación de información de forma automatizada, conseguir una base de información más robusta sobre ataques y expandir la cantidad de ataques que se pueden automatizar. Además, para ayudar con la investigación en el campo de seguridad proponemos un sistema que permita su estudio, de este modo y sin necesidad de ofrecer información sensible se pueden realizar pruebas acerca de la efectividad de los métodos y crear ambientes según lo requerido para un proyecto específico.

Dentro del análisis que realizamos encontramos que los *honeypots* existentes tienen un tiempo de vida corto y no necesariamente se ajustan a las necesidades de investigación. Esto se da porque son herramientas utilizadas para un estudio específico y desarrollados con ese único fin, aunque esto es una aproximación que ha dado resultados aún no se ha logrado establecer una herramienta estandarizada que satisfaga las necesidades de los investigadores para obtener información. En nuestro caso específico no pudimos utilizar estas herramientas debido a su especificidad, baja cantidad de configuraciones e incapacidad de generar una captura de la información. Por esto creemos en la necesidad de generar una herramienta que permita satisfacer las necesidades de los investigadores al momento de obtener información sobre los ataques respectivos.

Adicionalmente al momento de buscar acerca de *payloads* de *XSS* y *SQLI* encontramos existen diferentes herramientas para realizar este tipo de ataques y de este modo obtener la información necesaria pero debido a la necesidad de automatización no pudieron ser utilizadas. La solución que encontramos fue buscar listas de *payloads* durante este proceso encontramos diferentes fuentes no oficiales las cuales fueron utilizadas para esta investigación. Debido a la importancia que tiene esta información es necesario poder tener una fuente confiable que valide la veracidad acerca de si lo expuesto y que regule el acceso únicamente para realizar los estudios.

Por otra parte, los resultados presentados en la validación de los métodos de detección no fueron los mejores, esto se le puede atribuir a la diversidad que tiene el *dataset* de entrenamiento seleccionado, puesto que de las 25000 peticiones anómalas muchas de estas correspondían a comportamiento anómalo en una aplicación (por ejemplo, insertar texto en un campo para el número de teléfono) y otros ciberataques los cuales no están definidos en el alcance de este proyecto. También incluía ciberataques XSS y SQL Injection pero estos eran una parte

mínima del total del *dataset*, por lo que los algoritmos de detección entrenados con el *dataset* que se generó (generado únicamente con muestras de ciberataques XSS y SQL Injection) no estaban preparados para detectar las demás anomalías en el *dataset*, dando como resultado unos indicadores no tan buenos, aunque se superó el Accuracy del baseline.

Cuando realizamos el experimento con el *dataset* procesado por nuestro algoritmo de extracción de características, se obtuvieron muy buenos resultados. Esto demuestra que las características para los ataques *SQLI* y *XSS* son idóneas y además los métodos de detección de *machine learning* son efectivos al momento de realizar estas clasificaciones.

Como se habla de la ciencia de los datos, se propone como trabajo a futuro una retroalimentación a los algoritmos de clasificación propuestos en este proyecto, mejorando las características de clasificación y ajustándolas no solo a los dos ciberataques aquí trabajados sino hacia diferentes tipos de ciberataques. Adicionalmente realizar mejoras en el algoritmo de extracción para reducir la ocurrencia de falsos positivos.

Para finalizar, se extiende la invitación a conocer todo el trabajo realizado por el grupo de investigación i2t de la Universidad Icesi.

7. CONOCIMIENTOS APRENDIDOS EN LA CARRERA UTILIZADOS EN ESTE PROYECTO

Asignatura	Conocimiento
COE	Escritura
Algoritmos y programación	Elaboración de algoritmos
Ingeniería de software	Recabación y análisis de requerimientos
Arquitectura de software	Análisis y diseño basado en atributos de calidad
Sistemas operativos	Despliegue, programación y utilización de servicios. Virtualización
Análisis y visualización de grandes conjuntos de datos	Entendimiento de cómo manejar los datos, al igual que la correcta manera de visualizarlos
Bases de datos	Diseño de modelo de datos
Redes	Comportamiento de protocolos de red. Uso de herramientas tipo sniffer.
Seguridad	Técnicas de prevención. Pentesting.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] S. Maconachy and W. Ragsdale, “A Model for Information Assurance: An Integrated Approach,” Proc. 2001 IEEE Work. Inf. Assur. Secur. US Mil. Acad. West Point, NY, pp. 5–6, 2001.

[2] E. Corchado and Á. Herrero, “Neural visualization of network traffic data for intrusion detection,” in Applied Soft Computing Journal, 2011, vol. 11, no. 2, pp. 2042–2056.

[3] C. Kruegel and G. Vigna, “Anomaly detection of web-based attacks,” in Proceedings of the 10th ACM conference on Computer and communication security - CCS ’03, 2003, p. 251.

[4] D. Atienza, Á. Herrero, and E. Corchado, “Neural analysis of HTTP traffic for web attack detection,” in Advances in Intelligent Systems and Computing, 2015, vol. 369, pp. 201–212.

[5] D. Kaur and P. Kaur, “Empirical Analysis of Web Attacks,” in Procedia Computer Science, 2016, vol. 78, pp. 298–306.

[6] P. Seshagiri, A. Vazhayil, and P. Sriram, “AMA: Static Code Analysis of Web Page for the Detection of Malicious Scripts,” in Procedia Computer Science, 2016, vol. 93, pp. 768–773.

[7] K. Chandrasekar et al., “Internet Security Threat Report - April 2017,” ISTR, no. April, 2017.

[8] P. Next, P. June, N. Previous, and N. Previous, “The Impact of a Security Breach 2017,” no. June, 2017.

[9] I. Corona and G. Giacinto, “Detection of Server-side Web Attacks,” J. Mach. Learn. Res., vol. 11, pp. 160–166, 2010.

[10] T. Segaran, Programming Collective Intelligence: Building Smart Web 2.0 Applications. 2007.

[11] A.-A. Adebayo and S. A Onashoga, “A hybrid approach for detecting malicious web pages using decision tree and Naïve Bayes algorithms,” Georg. Electron. Sci. J. Comput. Sci. Telecommun., vol. 48, pp. 9–17, 2016.

[12] I. Alexa Internet, “Alexa.” [Online]. Available: https://www.alexa.com/.

[13] OpenDNS, “PhishTank.” [Online]. Available: http://www.phishtank.com/.

