Programa de Doctorado en
Ingeniería de la Información y el Conocimiento

TÉCNICAS DE APRENDIZAJE AUTOMÁTICO APLICADAS A LA MEJORA DE DETECCIÓN DE ATAQUES EN APLICACIONES WEB.

Tesis Doctoral presentada por
TOMAS MIGUEL SUREDA RIERA

Directores:
Dr. Juan Ramón Bermejo Higuera
Dr. Javier Bermejo Higuera
Dr. José Javier Martínez Herráiz

Alcalá de Henares, 2022

Agradecimientos

"Si crees que puedes, ya estás a medio camino"
(Theodore Roosevelt)

Volviendo ahora la mirada atrás, en esta etapa final, en la que uno se sienta delante de un documento en blanco intentando plasmar en unas pocas líneas todo aquello que ha significado este proyecto: anhelos, ilusiones, preocupaciones, decepciones, cansancio, decisión, determinación… ahora es cuando todo este viaje cobra sentido; como diría *Kavafis* en su poema *Ítaca: "(…) Ten siempre a Ítaca en tu mente. Llegar allí es tu destino. Mas no apresures nunca el viaje. Mejor que dure muchos años y atracar, viejo ya, en la isla, enriquecido de cuanto ganaste en el camino (…)".*

Este viaje no hubiese sido posible sin el empuje y ánimo de los Doctores Juan Ramón y Javier Bermejo Higuera; ellos dos creyeron en mí y me prestaron su apoyo y confianza incondicional desde el primer momento, animándome a iniciar y perseverar en esta aventura cuando ni yo mismo tenía claro que pudiera llegar a buen puerto. Juan Ramón, Javier, sabéis que tenéis mi eterna gratitud. Gracias de todo corazón.

Gracias también al Doctor José Javier Martínez Herráiz por haberme brindado su ayuda y orientación en todas las etapas de la tesis, aceptando dirigir mi tesis y resolviendo todas mis dudas (que han sido muchas) dentro de los circuitos administrativos de la Universidad. José Javier, has conseguido que me

sienta parte de esta Universidad, a pesar de la distancia física que me separa de ella.

Mención especial merece el Doctor Juan Antonio Sicilia Montalvo, gracias al cual se desbloqueó la publicación de mi primer artículo como autor principal. Juan Antonio, si no llega a ser por ti, todo se hubiera demorado mucho más.

Nuria García Humanes navegó entre el texto de esta tesis buscando errores estilísticos y discursivos y aportando sugerencias de mejora. Mil gracias por tu ayuda, Nuria.

Finalmente, pero no menos importante, quisiera agradecer y dedicar esta tesis a mi familia. Mi mujer Fanny y mi hija Jessica han tenido que lidiar con mis altibajos, frustraciones, inseguridades, miedos y también alegrías soportando en muchos casos mis disertaciones crípticas sobre datos, algoritmos y modelos. Esta tesis es también vuestra tesis.

Otros integrantes de mi familia llegaron tarde para poder ver el nacimiento de esta tesis: unos porque ya no están, otros porque no quisieron estar. A todos ellos, va dedicada también esta tesis.

Resumen

Los portales de aplicaciones y servicios web suelen ser una de las puertas de entrada para el lanzamiento de ataques y otros tipos de actividades malintencionadas contra empresas y diversos tipos de entidades. Desde bancos a webs de comercio electrónico, pasando por las infraestructuras de sistemas sanitarios, sistema judicial, etc., los posibles perjuicios económicos, reputacionales, de fuga de información y de otra índole ocasionados no solo a las organizaciones, sino también a los usuarios legítimos de las aplicaciones y servicios web por un ataque, son incalculables. En un afán de proporcionar una capa de protección adicional contra este tipo de ataques, se ha investigado abundantemente sobre técnicas de protección web: desde un enfoque más clásico basado en reglas de protección que deben actualizarse constantemente hasta las técnicas basadas en la detección de anomalías, el número de estudios Con esta tesis, se pretende contribuir a afianzar el conocimiento sobre las técnicas de detección de anomalías mediante tres artículos en los que se aporta conocimiento a la comunidad científica mediante la primera revisión sistemática de literatura de las técnicas de detección de anomalías aplicadas a la protección de aplicaciones web. Posteriormente se plantea una nueva metodología para la comparación objetiva de herramientas de protección web, demostrando su aplicabilidad mediante la comparación de diversas herramientas WAF y RASP. Por último, se facilita a la comunidad científica un nuevo dataset multietiqueta con el que se entrenan nuevos diseños de modelos de clasificación capaces de identificar los ataques web mediante patrones de ataque CAPEC.

Abstract

Application portals and web services are often one of the gateways for launching attacks and other types of malicious activities against companies and various types of entities. From banks to e-commerce sites, healthcare systems infrastructures, judicial systems, etc., the potential economic, reputational, information leakage and other types of damage caused not only to organizations, but also to legitimate users of web applications and services by an attack, are incalculable. In an effort to provide an additional layer of protection against this type of attacks, there has been abundant research on web protection techniques: from a more classical approach based on protection rules that need to be constantly updated to techniques based on anomaly detection, the number of studies on anomaly detection techniques is increasing. With this thesis, we aim to contribute to strengthen the knowledge on anomaly detection techniques through three articles that provide knowledge to the scientific community through the first systematic literature review of anomaly detection techniques applied to web application protection. Subsequently, a new methodology for the objective comparison of web protection tools is proposed, demonstrating its applicability by comparing different WAF and RASP tools. Finally, a new multi-label dataset is provided to the scientific community to train new classification model designs capable of identifying web attacks by means of CAPEC attack patterns.

Índice general

Índice de figuras

Índice de tablas

Esta tesis es un compendio de artículos científicos. A continuación, se muestran las referencias completas de las publicaciones que forman la tesis en el orden en que se ha desarrollado la investigación.

Título	***Prevention and Fighting against Web Attacks through Anomaly Detection Technology. A Systematic Review***
Revista	Sustainability. Special Issue "Security on Web-Based Applications: Technologies, Methodologies, Analysis Methods and Recent Advances in Cybersecurity"
ISSN	2071-1050
Editorial	mdpi
Volumen	12
Página	4945
Año	2020
DOI	10.3390/su12124945
Estado	Publicado
Factor de Impacto (2020)	3.251 (JIF), 0.56 (JCI)
Ranking	Q2 posición 124 de 274 (JIF), Q3 posición 163 de 306 (JCI)
Página web	https://www.mdpi.com/journal/sustainability/special_issues/security_web_based_applications

Título	***Systematic Approach For Web Protection Runtime Tools' Effectiveness Analysis***
Revista	CMES - Computer Modeling in Engineering and Sciences
ISSN	1526-1506
Editorial	Tech Science Press
Volumen	
Página	
Año	2022
DOI	10.32604/cmes.2022.020976
Estado	Publicado
Factor de Impacto (2021)	2.027 (JIF), 0.66 (JCI)
Ranking	Q3 posición 52 de 92 (JIF), Q2 posición 58 de 175 (JCI)
Página web	https://www.techscience.com/CMES

Título	***A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques***
Revista	Computers & Security
ISSN	1872-6208
Editorial	Elsevier Advanced Technology
Volumen	120
Página	
Año	2022
DOI	10.1016/j.cose.2022.102788
Estado	Publicado
Factor de Impacto (2021)	5.105 (JIF), 1.39 (JCI)
Ranking	Q2 posición 42 de 164 (JIF), Q1 posición 32 de 246 (JCI)
Página web	https://www.journals.elsevier.com/computers-and-security

1. Introducción

1.1. Motivación y objetivos

Desde la aparición de los servicios y aplicaciones web ha cambiado el modo de vida de nuestras sociedades, permitiendo que operaciones tales como la realización de transferencias bancarias, realizar reservas de vuelos y hoteles, comprar determinados productos, etc., que anteriormente requerían una inversión significativa de tiempo y/o desplazamientos, sean realizadas de forma relativamente fácil e inmediata desde la comodidad de nuestros hogares o lugares de trabajo. Más aún, gracias a la implantación a gran escala de los servicios y aplicaciones web, se ha producido una expansión exponencial del conocimiento disponible mundialmente. Desafortunadamente, esta gran expansión de los servicios y aplicaciones web implica un evidente aumento del riesgo potencial de acceso y mal uso de los servicios e información disponibles, por lo cual la detección y prevención de ataques a aplicaciones web ha devenido un área de especial interés.

En el año 2020 el número de incidentes de ciberseguridad reportados creció de forma exponencial, debido principalmente a la digitalización forzosa de negocios, empresas e instituciones de todo tipo, que tuvo que llevarse término de forma apresurada a causa de la crisis sanitaria global originada por la pandemia del virus SARS-CoV-2. Ese cambio de paradigma global, llevado a cabo muchas veces sin aplicar las mínimas garantías de seguridad y protección, permitió la proliferación de nuevas oportunidades de ataque y actores hostiles. Se publicaron en ese año diversos avisos de Vulnerabilidades y Exposiciones Comunes *(Common*

Vulnerabilities and Exposures – CVE) que afectan a servidores web como Oracle: CVE – 2020 – 14882 (MITRE Corporation, 2020) y Apache: CVE – 2020 – 1938 (MITRE Corporation, 2019). Del mismo modo, los principales gestores de contenido *(Content Management System – CMS)* se han visto afectados por diversas vulnerabilidades: por ejemplo, una simple búsqueda en la web del CVE retorna a día de hoy (20 de marzo de 2022) 4.006 vulnerabilidades para el gestor de contenido WordPress[1], 1.271 vulnerabilidades para el gestor de contenido Joomla![2] y 1.101 vulnerabilidades para el gestor de contenido Drupal[3]. Debe destacarse que el gestor de contenido WordPress es el más usado según las estadísticas recopiladas por W3Techs (W3Techs, 2022), estando presente en el 43% de los sitios web analizados y representando una cuota de mercado del 64,7% entre todos los gestores de contenidos.