[14] R. Wang, Y. Zhu, J. Tan, and B. Zhou, “Detection of malicious web pages based on hybrid analysis,” J. Inf. Secur. Appl., vol. 35, pp. 68–74, Aug. 2017.

[15] “Malware Domain List.” [Online]. Available: http://www.malwaredomainlist.com/mdl.php.

[16] C. C. Urcuqui, M. G. Peña, J. L. O. Quintero, and A. N. Cadavid, “Antidefacement,” Sist. Telemática, vol. 14, no. 39, pp. 9–27, Feb. 2017.

[17] J. Saxe and Invincea Labs, “Why security data science matters and how it’s different: pitfalls and promises of data science-based breach detection and threat intelligence,” in Black Hat USA, 2015.

[18] C. Urcuqui, A. Navarro, J. Osorio, and M. García, “Machine Learning Classifiers to Detect Malicious Websites,” 2017.

[19] N. Khan, J. Abdullah, and A. S. Khan, “Defending malicious script attacks using machine learning classifiers,” Wirel. Commun. Mob. Comput., vol. 2017, pp. 1–9, 2017.

[20] B. Salem and T. Karim, “Classification features for detecting server-side and client-side web attacks,” in IFIP International Federation for Information Processing, 2008, vol. 278, pp. 729–733.

[21] M. A. Wazzan and M. H. Awadh, “Towards improving web attack detection: Highlighting the significant factors,” in 2015 5th International Conference on IT Convergence and Security, ICITCS 2015 - Proceedings, 2015.

[22] D. R. Patil and J. B. Patil, “Detection of Malicious JavaScript Code in Web Pages,” Indian J. Sci. Technol., vol. 10, no. 19, pp. 1–12, 2017.

[23] M. Fraiwan, R. Al-Salman, N. Khasawneh, and S. Conrad, “Analysis and Identification of Malicious JavaScript Code,” Inf. Secur. J., vol. 21, no. 1, pp. 1–11, 2012.

[24] A. E. Nunan, E. Souto, E. M. Dos Santos, and E. Feitosa, “Automatic classification of cross-site scripting in web pages using document-based and URL-based features,” Proc. - IEEE Symp. Comput. Commun., pp. 000702–000707, 2012.

[25] H. Chauhan and A. Chauhan, “Implementation of decision tree algorithm c4. 5,” Int. J. Sci. Res. Publ., vol. 3, no. 10, pp. 4–6, 2013.

[26] G. Biau, “Analysis of a Random Forests Model,” J. Mach. Learn. Res., vol. 13, pp. 1063–1095, 2010.

[27] J. Han and M. Kamber, Data mining: Concepts and Techniques, Book. 2006.

[28] R. Kohavi, “A study of cross validation and bootstrap for accuracy estimation and model selection,” in 14th International Joint Conference on Artificial Intelligence (IJCAI), 1995, vol. 2, pp. 1137–1143.

[29] M. Kuhn et al., “Package ‘caret’ Classification and Regression Training Description Misc functions for training and plotting classification and regression models,” p. 213, 2017.

[30] G. Á. M. Carmen Torrano Giménez, Alejandro Pérez Villegas, “Http Dataset Csic 2010,” 2010. [Online]. Available: http://www.isi.csic.es/dataset/.

[31] J. Cohen, “A Coefficient of Agreement for Nominal Scales,” Educ. Psychol. Meas., vol. 20, no. 1, pp. 37–46, Apr. 1960.

[32] D. M. W. Powers and Ailab, “EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC, INFORMEDNESS, MARKEDNESS & CORRELATION,” J. Mach. Learn. Technol. ISSN, vol. 2, no. 1, pp. 2229–3981, 2011.

[33] Dewhurst Security, “DVWA - Damn Vulnerable Web Application,” 2016. [Online]. Available: http://www.dvwa.co.uk/.

[34] Selenium (Hrsg.), “Selenium - Web Browser Automation,” 2013. [Online]. Available: http://www.seleniumhq.org

[35] J. Rollins, “Why we need a methodology for data science,” IBM Big Data & Analytics Hub, 2015. [Online]. Available: http://www.ibmbigdatahub.com/blog/why-we-need-methodology-data-science.

9. ANEXOS

9.1. Anexo 1 – Cuadro comparativo Honeypot Server

A continuación, se muestra una tabla comparativa entre 17 diferentes Honeypots, los criterios de evaluación se definieron en: Última fecha de actualización, vulnerabilidades, tipo de interacción y soporte académico. Adicionalmente existe un campo de comentarios donde se destacan rasgos relevantes de cada Honeypot.

Las características deseables en los Honeypots son: emulación de servicios, captura de tráfico web, exposición de vulnerabilidades.

Tabla 6: Análisis comparativo Honeypots server.

Nombre	Última actualización	Vulnerabilidades	Tipo interacción	Soporte académico	Comentarios
LaBrea	2003-10-30	Gusanos informáticos.	No especificado	Citado en: Virtual Honeypots: from botnet tracking to intrusion detection. The honeynet project: Trapping the hackers.	Su última actualización se realizó hace 15 años. Su diseño está enfocado en obtener información acerca de gusanos informáticos lo cual no es el objeto de estudio del proyecto.

				LaBrea: “Sticky” Honeypot and IDS	
HoneySink	2011-08-31	Sink	No especificado	No encontré	Es un Honeypot tipo sink Los Honeypot tipo sink sirven para detener el flujo de tráfico que se dirige a páginas maliciosas. No representa mayor utilidad para la captura de información. No tiene características relevantes para nuestro proyecto.
KFSensor	Soporte actual	Emulación de servicios	Alta	KFSensor overview 3 citas. Diferentes artículos, pero con pocas citaciones.	Capacidad de emular servicios, Capturar tráfico, Enviar alertas y genera reportes. Es un Honeypot bastante completo tiene todas las características necesarias para el desarrollo del proyecto

Honeyd	2008-12-4	Kuang2 mydoom telnet emul pop3 iis ftp smpt	No especificado	Developments of the honeyd virtual Honeypot citado 39 veces	Está diseñado para vulnerabilidades importantes en años pasados. No tienes soporte y no está actualizado. No tiene los requerimientos necesarios para nuestro proyecto.
UDPot	2014-3-18	Sinkhole DSN emulator	Configurable	No se encontraron citaciones	Honeypot simple. Funciona como sinkhole y emulador de DNS
Conpot	2018-1-11	HTTP SMPT MODBUS	Baja	Conpot ics/scada Honeypot 8 citaciones	Desarrollado bajo HoneyProject por lo tanto tiene un soporte de investigación. Aún se encuentra en desarrollo y tiene módulos para emular diferentes servicios. Permite captura del tráfico. Falta algunas características para el desarrollo del proyecto