La detección y prevención de ataques web se ha convertido en un área de especial interés y estudio como demuestran los numerosos estudios e investigaciones realizadas hasta la fecha, destacando en particular aquellas investigaciones que contemplan el uso de técnicas de detección de anomalías y aprendizaje automático, por ejemplo: "A Distributed Deep Learning System for Web Attack Detection on Edge Devices" (Tian et al., 2020), "A Novel Web Attack Detection System for Internet of Things via Ensemble Classification" (Luo et al., 2021), "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection" (Kim et al., 2020), "A novel architecture for web-based attack detection

[1] https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=wordpress
[2] https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=joomla
[3] https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=drupal

using convolutional neural network" (Tekerek, 2021). Sin embargo, a pesar del enorme interés suscitado en la comunidad científica por este tipo de técnicas de protección, no existen revisiones sistemáticas de literatura que permitan una visión adecuada del estado del arte; en la investigación llevada a cabo para la preparación de esta tesis doctoral, únicamente se recuperaron diversas *"reviews"* y *"surveys"*, por ejemplo: "A comprehensive survey on network anomaly detection" (Fernandes et al., 2019), "Machine Learning Techniques for Network Anomaly Detection: A Survey" (Eltanbouly et al., 2020), "A holistic review of Network Anomaly Detection Systems: A comprehensive survey" (Moustafa et al., 2019). De aquí surge el primer motivo que lleva a mejorar el estado del arte disponible sobre la materia: **Realizar una revisión sistemática de literatura sobre el uso de técnicas de detección de anomalías y aprendizaje automático aplicadas a la detección y prevención de ataques web.**

Por otra parte, abundan en el mercado diferentes herramientas Open Source y Comerciales destinadas a la protección de las aplicaciones y servicios web, destacando los *"Cortafuegos de la Capa de Aplicación" (Web Application Firewall – WAF)* y la *"Autoprotección de Aplicaciones en Tiempo de Ejecución" (Runtime Application Self-protection – RASP)*. Las herramientas WAF y RASP son herramientas complejas, difíciles de configurar y comparar debido al sutil equilibrio entre Falsos Positivos (FP) y Falsos Negativos (FN) que pueden tolerarse, en función de las características de las organizaciones y los niveles de criticidad de los diferentes escenarios en los que están implantados las aplicaciones y/o servicios web. Revisando la literatura científica disponible, no se encontró

ninguna metodología ni marco de trabajo que permita la comparación de estas u otras herramientas de protección de aplicaciones o servicios web, por lo que a partir de estas limitaciones, se deriva el segundo objetivo de este trabajo: **Desarrollar una metodología experimental específica que permita la comparación de diferentes herramientas de protección web, demostrando su aplicabilidad mediante la comparación de diferentes herramientas WAF y RASP de forma objetiva en función de las puntuaciones obtenidas por la métrica F-Score.**

Los resultados obtenidos en la revisión sistemática de literatura llevada a cabo en el primer objetivo de este trabajo pusieron de manifiesto diversas carencias en la investigación dirigida a la prevención de ataques web mediante técnicas de detección de anomalías y aprendizaje automático, entre otras:

a) la falta de datasets públicos y actualizados destinados al entrenamiento de modelos de aprendizaje automático para la detección y prevención de ataques a aplicaciones y servicios web, permitiendo, además, la replicación y validación de los resultados experimentales obtenidos en las diferentes investigaciones,

b) indeterminación del tipo o tipos de ataque en las etiquetas de clasificación de los datasets disponibles en la actualidad, reflejando éstas únicamente si cada uno de los registros de los datasets corresponde a una petición normal o se trata de un posible ataque.

A partir de las carencias detectadas, se establece el tercer objetivo de este trabajo, a saber: **Creación de un dataset multietiqueta** (de Carvalho & Freitas, 2009) **destinado al entrenamiento de modelos de aprendizaje**

automático que, además de informar si la petición web analizada es un presunto ataque o no, clasifique el tipo o tipos de ataque de que se trata, de acuerdo a la clasificación CAPEC (MITRE Corporation, 2022a)**.**

Como paso posterior y lógico a la creación del dataset multietiqueta adecuado para el entrenamiento de modelos de aprendizaje automático, se deriva el cuarto objetivo de este trabajo: **Diseñar un modelo de clasificación multietiqueta que permita la identificación del tipo o tipos de ataque presentes en una determinada petición realizada a un servidor web.**

Esta tesis se fundamenta bajo el formato de compendio de publicaciones de artículos, cuyos resultados garantizan el cumplimiento de los objetivos marcados en este trabajo de investigación. El resto de la memoria de tesis se organiza de la siguiente forma:

Tras esta introducción, en la Sección 2, se resume el contexto y el estado del arte obtenido a partir de los resultados de la revisión sistemática de literatura publicada como uno de los artículos que componen esta tesis, el cual se transcribe junto al resto de artículos en la Sección 4.

En la Sección 3, se discuten las aportaciones conseguidas con el desarrollo del trabajo de investigación plasmado en la presente tesis.

El artículo presentado en la Sección 4.1, con título *"Prevention and Fighting against Web Attacks through Anomaly Detection Technology. A Systematic Review"* presenta los hallazgos y limitaciones encontradas respecto al estado del

arte de las técnicas de aprendizaje automático aplicadas a la prevención y detección de ataques en aplicaciones y servicios web.

El artículo presentado en la Sección 4.2, con título *"Systematic Approach For Web Protection Runtime Tools Effectiveness Analysis"* presenta una nueva metodología experimental que permite la comparación de cualquier herramienta de protección web, mediante el uso de bancos de prueba y la elección de diferentes indicadores (TP,TN,FP,FN) y métricas (F-Score, Precision, Recall, Accuracy).

En la Sección 4.3, se presenta el artículo titulado *"A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques"* en el que se examinan las características del nuevo dataset multietiqueta *"SR-BH 2020"* (Sureda Riera et al., 2022), creado como intento de solucionar parte de las carencias detectadas en la revisión sistemática de literatura publicada. En este artículo se presenta, además, una nueva técnica de extracción de características, mediante la suma de la media de la asignación de los correspondientes valores ASCII de los distintos campos que componen una petición web. Por otra parte, se diseñan y evalúan diversas combinaciones de modelos de clasificación multietiqueta y algoritmos de clasificación (LightGBM y CatBoost).

Finalmente, en la Sección 5 se desarrollan las conclusiones del trabajo realizado en base a los distintos artículos publicados, planteándose las líneas futuras de investigación.

2. Contexto y estado del arte

2.1. Patrones de ataque

2.1.1. Historia

Los patrones de ataque forman parte del arsenal de seguridad en el que, además de estos, se incluyen herramientas como los casos de mal uso y abuso, modelos de amenazas, árboles de ataque, etc. Los patrones de ataque ayudan a categorizar los ataques a una aplicación de manera efectiva, aportando detalles sobre cómo se llevan a cabo los ataques con el fin de que los desarrolladores puedan ayudar a prevenirlos (Sethi & Barnum, 2013). El concepto de patrones de ataque deriva de la noción de los patrones de diseño, los cuales capturan el contexto y los detalles de alto nivel de una solución general repetible para tratar un problema común en el diseño de software (Gamma et al., 1995).

El término "patrones de ataque" se introdujo en el documento *Attack Modeling for Information Security and Survivability* (Moore et al., 2001) y se amplió su uso con la publicación del libro *Exploiting Software: How to Break Code* (Hoglund & McGraw, 2004). A partir de ese momento, se llevaron a cabo diversos esfuerzos por establecer una definición y esquemas comunes para los patrones de ataque, así como la creación de un organismo independiente que actúe como recopilador y difusor de catálogos de patrones de ataque comunes; en este estadio surge la iniciativa, patrocinada por el Departamento de Seguridad Nacional de Estados Unidos (*U.S. Department of Homeland Security – DHS*), de la creación

de la Enumeración y Clasificación de Patrones de Ataque Comunes (*Common Attack Pattern Enumeration and Classification – CAPEC*), donde se detallan conjuntos de instancias de patrones de ataque.

2.1.2. Concepto

Un patrón de ataque es un mecanismo de abstracción para describir cómo se ejecuta un tipo de ataque, proporcionando una descripción del contexto en el que es aplicable y ofreciendo recomendaciones para la mitigación del ataque. El objetivo de los patrones de ataque es la representación de manera formalizada de la perspectiva del atacante, de forma que se ofrezca a los desarrolladores de software una visión objetiva de la forma cómo su software es probable que sea atacado, contribuyendo de esta forma a la construcción de software más seguro (Barnum, 2007).

2.1.3. Catálogo de patrones de ataque

La iniciativa CAPEC ofrece un listado completo y actualizado de patrones de ataque; las entradas de CAPEC están relacionadas con la lista de Enumeración de Debilidades Comunes (*Common Weakness Enumeration – CWE*) (MITRE Corporation, 2022c) y la lista de Vulnerabilidades y Exposiciones Comunes (*Common Vulnerabilities and Exposures – CVE*) (MITRE Corporation, 2022b), ya que un patrón de ataque no deja de ser un método de aprovechar una vulnerabilidad para perpetrar un ataque; así, la mayoría de entradas en la lista

CAPEC contienen un flujo de ejecución detallado para que un adversario explore en busca de objetivos potenciales, experimente con sus activos y mecanismos defensivos y lleve a cabo la explotación de la vulnerabilidad.

Como ejemplo de un patrón de ataque catalogado según CAPEC, se revisará a continuación la entrada "CAPEC – 66: SQL Injection". Como puede observarse en la Figura 1 la entrada se divide en diferentes secciones, cada una de las cuales proporcionará información útil para la comprensión del ataque en cuestión, detallando entre otra información la descripción básica y extendida del ataque,

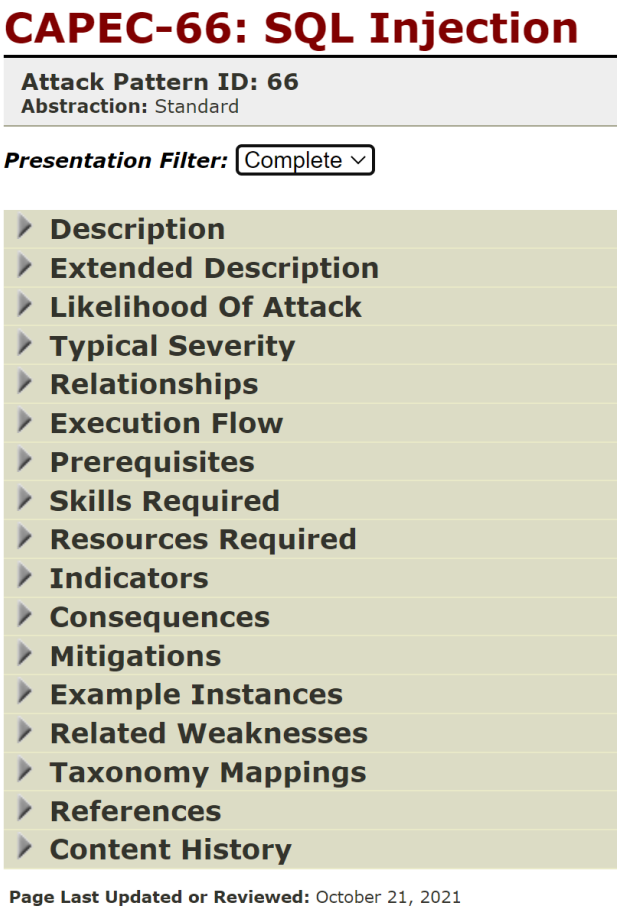
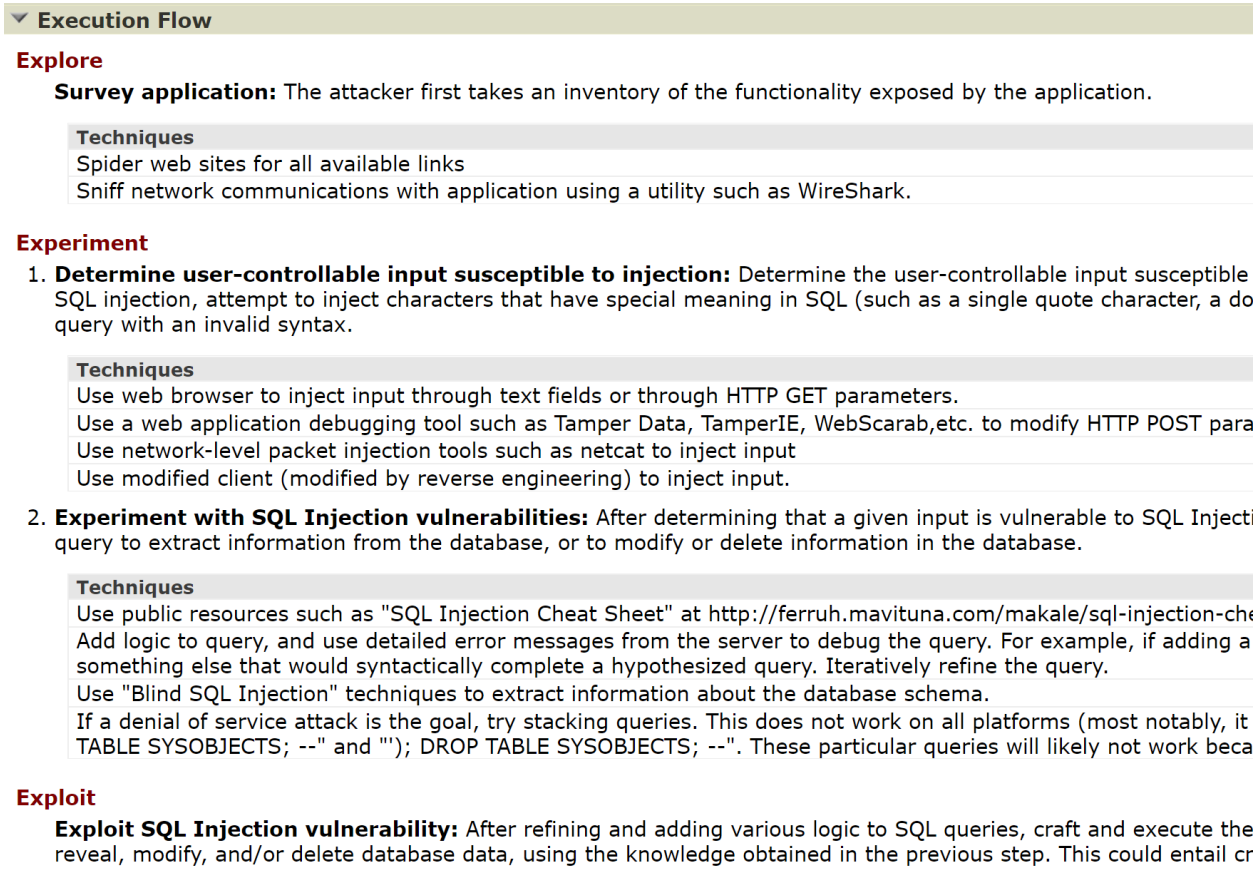


Figura 1: Secciones de una entrada CAPEC

severidad, flujo de ejecución, habilidades necesarias para realizar la explotación de

la vulnerabilidad en cuestión, mitigaciones posibles, etc. Véanse para un detalle del flujo de ejecución del ataque y las posibles mitigaciones que pueden aplicarse para la prevención de la explotación de la vulnerabilidad, la Figura 2 y la Figura 3 respectivamente.



Execution Flow

Explore

Survey application: The attacker first takes an inventory of the functionality exposed by the application.

Techniques
Spider web sites for all available links
Sniff network communications with application using a utility such as WireShark.

Experiment

1. **Determine user-controllable input susceptible to injection:** Determine the user-controllable input susceptible SQL injection, attempt to inject characters that have special meaning in SQL (such as a single quote character, a do query with an invalid syntax.

Techniques
Use web browser to inject input through text fields or through HTTP GET parameters.
Use a web application debugging tool such as Tamper Data, TamperIE, WebScarab,etc. to modify HTTP POST para
Use network-level packet injection tools such as netcat to inject input
Use modified client (modified by reverse engineering) to inject input.

2. **Experiment with SQL Injection vulnerabilities:** After determining that a given input is vulnerable to SQL Injecti query to extract information from the database, or to modify or delete information in the database.

Techniques
Use public resources such as "SQL Injection Cheat Sheet" at http://ferruh.mavituna.com/makale/sql-injection-che
Add logic to query, and use detailed error messages from the server to debug the query. For example, if adding a something else that would syntactically complete a hypothesized query. Iteratively refine the query.
Use "Blind SQL Injection" techniques to extract information about the database schema.
If a denial of service attack is the goal, try stacking queries. This does not work on all platforms (most notably, it TABLE SYSOBJECTS; --" and "'); DROP TABLE SYSOBJECTS; --". These particular queries will likely not work beca

Exploit

Exploit SQL Injection vulnerability: After refining and adding various logic to SQL queries, craft and execute the reveal, modify, and/or delete database data, using the knowledge obtained in the previous step. This could entail cr

Figura 2: Flujo de ejecución de un ataque

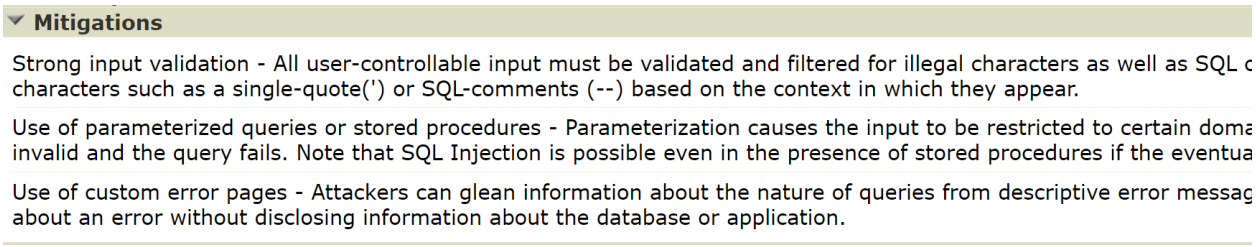


Figura 3: Posibles mitigaciones de una vulnerabilidad

2.2. Herramientas y técnicas de protección web

La historia del desarrollo del software demuestra que las aplicaciones se diseñan generalmente con la intención de proporcionar características funcionales y atractivas para el consumidor final y no con la seguridad en mente (seguridad por diseño). La corrección de errores de seguridad detectados en las aplicaciones una vez desarrolladas no suele ser viable económicamente, debido a los costes y recursos necesarios para integrar la seguridad en una aplicación con decenas o cientos de miles de líneas de código sin que se produzcan problemas de compatibilidad o errores sobrevenidos. Por ello, una práctica común en aplicaciones web, es interponer una capa adicional de protección entre la aplicación y los usuarios finales.

Según Liao et al. (Liao et al., 2013), los Sistemas de Detección de Intrusiones *(Intrusion Detection Systems – IDS)* pueden dividirse en tres categorías en función de la técnica usada para su detección:

- Basados en una base de datos de firmas *(Signature-based detection – SD)*: una firma corresponde a un patrón de ataque conocido; para detectar posibles intrusiones, los datos de la petición web se comparan con la base de datos de firmas.
- Basados en la detección de anomalías *(Anomaly-based detection – AD)*: una anomalía se detecta cuando hay una desviación del comportamiento habitual, que se representa mediante perfiles (estáticos o dinámicos). La detección de anomalías se realiza comparando los perfiles normales con los eventos observados para detectar ataques o intrusiones. Este tipo de IDS es capaz de detectar malware y ataques desconocidos mediante la aplicación de algoritmos de *Machine Learning* (Ahmad et al., 2018; Maseer et al., 2021).
- Basados en el análisis del protocolo de estado *(Stateful Protocol Analysis – SPA)*: en este caso, el IDS puede conocer y rastrear los estados del protocolo (por ejemplo, emparejando las peticiones con las respuestas). Aunque el proceso de SPA se parece al de AD, son esencialmente diferentes. AD adopta perfiles específicos de red o de host precargados, mientras que SPA depende de perfiles genéricos desarrollados por el proveedor para protocolos específicos. Por lo general, los modelos de protocolo de red en SPA se basan originalmente en los estándares de protocolo de las organizaciones internacionales de normalización, por ejemplo, el *Internet Engineering Task Force (IETF*).

Por otra parte, Prokhorenko et al. (Prokhorenko et al., 2016) clasifican las técnicas de protección web según el siguiente esquema:

- Enfoques de caja blanca *(White-box approaches)*: estos enfoques requieren acceso al código fuente de la aplicación. El análisis del código fuente puede llevarse a cabo antes de la ejecución del código (análisis estático), o bien en tiempo de ejecución de la aplicación (análisis dinámico). El análisis estático implica que los errores detectados en el código de la aplicación deben solucionarse antes de que ésta pueda ser usada en un entorno de producción, mientras que el análisis dinámico sigue el flujo del código de la aplicación en tiempo real, permite prevenir la ejecución de ciertas partes del código y proporciona acceso a los datos proporcionados por el usuario. Asimismo, permite la construcción de un modelo de comportamiento esperado de la aplicación que puede ser usado en la fase de decisión de la detección de intrusiones para comprobar si el comportamiento de la aplicación es el esperado (Tonella & Ricca, 2004). Las herramientas de protección web *Runtime Application Self Protection - RASP* son un claro ejemplo del uso del enfoque de caja blanca en tiempo de ejecución.
- Enfoques de caja negra *(Black-box approaches)*: estos enfoques no requieren acceso al código fuente de la aplicación, ya que considerar una aplicación web como una caja negra significa que las técnicas de protección únicamente tienen acceso a las entradas y salidas de la aplicación. La ventaja de aplicar los enfoques de caja negra es que estas técnicas pueden desplegarse para cualquier aplicación web, aunque no se disponga del

código fuente. Sin embargo, este enfoque puede llevar a una menor tasa de detección de ataques ya que no se dispone de un conocimiento profundo de cómo debe comportarse la aplicación. Las herramientas de protección web *Web Application Firewall – WAF* son representativas de este enfoque, ya que funcionan a nivel de la capa 7 del modelo OSI (capa de aplicación), analizando el tráfico bidireccional generado entre la aplicación y el cliente web, mediante un conjunto de reglas que deben actualizarse con frecuencia, con el objeto de detectar y bloquear el tráfico malicioso.