Bifrozt	2016-3-17	No especificado	No especificado	No se encontraron citaciones	No existe documentación acerca de este Honeypot.
Beeswarm	2017-1-17	Observa la diferencia entre tráfico esperado y tráfico actual	No especificado	No se encontraron citaciones	El proyecto fue descontinuado y su página se encuentra caída por lo cual no se pudo obtener más información.
Bait and Switch	2003-09-28	Redirecciona el tráfico malicioso hacia el Honeypot.	No especificado	No se encontraron citaciones	Enfocado a un ámbito empresarial que busque generar información para I&D.
Artillery	2017-12-19	Combinación entre Honeypot y sistema de detección. Monitorea más que generar información	No especificado	No se encontraron citaciones	Sistema de monitoreo. Linux

Slimp-Honeypot	2013-5-27	Captura información de puertos y tráfico en general.	Baja	No se encontraron citaciones	Se basa en un sistema simple de escucha.
HoneyWRT	2015-4-14	Remote Desktop Protocol (RDP) (TCP/3389) Virtual Network Computer (VNC) (TCP/5900) Fake Shoutcast Server (TCP/8000) Tomcat Admin Page /manage/html (TCP/8080) Microsoft SQL Server (MSSQL) (TCP/1433) Fake Telnet Server (TELNET) (TCP/23)	Baja	No se encontraron citaciones	Realizado en Python, no tiene las características principales que se buscan para el desarrollo en este proyecto.

Amun	2014-03-21	Diferentes módulos de vulnerabilidades	No especificado	Citado en 20 artículos diferentes en Google Scholar	Hecho en Python, no tiene la documentación completa y se encuentra descontinuado.
TelnetHoney	2016-01-20	Telnet	No especificado	No se encontraron citaciones	No tiene documentación. Honeypot dedicado a telnet
Hontel	2017-11-29	Honeypot para telnet	No especificado	No se encontraron citaciones	Honeypot dedicado a telnet. No cumple con las características para nuestro proyecto
MTPot	2017-03-20	Especializado para mirei malware	No especificado	No se encontraron citaciones	Es un Honeypot diseñado con características específicas para un virus específico, no cumple con las características para nuestro proyecto.

Heralding	2018-1-21	Especializado en captura de credenciales	No especificado	No se encontraron citaciones	Específico para credenciales.

9.2. Anexo 2 – Cuadro comparativo Honeypot Web

Luego de hacer una revisión de los Honeypots server seleccionados, notamos que estos no satisfacen las necesidades planteadas para el proyecto porque no permitían una emulación completa del servicio web, que es nuestro principal enfoque. Al no poder emular completamente el servicio web y sus vulnerabilidades no nos permite obtener un *dataset* con todas las características necesarias. Al encontrar estas limitaciones iniciamos una búsqueda de un Honeypot web que cumpla con las características de emulación de vulnerabilidades de aplicaciones web y tenga la capacidad de caracterizarlas.

Características deseables:

- Emulación de vulnerabilidades como XSS, SQL Injection y si es posible XEE.
- Tener la capacidad de clasificar un ataque a este tipo de vulnerabilidad.
- Tener la posibilidad de exportar la captura del tráfico a archivo PCAP o Alguno similar que permita realizar la conversión.

Tabla 7: Análisis comparativo Honeypots web.

Nombre	Fecha Actualización	Vulnerabilidades	Tipo interacción	Comentarios
Gastopf	4/03/2018	Remote File Inclusion Local File Inlcusion HTML Injection SQL Injection	No especificado	A pesar de que su ultimo commit es reciente el proyecto como tal no tiene actualizaciones recientes. Queda pendiente revisar la parte de captura de trafico

Snare/Tanner	9/03/2018	Creación de clones de páginas Falta mirar cómo funciona las vulnerabilidades	No especificado	Es un Honeypot que tiene actualizaciones recientes, dice ser el sucesor de Gastopf. Queda pendiente revisar cómo funciona y como se ejecuta la captura de trafico
Phpmyadmin_Honeypot	3/07/2015	Emulación de administrador de PHP	No especificado	Enfocado a otro tipo de vulnerabilidad
Nodepot	23/08/2015	Joomla attack (SQLInjection) Technote 7 (Remote File Inclusion) OpenFlash (File upload) SQL Injection Joomla JCE (Cross-Site Scripting)	No especificado	Falta revisar los métodos de captura de tráfico. Posee elementos de tecnologías con vulnerabilidades
basic-auth-pot	15/01/2015	Autenticación flaws	No especificado	Enfocado en autenticación

Shadow Daemon	3/02/2018	SQL injections XML injections Code injections Command injections Cross-site scripting Local/remote file inclusions Backdoor access	Alta interacción	Vale la pena revisar es bastante completo. Hay que complementar con una aplicación vulnerable
Serveltpot	12/05/2013	No especificado	No especificado	Viejo, incompleto
Google Hack Honeypot	07/04/2007	Ataques mediante servidores de búsqueda	No especificado	Viejo, no se adapta a nuestro problema
Smart-Honeypot	18/04/2014	No especificado	No especificado	Falta información, parece ser muy simple
Bukkit Honeypot	12/05/2011	Bukkit	No especificado	Desarrollado para Bukkit
Laravel Aplication Honeypot	9/05/2017	Spam Prevention	No especificado	Desarrollado para prevenir spam

Stack-Honeypot	30/1/2014	Spam Prevention	No especificado	Diseñado para prevenir spam
EoHoneypotBundle	26/4/2016	Spam Prevention	No especificado	Diseñado para prevenir spam
ShockPot	17/12/2015	CVE-2014-6271.	No especificado	Exploit de bash
Django-admin-Honeypot	7/08/2016	Fake Django admin screen	No especificado	Diseñado para simular un login de Django
WebTrap	1/1/2018	Login Problems	No especificado	Clona páginas reales para timar a los hackers
honeyhttpd	21/1/2018	No especificado	No especificado	Captura los datos de las peticiones realizados a servidores web. soporta Python 2 y 3

9.3. Anexo 3 – Cuadro comparativo aplicaciones vulnerables

Para complementar a los Honeypots web que se van a analizar es necesario tener una aplicación vulnerable que permita que los Honeypots se encarguen del análisis y clasificación. Por lo tanto, se realizarán comparaciones entre las aplicaciones vulnerables en las cuales se realizan pen-testing.

Nombre	Fecha Actualización	Vulnerabilidades	Lenguaje	Comentarios
BTSLab	1/12/2016	SQL Injection XSS CSRF Clickjacking SSRF File Inclusion Code Execution Insecure Direct Object Reference Unrestricted File Upload Open URL Redirection SSI injection	PHP	Vale la pena analizar
Bodge It	8/1/2018	XSS SQL Injection CSRF Insecure Object References	Java	Ya no se trabaja en ella

Juice Shop Project	9/03/2018	No especificado	Node.js	Amplia variedad de vulnerabilidades, pero no especificadas. Se basa en gamificación.
Bricks	30/11/2013	SQL Injection	PHP MySQL	Enfocado en SQL Injection Muy viejo
bWAPP	4/11/2014	Over 100 bugs	PHP MySQL	Tiene muchas vulnerabilidades
DVWA	12/2/2018	Vulnerabilidades más comunes	PHP MySQL	Bastante documentada y al parecer usada

9.4. Anexo 4 – Características de detección

- **Información general de la petición:**

Nombre	**Descripción**	**Tipo**	**Ataque asociado**
Req-length	Longitud de la petición	Entero positivo	Ataques de Buffer Overflow
URI-length	Longitud URI	Entero positivo	Buffer Overflow,

			URI decoding
Req-method	Tipo de la petición (GET, POST, HEAD, …)	Simbólico	-
Req-resource-type	Tipo de recurso solicitado (HTML, asp, PHP, …)	Nominal	-
Num-param	Número de parámetros	Entero positivo	Validación de entradas
Num-arg	Número de argumentos	Entero positivo	Validación de entradas
Is-req-correct	¿La petición se ajusta al protocolo HTTP?	Booleano	Anomalías de la URL, URL decoding

- **Contenido de la petición:**

Nombre	**Descripción**	**Tipo**	**Ataque asociado**
Num-NonPrintChars	Número de caracteres especiales y Shell codes en la petición HTTP (x86, carriage return, punto y coma)	Entero positivo	Buffer Overflow, códigos Shell, URL decoding
SQL-cmd-tricks	¿La petición contiene comandos SQL (“--,OR 1==1, …)?	Booleano	Inyección SQL

Shell-cmds	¿La petición contiene comandos Shell (Todos los comandos Shell de los sistemas operativos)?	Booleano	Inyección de comandos
Sensitive-files	¿La petición contiene referencias a información sensible (etc/passwd, …)?	Booleano	Fuga de información, accesos no autorizados
Directory-traversal	¿La petición contiene trucos de recorrido de directorio (presencia de token como “../”, …)?	Booleano	Recorrido de directorio
Oversized-values	¿La petición contiene potencialmente valores numéricos sobredimensionados?	Booleano	Malinterpretación del valor
Default-login-passwd	¿La petición incluye usuarios y contraseñas predeterminados de fábrica (guest, anonymous, root, admin, …?	Booleano	Ataques de diccionario, fuerza bruta, ...
Script-injection	¿La URI contiene un tag de script (“<script”, “<meta”, …)?	Booleano	Cross Site Scripting

- **Respuesta del servidor:**

Nombre	**Descripción**	**Tipo**	**Ataque asociado**
Resp-code	Código de respuesta de la petición HTTP (200, 404, 500,	Nominal	-

	…)		
Is-html-Response	¿La respuesta es un archivo HTML?	Booleano	-
Response-time	Tiempo transcurrido desde la correspondiente petición HTTP	Real	DoS
Script-type	El tipo de script incluido en la respuesta (Java, Visual Basic, …)	Nominal	Cross Site Scripting
Writing-script	¿El flujo de respuesta incluye script de funciones de escritura (document.write(), …)?	Booleano	Fijación de ID de sesión, ...

- **Historial de solicitudes:**

Nombre	**Descripción**	**Tipo**	**Ataque asociado**
Num-Req-Same-Host	Número de peticiones emitida por la misma fuente	Entero positivo	Flooding, escaneo de vulnerabilidades
Num-Req-Same-URL	Número de peticiones con la misma URL	Entero positivo	Flooding de la misma fuente/múltiples fuentes.
Num-Req-Same-Host-Diff-URI	Número de peticiones emitida por la misma fuente y solicitando URL diferentes	Entero positivo	Escaneo de vulnerabilidades

Inter-Req-Interval	Intervalo de tiempo de la solicitud Inter	Entero positivo	Flooding, escaneo de vulnerabilidades

9.5. Anexo 5 – Características de detección 2

- **URL**

Nombre	**Descripción**
Navegación forzada	URL ilegal que permita el acceso a la página forzando una URL.
Violación de acceso	URL contiene caracteres no permitidos o un tiempo de acceso válido expirado.
Hijack a la sesión	URL contiene ID de la sesión inválida La URL contiene un valor de ID de sesión que es igual a la ID de sesión que el servidor estableció para esta sesión.
Overflow del buffer	Longitud de la URL inaceptable

- **Método**

Nombre	**Descripción**
Pérdida de información	Utilizar métodos HTTP ilegales en la petición o métodos SOAP no permitidos.

Violación RFC	GET o HEAD contienen body.
Overflow del buffer	Longitud de la petición POST ilegal.
Violación de entrada	La petición contiene Sting de consulta o datos POST no permitidos.
HTTP Request Smuggling	Longitud de cero en el contenido POST

- **Host**

Nombre	**Descripción**
Cliente no navegador	Cabecera de host no existe en las peticiones HTTP 1.1. Cabecera de host contiene dirección IP.

- **Parámetros**

Nombre	**Descripción**
Violación de acceso	Los parámetros de la petición contienen caracteres ilegales.
Manipulación de parámetros	La solicitud contiene un tipo de datos de parámetro ilegal, contiene un valor numérico de parámetro que no está en el rango permitido, contiene un valor de parámetro estático ilegal o contiene un valor de parámetro alfanumérico que no cumple con el campo de expresión

	regular.
Violación de entrada	La solicitud contiene un parámetro dinámico cuyo valor cambió a vacío o el valor contiene un meta carácter ilegal, contiene un número ilegal de parámetros obligatorios o puede contener un parámetro no permitido. La longitud del valor del parámetro no está permitida o la solicitud de varias partes tiene un parámetro con un valor NULL.
HTTP parameter pollution attack	La petición contiene el mismo nombre que muchos nombres de parámetros o contiene decodificación múltiple para el URI o el valor del parámetro.