- Protección contra ataques de inyecciones SQL: estas protecciones tienen como objetivo eludir o detectar al menos ciertos tipos de ataques de inyección SQL. Las vulnerabilidades producidas por inyección SQL se atribuyen tradicionalmente a la falta de sanitización de entradas. Las técnicas de sanitización de entradas incluyen consultas SQL parametrizables *(parameterizable queries)* en C#, el método *PreparedStatement* en java, o el uso de funciones de escape como *mysql_real_escape_string()* en mysql. En el caso de aplicaciones heredadas *(legacy applications)*, existen problemas adicionales como la identificación de las funciones de llamada críticas, ya que son dependientes del lenguaje de programación usado y pueden requerir análisis estáticos complejos (Zheng & Zhang, 2013) además de la necesidad formal de verificar el correcto funcionamiento de las funciones de sanitización (Yu et al., 2010).
- Protección contra ataques XSS: su objetivo es eludir o detectar al menos, ciertos tipos de ataques XSS. La mayoría de soluciones de defensa estándar

en la industria contra este tipo de ataques se basan en el saneamiento de las entradas de usuario (Weinberger et al., 2011). Los ataques XSS almacenados y reflejados se producen debido a que los navegadores asumen erróneamente que las páginas web generadas por el servidor son de confianza; en este contexto, se ha sugerido que este tipo de ataques deben ser vistos como un problema de confianza y no como un problema de saneamiento de entradas de usuario (Van Gundy & Chen, 2012). Otros enfoques de defensa contra este tipo de ataques se basan en la monitorización de las respuestas HTTP retornadas por el servidor web, mediante un proxy inverso interpuesto (Nadji et al., 2009).

- Protección contra ataques SVG incrustados: su objetivo es eludir o detectar ataques incrustados en imágenes del tipo Scalable Vector Graphics (SVG) o en contenidos Flash. La capacidad de incluir código JavaScript incrustado en imágenes SVG puede dar lugar a nuevos tipos de ataques web; se ha propuesto la validación de la imagen SVG contra un esquema XML como forma de protección, si bien se ha demostrado que esta validación no es suficiente, ya que los esquemas XML se centran en la estructura del documento más que en el contenido (Heiderich et al., 2011). Otro enfoque de protección contra este tipo de ataques puede abordarse a través de la "purificación" de las imágenes SVG *(SVG purification)*; mediante esta técnica, se elimina el contenido malicioso de la imagen permitiendo únicamente el contenido incorporado en una lista blanca (Heiderich et al., 2011).

- Monitorización de las entradas del usuario: este enfoque asume que la aplicación es una caja negra, por lo que focaliza la observación de las entradas del usuario para detectar determinados tipos de ataques, monitorizando principalmente los parámetros GET, POST y las cabeceras HTTP (Buehrer et al., 2005). A diferencia de los enfoques basados en el proxy inverso, la monitorización de la entrada del usuario permite la detección de posibles ataques en una fase más temprana, lo que hace posible tanto la detección como la prevención de estos. El problema principal de los enfoques de monitorización de las entradas de los usuarios radica en la selección de las características correctas que deben observarse, ya que no siempre existe una diferencia obvia entre las entradas de usuario normales y maliciosas. Normalmente, se establece una fase de entrenamiento en la cual el sistema aprende qué tipo de información suministran los usuarios durante el uso habitual del sistema; la información capturada en la fase de entrenamiento, se utiliza en la fase de monitorización en tiempo real para determinar si la entrada de usuario monitorizada difiere de las entradas esperadas, siendo la determinación del umbral de diferencia entre la entrada real y la esperada uno de los problemas a los que se enfrenta este enfoque.
- Contaminación de los parámetros HTTP *(HTTP parameter pollution)*: este enfoque pretende eludir o detectar los ataques por contaminación de parámetros HTTP, el cual es posible porque la especificación del protocolo HTTP sólo define la estructura de los parámetros de entrada (suministrados por los usuarios) sin imponer suficientes restricciones a los nombres y

valores reales de los parámetros. La petición HTTP recibida por el servidor web, debe ser preparada para poder ser procesada por dicho servidor; esto incluye la generación de estructuras de diccionarios que contengan las claves y los valores de los parámetros pasados. Si una petición web incluye dos o más parámetros con el mismo nombre, se provocan ambigüedades en el lado del servidor, ya que el diccionario no puede contener la misma clave más de una vez (Balduzzi et al., 2011).

- Manejo inadecuado de las entradas del usuario: este enfoque asume que forzar un tratamiento uniforme de las entradas de los usuarios es suficiente para detectar o eludir algunos tipos de tipos de ataques. La verificación de la entrada del usuario se suele implementar en el lado del cliente para mejorar la experiencia del usuario, ya que la comprobación de la entrada del usuario en el lado del cliente es significativamente más rápida que la comprobación en el lado del servidor. Esta diferencia se explica por el notable tiempo de ida y vuelta de los paquetes al servidor. No obstante, para fines de seguridad, únicamente puede usarse la verificación en el lado del servidor (Bisht et al., 2011; Skrupsky et al., 2013). Se han propuesto enfoques que comparan los procedimientos de saneamiento del lado cliente y servidor, informando de las discrepancias entre ellos. Si una comprobación de sanidad del lado del servidor es menos restrictiva que la correspondiente del lado del cliente, podría ser posible un ataque, ya que los usuarios maliciosos de las aplicaciones web pueden desactivar la comprobación de sanidad del lado del cliente, que es más restrictiva. El "rastreo de contaminación" *(taint tracking)* se utiliza habitualmente para

analizar los aspectos de la aplicación relacionados con el manejo de entradas por parte del usuario (Livshits et al., 2009; Tripp et al., 2009). Como el conjunto de funciones sensibles (funciones de acceso a la base de datos, llamadas al sistema, etc.) se conoce de antemano y las entradas del usuario sólo pueden pasarse a través de un conjunto predefinido de medios (parámetros GET o POST, cookies o cabeceras), la principal tarea de los enfoques basados en *taint tracking* es rastrear los flujos de datos para detectar situaciones en las que las entradas del usuario se pasan a funciones sensibles.

- Aislamiento de las entradas de los usuarios: este enfoque asume que la separación de las entradas de los usuarios del resto de datos procesados por la aplicación es adecuada para eludir o detectar diversos tipos de ataques. Aunque centrarse en la entrada del usuario es beneficioso en términos de detección temprana, la falta de separación del contexto entre los diferentes tipos de datos (como la entrada del usuario y el contenido generado por la aplicación) y el código complica el análisis de la entrada del usuario. El resultado generado por una aplicación web es simplemente una cadena de bytes que no contiene información semántica o estructural. Dado que la confianza en los datos devueltos se basa en primer lugar en la fuente de los datos, la adición de metainformación para separar la entrada del usuario de la salida de la aplicación web puede permitir distinguir entre ambas mientras se analiza la cadena generada (como la página web o la consulta SQL). El uso de un conjunto de símbolos de protección para rodear las partes de la cadena proporcionadas por el usuario permite la separación de

datos (Bravenboer et al., 2010); el problema de este enfoque es que los símbolos de protección forman parte del mismo dominio de caracteres de la cadena. Si los símbolos utilizados para la separación son conocidos por el atacante, éste puede suministrar el mismo conjunto de símbolos para obtener el control del cambio de contexto (Su & Wassermann, 2006).

- Protecciones basadas en control de acceso: se asume que un control formal de acceso es adecuado para eludir o detectar diferentes tipos de ataques. Este tipo de protección es muy específico de la aplicación, ya que diferentes aplicaciones tienen diferentes conjuntos de roles. La definición inicial de los roles junto con los correspondientes recursos externos a los que acceden debe ser proporcionada por el desarrollador de la aplicación. Una vez establecidas las definiciones iniciales, estos enfoques pueden utilizar diversas técnicas para derivar automáticamente los intentos de acceso, por ejemplo, escaneando las páginas web en busca de diversas características, como hipervínculos o cualquier intento de acceso a objetos específicos (Sun et al., 2011).

2.3. Detección de anomalías

La detección de anomalías se define como *"el proceso de encontrar valores atípicos en un conjunto de datos determinado"* (Kotu & Deshpande, 2019); en este contexto, valores atípicos son datos, incluidos en un conjunto de datos, cuyo comportamiento no se ajusta al esperado.

Es práctica común clasificar los algoritmos de detección de anomalías según su finalidad; las principales categorías son las siguientes (Hodge & Austin, 2004):

- Algoritmos supervisados: modelan las relaciones entre los datos de entrada y la predicción, en función de las relaciones aprendidas a partir de los datos de entrada etiquetados. Ejemplo de este tipo de algoritmos son: Nearest Neighbor, Naive Bayes, Decision Trees, Linear Regression, Support Vector Machines (SVM) y Neural Networks.
- Algoritmos no supervisados: realizan la detección de patrones en los datos de entrada, ya que no existen datos de entrenamiento etiquetados. Como ejemplos de algoritmos de esta categoría, pueden citarse k-means y las reglas de asociación.
- Algoritmos semi-supervisados: utilizan pocos datos etiquetados y un gran número de datos sin etiquetar como parte del conjunto de entrenamiento, intentando explorar la información estructural contenida en los datos sin etiquetar como forma de generar modelos predictivos que funcionen mejor que aquellos que usan solamente datos etiquetados. Ejemplo de este tipo de algoritmos son: Generative models, Low-density separation y Graph-based methods.
- Algoritmos de refuerzo: el objetivo es el desarrollo de un sistema (llamado agente) que pretende mejorar su eficiencia realizando una determinada tarea en función de la interacción con su entorno, recibiendo recompensas que le permitan adaptar su comportamiento. A medida que el agente recibe recompensas, debe desarrollar la estrategia

adecuada (llamada política) que le lleve a obtener recompensas positivas en todas las situaciones posibles. Ejemplos comunes de algoritmos de refuerzo son: Q-Learning, Temporal Difference (TD) y Deep Adversarial Networks.

Los sistemas tradicionales de detección y prevención de intrusiones basados en firmas proporcionan muy buenos resultados de detección frente a ataques conocidos, pero tienen serias dificultades a la hora de detectar nuevas intrusiones, incluso si se construyen como variantes mínimas de ataques ya conocidos. Por el contrario, los algoritmos de detección de anomalías son capaces de detectar eventos de intrusión no vistos anteriormente, si bien su tasa de Falsos Positivos es sensiblemente superior que en los sistemas basados en firmas (García-Teodoro et al., 2009).

Un proceso de detección de anomalías implica el uso de diferentes estrategias y diferentes técnicas para conseguir el objetivo final: algoritmos de clustering, algoritmos de clasificación, técnicas de reducción de la dimensionalidad, uso de técnicas auxiliares, etc. A continuación, se detallan los principales algoritmos y técnicas detectadas en los diferentes estudios analizados.

2.3.1. Algoritmos de agrupamiento (clustering)

El clustering es la tarea de agrupar un conjunto de objetos de manera que los objetos de un mismo grupo sean más similares entre sí que los de otros grupos (clusters); dado que el objetivo de la agrupación es descubrir un nuevo conjunto de

categorías, los nuevos grupos son de interés en sí mismos, y su evaluación es intrínseca. En función de la comparación de los nuevos datos recibidos con el modelo generado por el algoritmo de clustering, se determina si se trata de un punto anómalo (Rokach & Maimon, 2005). Principales algoritmos de clustering:

- ***K-means***: es un algoritmo de clustering no supervisado que agrupa los objetos en k grupos en función de sus características. La agrupación se realiza minimizando la suma de las distancias entre cada objeto y el centroide de su grupo o cluster. El algoritmo k-means resuelve un problema de optimización. La función a optimizar es la suma de las distancias cuadráticas de cada objeto al centroide de su cluster.
- **Los *Modelos de Mezcla Gaussiana (Gaussian Mixture Models)***: se trata de un modelo probabilístico para representar subpoblaciones con distribución normal dentro de una población global. Los modelos de mezcla, en general, no requieren saber a qué subpoblación pertenece un punto de datos, lo que permite al modelo aprender las subpoblaciones automáticamente. Dado que no se conoce la asignación de la subpoblación, esto constituye una forma de aprendizaje no supervisado.
- **La *Distancia de Mahalanobis (Mahalanobis Distance)***: es una métrica de distancia multivariante que mide la distancia entre un punto (vector) y una distribución. El uso más común de la distancia de Mahalanobis es encontrar valores atípicos multivariantes, que indican combinaciones inusuales de dos o más variables.