- **Cookies**

Nombre	**Descripción**
HTTP parser attack	La header de la cookie no es compatible con RFC.
Violación de acceso	Cookie de sesión CSRF inyectada en la respuesta.
Overflow del buffer	La petición contiene una longitud de la cookie ilegal.
Violación de cookie	La petición tiene un dominio de cookie modificado no permitido.

- **Referrer URL**

Nombre	Descripción
Ataque automatizado	El header del referrer contiene una referrer URL no identificada.

- **Información delicada**

Nombre	Descripción
Pérdida de información	La respuesta contiene información delicada como información de tarjetas para los pagos.

- **Archivo sospechoso**

Nombre	Descripción
Virus	La petición incluye un archivo que contiene un gusano o un virus.

- **Patrón sospechoso**

Nombre	Descripción
Firma de ataque	La petición o la respuesta contiene un patrón que concuerda con la firma de un ataque.

- **IP**

Nombre	**Descripción**
Ataque DoS	La petición contiene una IP desconocida o que esté en la lista de bloqueados.

9.6. Anexo 6 – Características de detección 3

Number_Of_Redirections	Difference_In_Redirections	Number_Of_Instantiated_Objects	Difference_In_Instantiated_Objects	Lenght_Of_Longest_Single_Evaluated_Code
Total_Bytes_Of_Evaluated_Code	Total_Bytes_Allocated	Number_Of_Executions	Length_Of_Longest_Unprintable_String	Number_Of_Unprintable_Strings
Ratio_Of_Definitions_To_Uses	Lenght_Of_Method_Call_Parameter	Total_Number_Of_Method_Calls	Memory_Overflow_Ocurred	Unprintable_String_Used_For_Instantiated_Oject_Parameter
Unprintable_Strings_AND/OR_Redirects	Method_Cal	open	run	Savetofile
Send	Setattribute	Write	timespanformat	shellexecute
playerproperty	uploadlogs	hgs_startnotify	downloadandinstall	getvariable

linksbicons	openurl	getresponsehea der	keyframe	setrequesthe ader
Setslice	buildpath	getspecialfolder	environment	rawparse
iestartnative	setformatlikesa mple	createobject	specialfolders	replace
createnewfold ername	addevent	isversionsupport ed	split	new
evaluate	msdatasourceo bject	allowscriptacces s	concat	url
mode	type	Import	close	allowcontext menu
cachefolder	compressedpat h	console	printsnapshot	shownavigati onbuttons
snapshotpath	wkspictureinterf ace	zoom	script_Size	intent

9.7. Anexo 7 – Documento de requerimientos

Se requiere un sistema que tenga capacidad para realizar estudios de ataques web. Para esto debe permitir crear proyectos, cada proyecto tendrá un sistema controlado (Honeypot) el cual será objeto de estudio, dependiendo del objetivo del estudio se podrá probar la efectividad de un algoritmo de detección entrenado con un determinado *dataset* o generar el mismo mediante emulación de ataques y captura de ataques reales. Es importante para esta parte que los ataques que sean realizados puedan ser plenamente identificados para su estudio.

El sistema debe generar reportes de los estudios realizados. Estos reportes deben ser presentados en tablas y gráficos mostrando comparaciones entre los resultados obtenidos por los diferentes *datasets* usados o los algoritmos probados. Estos reportes deben ser sobre: porcentajes de error como falsos positivos, falsos

negativos y aciertos. e información adicional dependiendo de lo que se haya testeado.

El sistema debe permitir el manejo de usuarios y roles. Existen tres roles principales: Administrador del sistema, investigador y usuario común de Sniff. El administrador del sistema podrá modificar cualquier parámetro del sistema, el investigador podrá realizar las pruebas en los ambientes creados y realizar las modificaciones a los proyectos de los cuales haga parte. El usuario común de Sniff no podrá realizar modificaciones en estos ambientes y solo podrá observar los reportes generados.

El sistema debe tener un sistema de alertas de ataques. Cuando la aplicación se encuentre bajo un ataque de “cualquier tipo” debe lanzarse una alerta al administrador del sistema con la información del ataque, tipo de ataque, recursos afectados, que está ocurriendo para que haga las labores necesarias para proteger al sistema del ataque. Adicionalmente se debe realizar una captura de todo el tráfico que ocurra durante el ataque para su posterior análisis y adición a los entornos de investigación.

Estudio ataques web:

- El sistema debe permitir crear un proyecto con: nombre, colaboradores, fecha de creación, autor, estado y tipo de estudio.

- El sistema debe permitir adicionar y remover colaboradores de un estudio.

- El sistema debe permitir modificar el estado del proyecto

- El sistema debe permitir modificar el tipo de algoritmo de clasificación y su método de entrenamiento.

- El sistema debe permitir modificar el *dataset* con el cual se entrenó el algoritmo de clasificación

- El sistema debe permitir modificar el ambiente controlado para añadir o remover vulnerabilidades

- El sistema debe permitir volver a un estado inicial de un ambiente controlado

- El sistema debe permitir realizar ataques a un ambiente controlado mediante un simulador de ataques

- El sistema debe permitir configurar los parámetros del simulador de ataques, tipo de vulnerabilidades y frecuencia.

Reportes:

- El sistema debe generar reportes comparativos con los cambios realizados en los proyectos: indicadores estadísticos para la evaluación de estos (aciertos, falsos positivos, falsos negativos).

- El sistema debe generar reportes de los ataques que se hayan realizado a la aplicación.

- El sistema debe generar reportes de funcionamiento del sistema de detección

Sistema de Roles:

- El sistema debe permitir crear roles, modificar y eliminar roles.

Usuarios:

- El sistema debe permitir crear: nombre, correo, contraseña, rol, estado y modificar un usuario

- El sistema debe permitir modificar un rol a un usuario.

- El sistema debe permitir modificar la contraseña de un usuario

- El sistema debe permitir cambiar el estado de un usuario

- El sistema debe permitir limpiar una contraseña mediante un correo.

Servicio de análisis:

- Capacidad de analizar una captura de tráfico web para determinar si pertenece a un ataque y a qué tipo.

Administración del sistema

- El sistema debe poder modificar sus parámetros.