- **La *Propagación por Afinidad (Affinity Propagation)***: no requiere que se determine el número de clusters antes de ejecutar el algoritmo. Los puntos de datos pueden verse como una red en la que todos los puntos de datos envían mensajes a todos los demás puntos. El objeto de los mensajes es la determinación de los puntos que son un ejemplar. Los ejemplares son puntos que explican "mejor" los otros puntos de datos y son los más significativos de su agregación. Todos los puntos de datos quieren determinar colectivamente los puntos de datos que son un ejemplar para ellos (Frey & Dueck, 2007). Estos mensajes se guardan en dos matrices:
    - a) *Matriz de Responsabilidad (Responsability Matrix) R*. En esta matriz, *r(i,k)* denota lo bien que se ajusta el punto *k* para ser un ejemplar del punto *i*.
    - b) *Matriz de Disponibilidad (Availability Matrix) A*. En esta matriz, *a(i,k)* refleja la precisión con la que el punto *i* seleccionaría el punto *k* como ejemplar.
- ***Density-based spatial clustering of applications with noise (DBSCAN)*** (Ester et al., 1996): es un algoritmo de clustering basado en la densidad; encuentra un número de grupos (clusters) a partir de una distribución de densidad dada de los nodos correspondientes. La agrupación se produce en función de dos parámetros: *Vecindad (Neighbourhood)*, distancia de corte de un punto respecto al punto central para que se considere parte de un cluster, comúnmente referido

como ε. *Puntos Mínimos (Minimun Points)*, número mínimo de puntos requeridos para formar un cluster, comúnmente referido como minPts.

- ***Factor de Valores Atípicos Locales basado en el Vecino Más Cercano (Nearest Neighbor based Local Outlier Factor)***: este algoritmo se basa en el concepto de densidad local, donde la localidad viene dada por los k vecinos más cercanos. La densidad se estima por la distancia entre los vecinos más cercanos. Si se compara la densidad de un objeto con las densidades de sus vecinos, se identificarán regiones con valores de densidad similares y puntos con valores de densidad muy inferiores a los obtenidos por sus vecinos. Estos puntos se consideran valores atípicos (Breunig et al., 2000).
- ***Maximización de Expectativas (Expectation–Maximization)***: se trata de una forma iterativa de aproximar la función para encontrar estimaciones de máxima verosimilitud para los parámetros del modelo cuando los datos son incompletos, faltan puntos de datos o hay variables latentes no observadas (ocultas) (Dempster et al., 1977; Gupta & Chen, 2011).

2.3.2. Algoritmos de clasificación

La idea que subyace a los algoritmos de clasificación es muy sencilla: se trata de predecir la clase objetivo analizando el conjunto de datos de entrenamiento; la clasificación de los nuevos datos como anómalos o no, depende de la clase en la que se clasifican.

Todos los algoritmos de clasificación pueden generalizarse como algoritmos que reciben un conjunto de entrenamiento y aprenden una función de clasificación de la forma $f: R^n \rightarrow \{+1, -1\}$. Esta función se aplica a las nuevas entradas y su valor representa la clase a la que se clasifica la entrada (Duda & Hart, 1973). Principales algoritmos de clasificación:

- ***Máquina de Vectores de Apoyo de Una Clase (One Class Support Vector Machine – OCSVM)***: este algoritmo aborda la detección de novedades, entendida en este contexto como la detección de eventos raros; es decir, aquellos eventos que ocurren esporádicamente y con un número de muestras muy reducido. El problema es que la forma habitual de entrenar un clasificador no funcionará, por lo que la idea es encontrar una función que sea positiva para las regiones con alta densidad de puntos y negativa para las densidades pequeñas (Schölkopf et al., 1999).
- ***Modelo de Markov Oculto (Hidden Markov Model – HMM)***: en este caso se trata de un modelo estadístico en el que se supone que el sistema a modelar es un proceso de Markov de parámetros desconocidos. En un modelo de Markov oculto, el estado no es directamente visible, sino que sólo son visibles las variables influidas por el estado. Cada estado tiene una distribución de probabilidad sobre los posibles símbolos de salida. En consecuencia, la secuencia de símbolos generada por un HMM proporciona cierta información sobre la secuencia (Franzese & Iuliano, 2019).

- ***K-Vecinos Más Cercanos (K-Nearest Neighbors – KNN)***: se clasifican los nuevos objetos según el resultado del objeto más cercano o los resultados de varios objetos más cercanos en el espacio de características del conjunto de entrenamiento. Un objeto se clasifica por el voto mayoritario de sus vecinos, y el nuevo objeto se asigna a la clase más común entre sus k vecinos más cercanos (k es un número entero positivo, y normalmente pequeño). Los vecinos se toman de un conjunto de objetos para los que se conoce la clasificación correcta (Altman, 1992).
- ***Clasificador Bayesiano Ingenuo (Naive Bayes Classifier)***: se basa en el teorema de Bayes con los supuestos de independencia entre predictores. El clasificador Naive Bayes asume que el efecto del valor de un predictor (x) sobre una clase determinada (c) es independiente de los valores de otros predictores, es decir, la independencia condicional de la clase (Webb et al., 2005).
- ***Árboles de decisión (Decision Trees)***: en un árbol de decisión se representan las reglas de decisión inherentes a los datos, mediante una estructura arbórea que divide los datos de forma recursiva. Cada rama del árbol de decisión representa una regla que decide entre un conjunto de valores de un atributo básico (nodos internos) o realiza una predicción de la clase (nodos terminales) (Payne & Meisel, 1977).
- ***Refuerzo de gradiente (Gradient Boosting)*:** se basa en la combinación secuencial de modelos predictivos débiles *(weak learners)* – normalmente árboles de decisión – para crear un modelo predictivo fuerte; cada uno de los árboles de decisión generados corrige los errores

detectados en el árbol de decisión anterior en la secuencia en función del parámetro denominado tasa de aprendizaje *(learning rate)*, el cual controla el grado de mejora de un árbol determinado respecto del anterior (Natekin & Knoll, 2013).

- ***LightGradient Boosting Machine*:** abreviado como LightGBM, es una biblioteca de código abierto que proporciona un entorno de refuerzo de gradiente rápido, descentralizado y de alto rendimiento sobre la base de un algoritmo de un árbol de decisión. Se basa en dos conceptos claves (Ke et al., 2017):
    - *Gradient-based One-Side Sampling (GOSS)*: una versión modificada del método de refuerzo de gradiente que utiliza únicamente grandes instancias de datos, obteniendo estimaciones de ganancia de información precisas con un tamaño de datos mucho menor, reduciendo de esta forma la complejidad computacional del modelo.
    - *Exclusive Feature Bundling (EFB):* se trata de un método de selección de características en el cual se agrupan características dispersas (en su mayoría nulas) que son mutuamente excluyentes entre sí.
- ***CatBoost:*** se trata de una biblioteca de código abierto que implementa un árbol simétrico que maneja características categóricas y ayuda a reducir los tiempos de predicción. Introduce el *Muestreo de Varianza Mínima (Minimal Variance Sampling - MVS)*, una versión ponderada del muestreo de refuerzo de gradiente estocástico que se utiliza para

normalizar los modelos de refuerzo. El uso de MVS reduce el número de ejemplos necesarios para cada iteración de refuerzo y mejora en gran medida la calidad del modelo, haciéndolo más general y con menos probabilidades de sobreajuste (Prokhorenkova et al., 2018).

2.3.3. Redes Neuronales

Una red neuronal es una red o circuito de neuronas, o en un sentido moderno, una red neuronal artificial, compuesta por neuronas o nodos artificiales (Hopfield, 1982). Las conexiones entre las neuronas se modelan como pesos. Un peso positivo refleja una conexión excitatoria, mientras que un peso negativo es indicativo de una conexión inhibitoria. Existe una combinación lineal que modifica todas las entradas aplicando los pesos correspondientes y procediendo a la suma de las entradas modificadas. También existe una función de activación que controla la amplitud de la salida. Cuando la red neuronal recibe un nuevo dato anómalo, tendrá dificultades para procesarlo, ya que está entrenada para procesar datos normales, por lo que generará un elevado error cuadrático medio (MSE).

2.3.4. Extracción y selección de características

Las características son las variables específicas que se utilizan como entrada a un algoritmo; pueden ser selecciones de valores brutos de los datos de entrada o bien pueden ser valores derivados de esos datos. La extracción de características parte de un conjunto inicial de datos medidos y construye valores derivados que

pretenden ser informativos y no redundantes, facilitando los pasos posteriores de aprendizaje y generalización. En los modelos de selección y extracción de características, una anomalía se define en función de la redundancia presente en el modelo, es decir, la información semántica se modela de múltiples maneras.

- ***N-Grams***: se trata de una secuencia consecutiva de n elementos que constituyen una muestra de texto; basándose en un modelo lingüístico probabilístico, se predice el siguiente elemento de la secuencia en forma de un modelo de Markov de orden (n-1) (Tomović et al., 2006).
- ***Bolsa de Palabras (Bag of Words – BOW)***: se codifican las palabras del texto (que representan características categóricas) en vectores de valor real, formando una lista de palabras únicas en el corpus de texto denominada vocabulario. Cada frase o documento puede representarse como un vector con un valor de 1 si la palabra está presente en el vocabulario o de 0 en caso contrario. Otra representación puede hacerse contando el número de veces que aparece cada palabra en el documento, utilizando la técnica *Term Frequency-Inverse Document Frequency (TF-IDF)* (Manning et al., 2008; Stephen, 2004).

2.3.5. Distribución de los atributos de los caracteres

El modelo de distribución de los caracteres de los atributos capta el concepto de un parámetro de consulta “normal” o “regular” observando su distribución de caracteres. El enfoque se basa en la observación de que los atributos tienen una

estructura regular, son en su mayoría legibles para el ser humano y casi siempre contienen sólo caracteres imprimibles. En el caso de los ataques que envían datos binarios, se puede observar una distribución de caracteres completamente diferente. Los caracteres de los atributos regulares se dibujan utilizando los valores correspondientes de la tabla ASCII (Kruegel et al., 2005).

- ***Distribución de Caracteres Idealizada (Idealized Character Distribution – ICD)*:** La IDC se obtiene durante la fase de entrenamiento a partir de las peticiones normales enviadas a la aplicación web, calculándose como el valor medio de todas las distribuciones de caracteres. Durante la fase de detección, se evalúa la probabilidad de que la distribución de caracteres de una secuencia sea una muestra real extraída de su ICD mediante la métrica Chi-Cuadrado (Kruegel et al., 2005).