Requerimientos no funcionales:

- Escalable
- Interoperable
- Mantenible
- Modificable
- Parametrizable
- Seguro

9.8. Anexo 8 – Análisis Dorfman

Debido al formato del documento, en este link se podrá descargar en extensión PDF el análisis.

9.9. Anexo 9 – Casos de uso

Campo	Descripción
Identificador	**CUU01**
Nombre	Generar Reporte Resultados
Actor Principal	Usuario Común

Objetivo en Contexto	Generar un reporte de resultados de un archivo PCAP proporcionado por el usuario determinando si la captura pertenece o no a un ciberataque
Precondiciones	El sistema tiene activo un método de detección de ciberataques
Disparador	El usuario común hace uso de la funcionalidad
Escenario	1. Usuario: activa la función 2. Usuario: Sube un archivo de tipo PCAP 3. Sistema: Analiza el archivo 4. Sistema: Genera un reporte con los ataques encontrados si se encontraron 5. Usuario: Descarga el reporte generado
Excepciones	1. El archivo subido no es de formato PCAP 2. Mensaje de error indicando que el formato no es compatible
Prioridad	Deseable

Campo	**Descripción**
Identificador	**CUI01**
Nombre	Crear Proyecto

Actor Principal	Investigador
Objetivo en Contexto	Crear un proyecto de investigación para que tenga todas las características necesarias
Precondiciones	El investigador debe haber ingresado al sistema con éxito
Disparador	El investigador inicia la función
Escenario	1. El investigador inicia la función 2. El investigador diligencia el nombre y tipo de estudio 3. El sistema agrega un nuevo proyecto 4. Inicia el caso de uso CUI0 5. Inicial el caso de uso CUI0
Excepciones	1. El nombre del proyecto ya existe 2. Mensaje de excepción indicando que el nombre de proyecto ya existe
Prioridad	Importante

Campo	**Descripción**
Identificador	**CUI02**
Nombre	Login

Actor Principal	Investigador
Objetivo en Contexto	Permitir al investigador acceder a las funciones del sistema
Precondiciones	El investigador posee una cuenta válida
Disparador	El investigador intenta ingresar al sistema
Escenario	1. El investigador ingresa al sistema 2. El investigador ingresa su usuario 3. El investigador ingresa su contraseña 4. El investigador ejecuta la función de ingreso 5. El sistema verifica los datos de acceso 6. El investigador ingresa al sistema
Excepciones	1. El usuario o contraseña son inválidos 2. El investigador tiene una cuenta que se encuentra deshabilitada 3. El sistema muestra un mensaje de error indicando los fallos por los cuales no se pudo ingresar
Prioridad	Importante

Campo	**Descripción**
Identificador	**CUI03**

Nombre	Editar Proyecto
Actor Principal	Investigador
Objetivo en Contexto	Permitir al investigador crear nuevas versiones del mismo proyecto o modificar los colaboradores de este
Precondiciones	El investigador ingreso correctamente al sistema
Disparador	El investigador inicia la función
Escenario	1. El investigador ejecuta la función 2. El investigador selecciona editar colaboradores 3. Inicial el caso de uso CUI0 4. El investigador selecciona crear nueva versión 5. Inicial el caso de uso CUI0 6. El investigador selecciona editar estado 7. El investigador selecciona el nuevo estado: en proceso, abandonado, terminado
Excepciones	1. El proyecto se encuentra en estado finalizado 2. El sistema lanza un mensaje indicando que el proyecto no se puede editar
Prioridad	Importante

Campo	Descripción

Identificador	CUI04
Nombre	Generar Reporte Investigación
Actor Principal	Investigador
Objetivo en Contexto	El investigador desea conocer los resultados de una versión del proyecto o su comparación con respecto a otra versión del mismo proyecto
Precondiciones	El investigador debe haber ingresado correctamente al sistema. Debe existir un proyecto con una versión.
Disparador	El investigador ejecuta la función
Escenario	1. El investigador ejecuta la función 2. El investigador selecciona el reporte de resultados de la versión actual 3. El investigador selecciona si desea comprar los resultados con otra versión 4. El sistema genera un reporte con los datos deseados 5. El investigador decide descargar o no los resultados
Excepciones	
Prioridad	Importante

Campo	Descripción
Identificador	**CUI05**
Nombre	Editar Participantes
Actor Principal	Investigador
Objetivo en Contexto	El investigador desea editar los participantes que se encuentran trabajando en un proyecto
Precondiciones	
Disparador	Se ha iniciado el caso de uso CUI03
Escenario	1. El investigador selecciona remover colaboradores 2. El investigador selecciona los colaboradores a remover 3. El sistema remueve los colaboradores seleccionados 4. El investigador selecciona adicionar colaboradores 5. El investigador ingresa los nombres de usuario de los colaboradores a agregar 6. El sistema adiciona los nuevos colaboradores
Excepciones	1. El nombre del usuario a agregar no existe 2. Mensaje de error indicando la inexistencia del usuario identificado con ese nombre de usuario.

Prioridad	Importante

Campo	Descripción
Identificador	**CUI06**
Nombre	Crear Versión
Actor Principal	Investigador
Objetivo en Contexto	Crear una nueva versión para un proyecto con características específicas para la investigación
Precondiciones	Existe un proyecto El investigador es colaborador del proyecto
Disparador	Caso de uso CUI01 Caso de uso CUI03
Escenario	1. El investigador diligencia la nueva versión con nombre y justificación. 2. El investigador selecciona configurar entorno 3. inicio caso de uso CUI0 4. El investigador selecciona configurar *dataset* 5. inicia caso de uso CUI0 6. El investigador selecciona configurar ciberataques 7. inicia el caso de uso CUI0

	8. El investigador selecciona configurar método de aprendizaje. 9. inicia el caso de uso CUI0
Excepciones	
Prioridad	Importante

Campo	Descripción
Identificador	**CUI08**
Nombre	Configurar Entorno Controlado
Actor Principal	Investigador
Objetivo en Contexto	Realizar las configuraciones necesarias al entorno como posibles vulnerabilidades, sistema operativo y servidores
Precondiciones	Debe existir una versión
Disparador	CUI06
Escenario	1. El investigador selecciona el sistema operativo del entorno

	2. El investigador selecciona los servidores disponibles que deban ser instalados en el entorno 3. El investigador selecciona qué vulnerabilidades debe ofrecer el entorno controlado
Excepciones	
Prioridad	Importante

Campo	Descripción
Identificador	**CUI09**
Nombre	Configurar *dataset*
Actor Principal	Investigador
Objetivo en Contexto	Configurar el *dataset* con el cual va a funcionar el detector de ataques
Precondiciones	Existe un entorno controlado
Disparador	CUI06

Escenario	1. El investigador selecciona un *dataset* existente o proporciona uno nuevo 2. El sistema carga el *dataset*
Excepciones	1. El *dataset* subido no cuenta con el formato especificado 2. Mensaje de error que muestre que el *dataset* es erróneo
Prioridad	Importante

Campo	**Descripción**
Identificador	**CUI10**
Nombre	Configurar Ciberataques
Actor Principal	Investigador
Objetivo en Contexto	Configurar los ciberataques que serán realizados al entorno controlado.
Precondiciones	Existe un entorno controlado
Disparador	CIU06
Escenario	1. El investigador selecciona los ataques que serán realizados

	2. El investigador configura los parámetros de los ataques seleccionados 3. El investigador selecciona la frecuencia de los ataques.
Excepciones	
Prioridad	Importante

Campo	Descripción
Identificador	**CUI11**
Nombre	Configurar Método de Aprendizaje
Actor Principal	Investigador
Objetivo en Contexto	Configurar el método de aprendizaje con el que funcionara el detector de ataques.
Precondiciones	Existe un *dataset* y un entorno controlado
Disparador	CUI06
Escenario	1. El investigador selecciona el algoritmo de detección

	2. El investigador selecciona el método de aprendizaje 3. El investigador selecciona la técnica de aprendizaje
Excepciones	1. El *dataset* no tiene las características necesarias
Prioridad	Importante

Campo	**Descripción**
Identificador	**CUI12**
Nombre	Crear Usuario
Actor Principal	Investigador
Objetivo en Contexto	Crear un nuevo usuario para acceso al sistema
Precondiciones	
Disparador	El administrador ejecuta la función
Escenario	1. El Investigador ingresa los datos del usuario: nombre, correo, rol y estado. 2. El sistema crea el usuario

Excepciones	1. El correo del usuario ya se encuentra registrado 2. La contraseña no cumple con los estándares 3.
Prioridad	Deseable

Campo	**Descripción**
Identificador	**CUI13**
Nombre	Editar Usuario
Actor Principal	Investigador
Objetivo en Contexto	Editar los datos del usuario
Precondiciones	El investigador se ha ingresado exitosamente
Disparador	El investigador ejecuta la función
Escenario	1. El investigador selecciona los datos a editar 2. El investigador diligencia los datos a editar 3. El sistema pide verificación de la contraseña 4. El investigador digita la contraseña 5. El sistema guarda los cambios

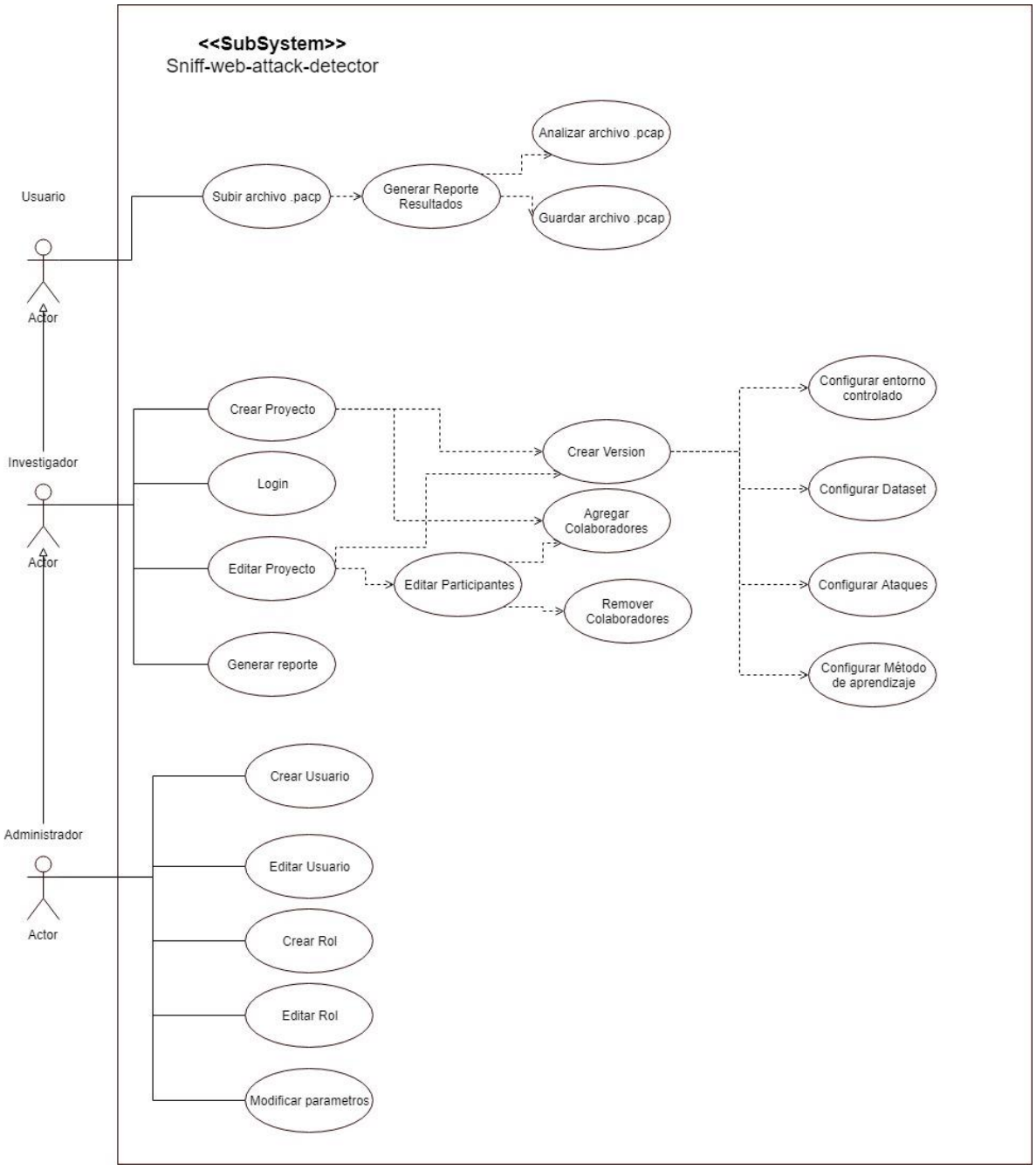
Excepciones	1. Los datos diligenciados no cumplen con el estándar 2. La contraseña es incorrecta
Prioridad	Deseable