2.3.6.Reducción de dimensionalidad

La reducción de la dimensionalidad es el proceso de reducción del número de variables aleatorias consideradas mediante la obtención de un conjunto de variables principales (Roweis & Saul, 2000). Principales técnicas de reducción de dimensionalidad:

- ***Análisis de Componentes Principales (Principal Component Analysis – PCA)***: es la técnica de reducción de dimensionalidad más utilizada; funciona reduciendo linealmente los datos existentes a un

espacio de menor dimensionalidad, tratando de preservar la máxima varianza de los datos en el espacio de menor dimensionalidad. PCA se define matemáticamente como una transformación lineal ortogonal que transforma los datos a un nuevo sistema de coordenadas tal que la mayor varianza por alguna proyección escalar de los datos viene a recaer en la primera coordenada, la segunda mayor varianza en la segunda coordenada, y así sucesivamente (Jolliffe, 2002).

- ***Análisis Discriminante Lineal (Linear Discriminant Analysis – LDA)***: se trata de una generalización del discriminante lineal de Fisher para encontrar una combinación lineal de características que caracterice o separe dos o más clases de objetos o eventos; es decir, se expresa una variable dependiente como una combinación lineal de otras características o medidas. El objetivo de un LDA es proyectar un espacio de características n dimensional en un subespacio más pequeño k donde $k \leq n-1$ (McLachlan, 2004).
- ***Mapa de Difusión (Diffusion Map)***: los mapas de difusión no son lineales y se centran en descubrir la variedad subyacente, es decir, la "superficie" restringida de menor dimensión en la que se insertan los datos. Consigue reducir la dimensionalidad reorganizando los datos según los parámetros de su geometría subyacente. Un mapa de difusión transforma los datos en un espacio de menor dimensión, de manera que la distancia euclidiana entre puntos se aproxima a la distancia de difusión en el espacio de características original. La dimensión del espacio de difusión viene determinada por la estructura geométrica

subyacente a los datos y la precisión con la que se aproxima la distancia de difusión (Coifman & Lafon, 2006).

2.4. Datasets

Uno de los mayores problemas a la hora de realizar experimentos para evaluar las técnicas de detección de anomalías en la web es la falta de un marco adecuado que garantice la reproducibilidad de los experimentos y la validez de las conclusiones obtenidas; dicho marco, debería contar como uno de sus principales componentes, con uno o varios conjuntos de datos con registros normales y de ataque actualizados. Uno de los intentos de establecer un marco adecuado fue el de DARPA/MIT Lincoln Lab en 1998 y 1999 (M. Lichman, 2000) y el conjunto de datos de la KDD Cup derivado de ellos. Sin embargo, estos marcos adolecen de importantes limitaciones y han sido criticados en repetidas ocasiones (Sommer & Paxson, 2010). La falta de datos públicos se explica por la naturaleza sensible de los datos: la inspección del tráfico de red puede revelar información altamente sensible de una organización. Debido a la falta de datos públicos, los investigadores se ven obligados a montar sus propios conjuntos de datos, generalmente sin acceso a redes de tamaño adecuado: la actividad encontrada en una pequeña red de laboratorio no puede generalizarse a una red de mayor escala (Sommer, 2008).

3. Contribución

3.1. Revisión sistemática de literatura

Como consecuencia de la investigación llevada a cabo para evaluar el estado del arte de la detección de ataques mediante técnicas de aprendizaje automático y/o detección de anomalías, se confeccionó y publicó la primera Revisión Sistemática de Literatura disponible dentro del corpus científico, específicamente orientada a la prevención y detección de ataques web mediante detección de anomalías. El artículo, cuyo título es *"Prevention and Fighting against Web Attacks through Anomaly Detection Technology. A Systematic Review"*, se publicó en el Special Issue *"Security on Web-Based Applications: Technologies, Methodologies, Analysis Methods and Recent Advances in Cybersecurity"* de la revista *Sustainability*. El factor de impacto de la revista es 3.251(JIF), 0.56(JCI).

3.2. Metodología para la comparación de herramientas de protección web

Como parte de la labor de investigación realizada, se detectó la ausencia de una metodología adecuada que permitiera la comparación en términos de eficacia de herramientas de protección contra ataques a aplicaciones y servicios web; a partir de la detección de esta carencia, se decidió el desarrollo de una metodología específica para la comparación de la eficacia de diversas herramientas de

protección, permitiendo una clasificación objetiva de las mismas mediante el uso de diversas métricas.

Adicionalmente al desarrollo de esta metodología, se realizó una comparativa de la eficacia de diferentes herramientas WAF y RASP, ya que se trata de dos de los tipos de herramientas más usadas en la detección y prevención de ataques contra aplicaciones y servicios web. Las herramientas WAF incluidas en la comparativa fueron *ModSecurity*[4] e *Imperva*[5], mientras que en las herramientas RASP se incluyó a *Fortify App Defender*[6] y *Contrast*[7].

Siguiendo la metodología diseñada, las herramientas evaluadas se interpusieron entre un escáner de vulnerabilidades (en este caso *OWASP ZAP*[8]) y un banco de pruebas simulando una aplicación web con diferentes tipos de vulnerabilidades (se eligió el proyecto *OWASP Benchmark*[9] y *The Web Application Vulnerability Scanner Evaluation Project - Wavsep*[10]), seleccionando una serie de casos vulnerables y no vulnerables de los bancos de prueba, de tal forma que los casos vulnerables sirvan para medir los Verdaderos Positivos (TP) y los no vulnerables los Verdaderos Negativos (TN).

Desde la herramienta OWASP ZAP, con las herramientas de protección interpuestas desactivadas, se lanzaron diferentes ataques contra los casos vulnerables y no vulnerables seleccionados. De esta forma, se identificaron las

[4] https://github.com/SpiderLabs/ModSecurity
[5] https://www.imperva.com/products/web-application-firewall-waf/
[6] https://www.microfocus.com/en-us/cyberres/application-security
[7] https://www.contrastsecurity.com/contrast-protect
[8] https://owasp.org/www-project-zap/
[9] https://owasp.org/www-project-benchmark/
[10] https://github.com/sectooladdict/wavsep

peticiones web lanzadas desde la herramienta OWASP ZAP que generaron alertas en los casos vulnerables (TP), así como las peticiones web que no generaron alertas en los casos no vulnerables (TN).

Posteriormente, y ya con las herramientas de protección activadas, volvieron a lanzarse nuevamente los ataques contra los casos vulnerables y no vulnerables seleccionados anteriormente. Una vez finalizados los ataques, se revisaron los logs de las herramientas de protección para identificar las peticiones web que previamente (con las herramientas de protección desactivadas) habían generado TP y TN, para de esta forma extraer los TP, TN, FP, FN. Los resultados obtenidos por cada una de las herramientas evaluadas se tabularon y ordenaron mediante el índice F-Score.

Este trabajo de investigación se detalló en un artículo titulado *"Systematic Approach For Web Protection Runtime Tools' Effectiveness Analysis"*, el cual se publicó en la revista *Computer Modeling in Engineering & Sciences"* (CMES). El factor de impacto de la revista es 2.027(JIF), 0.66(JCI).

3.3. Construcción de un dataset multietiqueta

Una de las conclusiones más significativas extraídas de la Revisión Sistemática de Literatura llevada a cabo para evaluar el estado del arte, fue la falta de datasets públicos, con datos procedentes de tráfico real, actualizados y correctamente etiquetados para permitir el entrenamiento de modelos capaces de

clasificar correctamente el tráfico como normal o ataque, además de informar del tipo o tipos de ataque que la petición web analizada representa.

Ese fue el punto de partida que condujo a la creación del dataset SR-BH 2020. El conjunto de datos se compone de peticiones web recogidas durante 12 días de julio de 2020 por un servidor web (Wordpress) instalado en una máquina virtual y expuesto a Internet. En este servidor se instaló la versión 2.9.2 de Modsecurity para Apache, con la versión 3.3.0 de Core Rule Set (CRS) en modo *"Detection Only"*, de modo que todas las peticiones (legítimas y maliciosas) se registraron en el log generado por ModSecurity, pero sin ser bloqueadas. Diariamente, los registros generados por ModSecurity se recopilaban y la máquina virtual se restauraba a un estado limpio.

Una vez finalizado el periodo de exposición del servidor web, los registros recogidos fueron procesados semiautomática y manualmente para revisar el etiquetado de peticiones web realizado por Modsecurity, corrigiendo si era necesario la asignación de normalidad/ataque a la solicitud web correspondiente y asegurando una asignación de clasificación CAPEC adecuada.

El resultado final es un dataset multietiqueta destinado especialmente a la detección de ataques web y compuesto por 907.814 solicitudes, de las cuales 525.195 son solicitudes normales y 382.619 son solicitudes anómalas, donde cada registro tiene 24 características diferentes y un conjunto de 13 etiquetas, correspondiendo 12 de ellas a las posibles clasificaciones CAPEC, siendo la etiqueta restante indicadora de la normalidad o no de la petición; véase la Tabla 1 para un detalle de las diferentes etiquetas incluidas en el dataset, así como el número de

peticiones web que se clasificaron bajo una etiqueta concreta: nótese que este número será superior al número total de peticiones del dataset, debido al hecho de que existen peticiones web que son una combinación de más de un tipo de ataque (véase la Tabla 2 para más detalle).

Clasificación CAPEC	Número de peticiones	% sobre el total
000 - Normal	525.195	57,85%
272 – Protocol Manipulation	9.153	1,01%
242 – Code Injection	15.827	1,74%
88 – OS Command Injection	7.482	0,82%
126 – Path Traversal	20.992	2,31%
66 – SQL Injection	250.311	27,57%
16 – Dictionary-based Password Attack	1.847	0,20%
310 – Scanning for vulnerable software	2.718	0,30%
153 – Input Data Manipulation	2.272	0,25%
274 – HTTP Verb Tampering	5.437	0,60%
194 – Fake the source of data	56.145	6,18%
34 – HTTP Response Splitting	19.738	2,17%
33 – HTTP Request Smuggling	1.059	0,12%

Tabla 1. Número de peticiones web agrupadas por clasificación CAPEC

Número de clasificaciones CAPEC diferentes	Número de peticiones web
1	898.576
2	8.132
3	1.088
4	18

Tabla 2. Número de clasificaciones CAPEC diferentes asignadas a una petición web

Para proteger los datos personales de los usuarios que accedieron al servidor web, se configuró el entorno para que todas las peticiones web pasaran por un router interpuesto entre el servidor web e Internet: de este modo, todas las

solicitudes recibidas por el servidor web parecían proceder de la dirección IP local del router.