Campo	**Descripción**
Identificador	**CUA01**
Nombre	Crear Rol
Actor Principal	Administrador
Objetivo en Contexto	Crear un nuevo rol en el sistema
Precondiciones	
Disparador	El administrador ejecuta la función
Escenario	1. El administrador ejecuta la función 2. El administrador diligencia la información del rol: nombre y descripción 3. El sistema crea el nuevo rol
Excepciones	1. El nombre del rol ya existe

Prioridad	Deseable

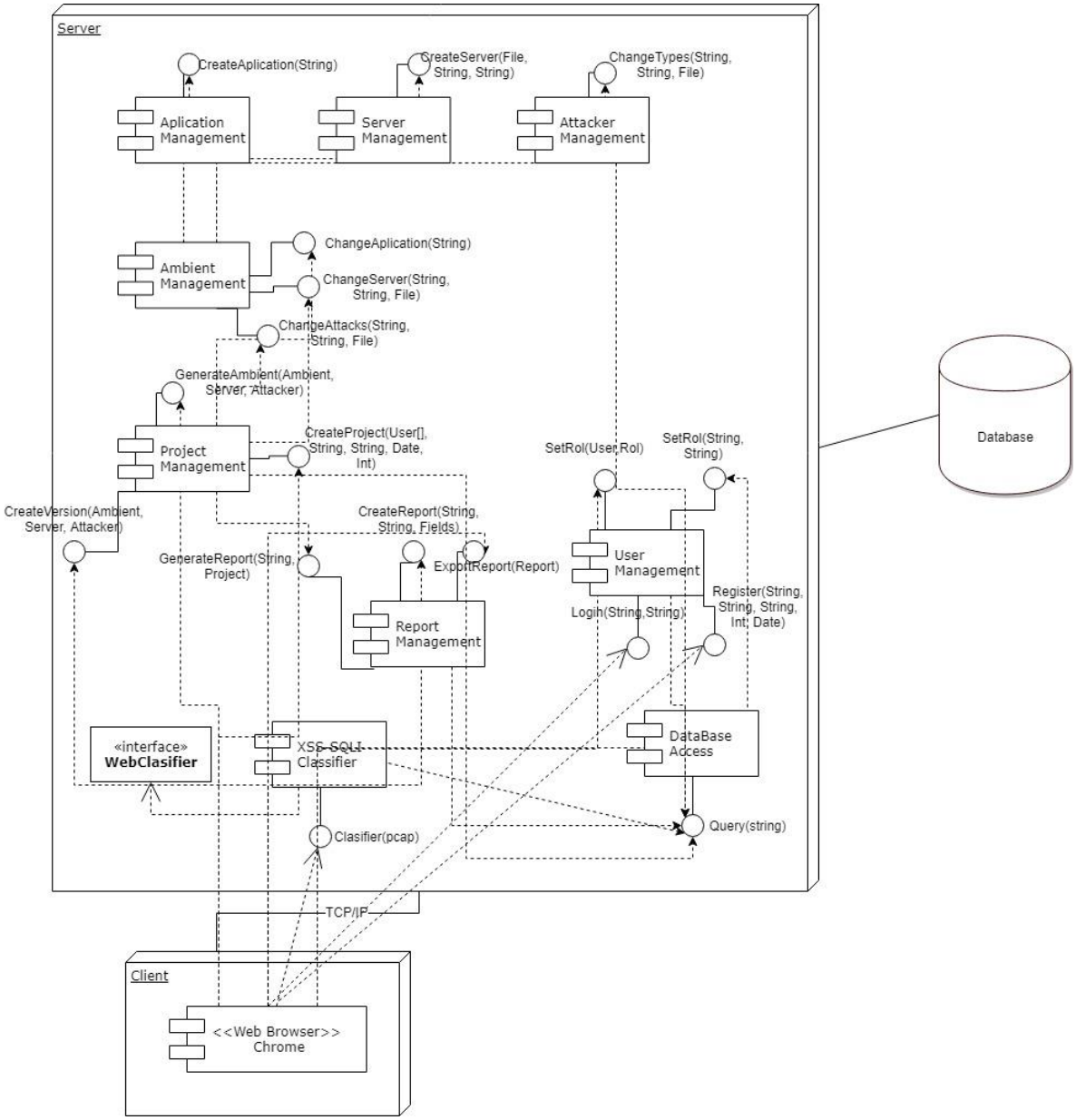
Campo	Descripción
Identificador	**CUA02**
Nombre	Editar Rol
Actor Principal	Administrador
Objetivo en Contexto	Editar un rol creado
Precondiciones	
Disparador	El administrador ejecuta la función
Escenario	1. El administrador ejecuta la función 2. El administrador diligencia los datos a editar 3. El sistema edita el rol
Excepciones	1. Los datos diligenciados no cumplen con los estándares
Prioridad	Deseable

Campo	Descripción
Identificador	**CUA03**
Nombre	Modificar Parámetros
Actor Principal	Administrador
Objetivo en Contexto	Modificar los parámetros del sistema
Precondiciones	
Disparador	El administrador ejecuta la función
Escenario	1. El administrador ejecuta la función 2. El administrador selecciona los parámetros a modificar 3. El administrador modifica los parámetros. 4. El sistema guarda los cambios
Excepciones	1. Los datos diligenciados no cumplen con el estándar
Prioridad	Deseable


<<SubSystem>>
Sniff-web-attack-detector
Usuario
Actor
Subir archivo .pacp
Generar Reporte Resultados
Analizar archivo .pcap
Guardar archivo .pcap
Investigador
Actor
Crear Proyecto
Login
Editar Proyecto
Generar reporte
Crear Version
Agregar Colaboradores
Editar Participantes
Remover Colaboradores
Configurar entorno controlado
Configurar Dataset
Configurar Ataques
Configurar Método de aprendizaje
Administrador
Actor
Crear Usuario
Editar Usuario
Crear Rol
Editar Rol
Modificar parametros

9.10. Anexo 10 – Diseño detallado

9.10.6. Diagrama de *deployment*



9.10.7. Modelo de datos