3.4. Preprocesado de los datos, transformación numérica y extracción de características

Antes de realizar el entrenamiento de los diferentes modelos de aprendizaje automático, es necesario realizar una revisión y preprocesado de los datos del dataset para evitar datos incoherentes y/o duplicados, corregir errores y, al mismo tiempo, adaptar los datos a la codificación numérica para que sean utilizables para los modelos de aprendizaje automático. La transformación numérica de los datos se llevó a cabo mediante un procedimiento inspirado en los trabajos de Kozik et al. (Kozik et al., 2015), Kruegel et al. (Kruegel et al., 2002) y Kruegel y Vigna (Kruegel & Vigna, 2003), calculando la media de la suma de los valores ASCII de los caracteres (aplicando una transformación a minúsculas) de cada uno de los campos de cada petición web. De esta forma, aquellos campos con una alta presencia de caracteres anómalos (como "../../..", presentes en un típico intento de ataque de "path traversal") obtendrán valores ASCII medios diferentes a los de aquellos campos de una petición web normal. Véase el proceso detallado de transformación numérica en la Figura 4.

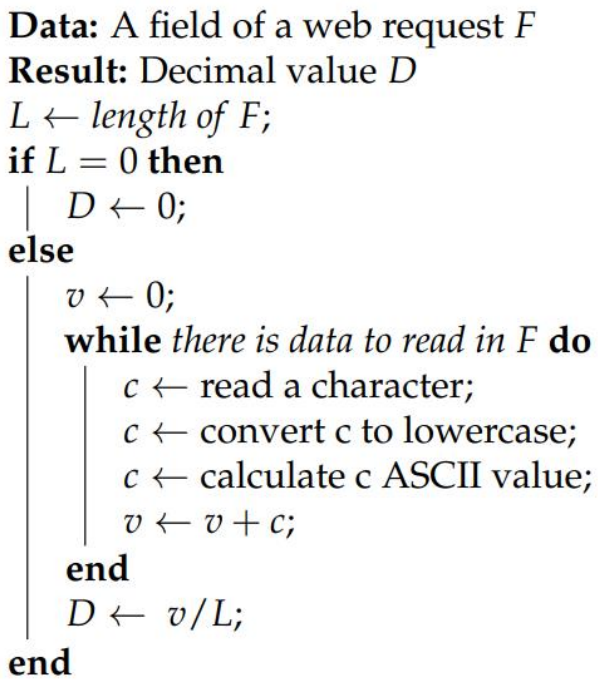


Figura 4. Cálculo de la media de la suma de los valores ASCII de los caracteres de un campo de una petición web

Una vez calculados los valores ASCII medios de cada característica, se genera un histograma para cada campo de forma que sea posible determinar y eliminar aquellas características que no aportan información diferencial. Como puede verse en el gráfico A de la Figura 5, la característica *cookie_value* aporta información útil para permitir la diferenciación de las peticiones web, ya que sus valores numéricos se distribuyen en los rangos 80-90 y 100-110. Sin embargo, en el gráfico B se observa que la característica *do_not_track_value* no aporta ninguna información útil, ya que todas las peticiones web tienen el mismo valor.

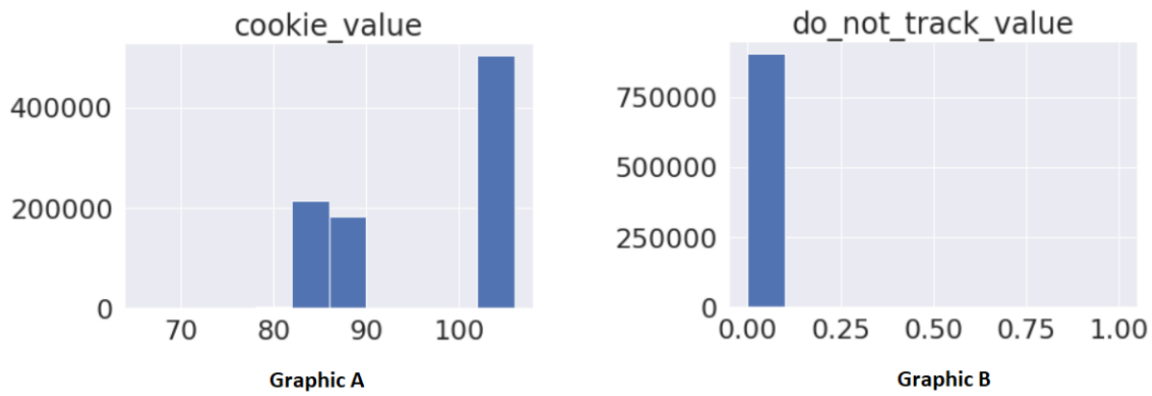


Figura 5. Histograma de los niveles de información de las características

Una vez eliminadas las características que no aportan información útil, se realiza una aproximación a la distribución normal de cada característica restante aplicando el logaritmo natural a cada de ellas y normalizando posteriormente sus valores mediante la clase *StandardScaler* de la biblioteca *Scikit-learn* (Pedregosa et al., 2011).

Finalmente, utilizando el algoritmo de clasificación *Random Forest* y la clase *RFECV* (Guyon et al., 2002) de la librería *Scikit-learn*, se realiza una eliminación recursiva de características mediante validación cruzada.

El proceso de generación del dataset y extracción de características, así como su uso en el entrenamiento y evaluación de los modelos de clasificación multietiqueta presentados en la sección 3.5, se detalla en el artículo titulado *"A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques"*, publicado en la revista *Computers & Security*. El factor de impacto de la revista es 5.105(JIF), 1.39(JCI).

3.5. Modelos de clasificación multietiqueta

A partir de los datos recopilados en el dataset *SR-BH 2020*, del procesamiento numérico de los mismos y de la extracción de características significativas, se procede al diseño de diversos modelos de clasificación multietiqueta de diferentes fases, combinados con los algoritmos de aprendizaje automático LightGBM y CatBoost. Posteriormente, se procede al entrenamiento de los modelos diseñados, usando los datos del dataset SR-BH 2020. El objetivo principal de esta fase es determinar qué combinación de modelo/algoritmo proporciona los mejores niveles de precisión en la predicción y clasificación de ataques web.

El dataset SR-BH 2020 consta de un conjunto de 13 etiquetas: la primera etiqueta del conjunto informa sobre la normalidad de la petición web, de tal forma que si el valor de la primera etiqueta es 1 (petición normal), puede asumirse que el valor de las 12 etiquetas restantes debería ser 0. En el caso de que el valor de la primera etiqueta del conjunto sea 0 (posible ataque), debería haber una o más etiquetas del conjunto restante con un valor igual a 1 (informando de la clasificación CAPEC del ataque). Estas asunciones permiten establecer una división de los modelos de clasificación por fases en función de los valores presentes en la primera etiqueta del conjunto. A continuación, se detallan los modelos y algoritmos analizados, así como la combinación de modelo/algoritmo y los resultados obtenidos por cada combinación.

3.5.1. Modelo de una única fase

En este caso se intenta predecir todo el conjunto de etiquetas asignado a la petición web, independientemente del valor obtenido por la primera etiqueta del conjunto; es decir, aunque la primera etiqueta indique que se trata de una petición web normal, se intentará predecir el resto de las etiquetas del conjunto. En este modelo se usan las clases *skmultilearn.problem_transform.BinaryRelevance* y *skmultilearn.problem_transform.ClassifierChain* de la librería *scikit-multilearn* (Szymański & Kajdanowicz, 2018).

La clase *skmultilearn.problem_transform.BinaryRelevance* transforma un problema de clasificación con N etiquetas en N problemas individuales de clasificación binaria, ignorando las correlaciones que existen en los datos de entrenamiento (se asume la independencia entre las distintas etiquetas).

Por su parte, la clase *skmultilearn.problem_transform.ClassifierChain* construye una secuencia condicional de clasificadores de etiquetas bayesianos, tal y como describen Read et al., formando una cadena de clasificadores binarios en la que cada clasificador de la cadena es responsable de aprender y predecir la asociación binaria de la etiqueta dado el espacio de características, modificado por todas las predicciones binarias anteriores sobre las etiquetas de la cadena (Read et al., 2011).

3.5.2. Modelo de dos fases

En este caso se realizará la predicción de los valores del resto de etiquetas del conjunto únicamente si el valor de la primera etiqueta es 0 (posible ataque). En el caso de que el valor de la primera etiqueta sea 1 (petición web normal), se asumirá directamente que todas las etiquetas restantes del conjunto tendrán un valor de 0.

La predicción de los valores de las etiquetas, se realizará mediante la clase *skmultilearn.problem_transform.BinaryRelevance* y la clase *MultiOutputClassifier,* del módulo *sklearn.multioutput,* integrado en la librería *Scikit-learn* (Pedregosa et al., 2011).

3.5.3. Modelo personalizado

Se genera un modelo personalizado en el que se calculan los mejores hiperparámetros para la clasificación de cada una de las etiquetas utilizando GridSearchCV con los algoritmos LightGBM y CatBoost. La predicción de cada etiqueta se realizará con el algoritmo correspondiente ajustado con los hiperparámetros calculados.

3.5.4. Métricas, escenarios y resultados obtenidos

La mayoría de los algoritmos de clasificación se basan en clasificación binaria o multiclase; en estos casos, las métricas clásicas de clasificación (F-Score,

Precision, Recall, etc.) son adecuadas. Sin embargo, en modelos de clasificación multietiqueta las métricas clásicas deben complementarse, ya que en estos modelos se introduce la noción de predicciones parcialmente correctas, de tal forma que es posible realizar predicciones correctas de un porcentaje del total de las etiquetas de cada observación (Gouk et al., 2016; Zhang & Zhou, 2014).

Por otra parte, es necesario que las métricas tengan sentido respecto al modelo evaluado y a la criticidad del escenario en el cual el modelo deba ser aplicado (Antunes & Vieira, 2015); así pues, las diferentes combinaciones de modelos y algoritmos diseñadas en este trabajo, se evalúan en cuatro diferentes escenarios de criticidad, aplicándose en cada uno de ellos la métrica de evaluación recomendada.

3.5.4.1. Métricas

Accuracy (para cada etiqueta): Es la medida de precisión con la que se evaluará la eficacia del modelo en la predicción de cada una de las etiquetas que componen una observación, aplicando la definición clásica de Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy – Exact match (EMR): es la medida de precisión que se utilizará para evaluar el rendimiento global del modelo; en nuestro caso, la medida exacta de la predicción para todas las etiquetas que componen una observación, ignorando las predicciones parcialmente correctas (considerándolas incorrectas) y

extendiendo el concepto de Accuracy utilizado para la predicción de una sola etiqueta al caso de varias etiquetas.

$$EMR = \frac{Núm.\,de\ observaciones\ con\ predicción\ exacta\ de\ sus\ etiquetas}{Número\ total\ de\ observaciones}$$

Precision: indica el nivel en que las predicciones del modelo se ajustan a la realidad: el cociente entre las predicciones realizadas por el modelo que coinciden con la clase positiva real (verdaderos positivos – TP) y todas las predicciones positivas producidas por el modelo (verdaderos positivos TP + falsos positivos – FP). Para tener en cuenta el desequilibrio de etiquetas, se calcula la proporción entre el número de etiquetas positivas correctamente predichas y el total de etiquetas positivas predichas promediadas en todos los casos, ponderado por el soporte (el número total de casos para cada etiqueta).

$$Precision = \frac{TP}{TP + FP}$$

$$Precision_{Ponderada} = \sum_{clases} Peso\ de\ las\ clases * Precision\ de\ cada\ clase$$

Recall: se trata del porcentaje de Verdaderos Positivos identificados correctamente, promediado en todas las instancias y ponderado por el soporte.

$$Recall = \frac{TP}{TP + FN}$$

$$Recall_{Ponderada} = \sum_{clases} Peso\ de\ las\ clases * Recall\ de\ cada\ clase$$

F-Measure (para cada etiqueta): es una medida de la exactitud de una prueba, teniendo en cuenta tanto Precision como Recall. También conocido como F1-Score o F-Score, este valor es una media armónica equilibrada de dos métricas: Precisión (P) y Recall (R)

$$F - Measure = \frac{2 \cdot P \cdot R}{P + R}$$

F-Measure (ponderada): puntuación de la medida F promediada en todas las instancias y ponderada por el soporte.

$$F - Measure_{Ponderada} = \sum_{clases} Peso\ de\ las\ clases * F - Measure\ de\ cada\ clase$$

Informedness: es una medida de cuánta información proporciona el sistema sobre las etiquetas positivas y negativas, es decir, cuán informado está un predictor para la condición deseada, en contraposición al azar, promediado en todas las instancias y ponderado por el soporte.

$$Informedness = \frac{TP}{FN + TP} + \frac{TN}{FP + TN} - 1 = \frac{TP}{FN + TP} - \frac{FP}{FP + TN}$$

$$Informedness_{Ponderada} = \sum_{clases} Peso\ de\ las\ clases * Informedness\ de\ cada\ clase$$

Markedness: es una medida de la confianza en las predicciones positivas y negativas del sistema; cuantifica la consistencia con la que el resultado incluye una variable de predicción como marcador, es decir, cómo la condición etiquetada para un determinado predictor se compara con el azar, promediado en todas las instancias y ponderado por el soporte.

$$Markedness = \frac{TP}{TP + FP} + \frac{TN}{FN + TN} - 1 = \frac{TP}{TP + FP} - \frac{FN}{FN + TN}$$

$$Markedness_{Ponderada} = \sum_{clases} Peso\ de\ las\ clases * Markedness\ de\ cada\ clase$$

ROC AUC: muestra la eficiencia global de un modelo de clasificación en todos los niveles de clasificación, trazando la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR). El AUC es una métrica bidimensional del área bajo la curva ROC completa, que oscila entre 0 (predicciones 100% inexactas) y 1 (predicciones 100% precisas), reflejando los grados de separabilidad y mostrando la capacidad de un determinado modelo para distinguir entre clases (Sureda Riera et al., 2020; Swets, 1996). En nuestro caso, el AUC del ROC se promedia entre todas las instancias y se pondera por el soporte.

Hamming Loss: es la proporción de etiquetas predichas incorrectamente sobre el número total de etiquetas. En la clasificación multietiqueta, Hamming Loss se calcula como la distancia de Hamming entre los valores verdaderos y los

predichos. Su valor oscila entre 0 y 1. Cuanto menor sea el valor, mejor será el rendimiento del modelo. Sea D un dataset con L etiquetas compuesto de $|D|$ observaciones multietiqueta $(x_i, Y_i), i = \{1,2,..|D|\}, Y_i \subset L$. Sea H un clasificador multietiqueta y $Z_i = H(x_i)$ el conjunto de etiquetas predichas por el clasificador H para la observación x_i

$$Hamming\ Loss(H, D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \Delta Z_i|}{|L|}$$

Donde Δ representa la diferencia simétrica de los conjuntos de etiquetas y corresponde a la operación XOR en lógica booleana (Schapire & Singer, 2000).

Jaccard Similarity: mide el grado de similitud entre dos conjuntos examinando la proporción de etiquetas positivas correctamente predichas en un conjunto potencialmente positivo (positivo esperado y positivo real).

$$J(T, P) = \frac{T\ \cap P}{T\ \cup P}$$

Donde T, P son las etiquetas reales y las etiquetas predichas respectivamente.

3.5.4.2. Escenarios

Escenario de muy alta criticidad: representa el desarrollo y la evaluación de aplicaciones críticas que tienen requisitos de seguridad muy elevados ya que necesitan proporcionar a sus clientes un sistema fiable. Ejemplos de este escenario

son los sitios web de los bancos en Internet, el comercio de acciones o los sistemas de comercio electrónico masivo. La máxima prioridad en este escenario es la eliminación del mayor número de ataques, asumiendo así la inversión de tiempo y recursos en remediar la ocurrencia de ataques inexistentes (Falsos Positivos). En este caso, la métrica óptima es **Recall**, ya que maximiza la detección de los ataques.

Escenario de alta criticidad: en este caso, el objetivo es lograr un equilibrio entre la prioridad de detectar y eliminar el máximo número de ataques y evitar la notificación excesiva de Falsos Positivos, ya que los recursos en este tipo de escenario deben gestionarse adecuadamente. Los requisitos de seguridad son altos, pero inferiores a los del escenario de alta criticidad. Las aplicaciones web de comercio electrónico son uno de los ejemplos de este tipo de escenario. Una métrica de elección en este escenario podría ser F-Score, pero el problema es que asigna la misma importancia a la Precisión y a Recall; la métrica **Informedness** parece ser una mejor alternativa, ya que no tiene sesgos y no presenta las desventajas del promedio armónico.

Escenario de criticidad media: este escenario presenta aplicaciones menos expuestas o críticas, normalmente con un presupuesto limitado, por lo que los recursos disponibles para remediar los ataques reportados son limitados. Por esta razón, tanto encontrar y eliminar el mayor número posible de ataques como ahorrar recursos en la reparación de los Falsos Positivos tienen la misma importancia. Ejemplos de este tipo de escenario son los sitios web, en los que los ataques suponen menores pérdidas económicas, o las aplicaciones de intranet que

son menos propensas a los ataques externos. La métrica adecuada para este escenario es **F-Score**.

Escenario de baja criticidad: este escenario representa las aplicaciones no críticas que no están excesivamente expuestas a ataques. Se caracterizan por estar implementadas con presupuestos reducidos, por lo que los recursos disponibles son limitados; el uso de estos recursos debe centrarse en los ataques confirmados. El objetivo es reportar el menor número posible de Falsos Positivos, aumentando la confianza en los ataques reportados. Ejemplos de estos escenarios son los portales web de las pequeñas y medianas empresas. La métrica **Markedness** parece ser la más adecuada en este escenario, ya que es capaz de dar mayor preponderancia a los valores de Accuracy, al tiempo que es capaz de considerar los ataques que quedan sin notificar.

3.5.4.3. Resultados obtenidos

En este trabajo se ha llevado a cabo la evaluación de las diferentes combinaciones de modelos y algoritmos desde dos ópticas diferentes: en primer lugar, se han evaluado desde una perspectiva comparativa en términos de eficacia de las distintas combinaciones; es decir, se ha evaluado qué combinaciones de modelos y algoritmos presentan mejores valores en términos de eficacia y eficiencia valorados en función de las distintas métricas descritas en la Sección 3.5.4.1. En segundo lugar, se ha evaluado las combinaciones de modelos y algoritmos más adecuados en los diferentes escenarios de criticidad, detallados en

la Sección 3.5.4.2, siguiendo las recomendaciones del trabajo de Antunes y Vieira (Antunes & Vieira, 2015).

En la Tabla 3 se aporta un resumen de las diferentes métricas obtenidas por las distintas combinaciones de modelos, clases y algoritmos. A partir de los datos relacionados en ella, se puede observar que la combinación del modelo de dos fases y el algoritmo CatBoost en la que obtiene mejores resultados en todas las métricas; dentro de esta combinación de modelo/algoritmo puede observarse como la clase *sklearn.multioutput.MultiOutputClassifier* es la que obtiene mejores resultados en todas las métricas, excepto en la métrica *Precision*, métrica en la que la clase *c* es la que obtiene mejores resultados.

En la Tabla 4 se detallan las métricas recomendadas según los escenarios de criticidad en que se aplica el modelo de protección (Antunes & Vieira, 2015). Puede observarse que, en todos los escenarios contemplados, la combinación del modelo de dos fases y el algoritmo CatBoost es la que obtiene mejores resultados. La clase *sklearn.multioutput.MultiOutputClassifier* es la que obtiene mejores resultados en todos los escenarios, excepto en el escenario de baja criticidad, en el que es superada por la clase *sklearn.multioutput.MultiOutputClassifier*.

Modelo	Clase	Algoritmo	Accuracy EMR	Precision	Recall	F-Measure	ROC AUC	Hamming Loss	Jaccard Similarity	Informedness	Markedness
Una Fase	BinaryRelevance	LightGBM	0,84419	0,89927	0,85112	0,87204	0,8982	0,01869	0,78439	0,80532	0,81899
		CatBoost	0,84939	0,90279	0,85734	0,87221	0,90139	0,01801	0,79153	0,77943	0,80261
	ClassifierChain	LightGBM	0,87224	0,8761	0,8736	0,87227	0,90421	0,01958	0,78432	0,81238	0,80567
		CatBoost	0,87213	0,87876	0,87343	0,87171	0,90281	0,01957	0,78264	0,81973	0,81282
Dos Fases	BinaryRelevance	LightGBM	0,84782	0,90075	0,85049	0,87216	0,89768	0,0186	0,78468	0,80426	0,82138
		CatBoost	0,85201	**0,90515**	0,85508	0,8768	0,90055	0,01797	0,791	0,81007	**0,82741**
	MultiOutputClassifier	LightGBM	0,88095	0,89137	0,88641	0,88615	0,91139	0,01754	0,80214	0,832	0,82028
		CatBoost	**0,88445**	0,89557	**0,88829**	**0,88912**	**0,91322**	**0,01703**	**0,80672**	**0,8358**	0,82563
Personalizado	-	LightGBM	0,85888	0,86108	0,8627	0,85874	0,87702	0,0214	0,76356	0,76248	0,79066
		CatBoost	0,88436	0,88853	0,8879	0,88501	0,90887	0,01756	0,80115	0,8269	0,81941

Tabla 3. Resumen de las métricas obtenidas por cada combinación de Modelo, Clase y Algoritmo.

Escenario	Métrica	Mejores Valores	Combinación Modelo - Clase - Algoritmo
Muy alta criticidad	Recall	0,88829	Dos fases - MultiOutput - CatBoost
		0,8879	Personalizado - CatBoost
		0,88641	Dos fases - MultiOutput - LightGBM
Alta criticidad	Informedness	0,8358	Dos fases - MultiOutput - CatBoost
		0,832	Dos fases - MultiOutput - LightGBM
		0,8269	Personalizado - CatBoost
Criticidad media	F-Measure	0,88912	Dos fases - MultiOutput - CatBoost
		0,88615	Dos fases - MultiOutput - LightGBM
		0,88501	Personalizado - CatBoost
Baja criticidad	Markedness	0,82741	Dos fases - Binary relevance - CatBoost
		0,82563	Dos fases - MultiOutput - CatBoost
		0,82138	Dos fases - Binary relevance - LightGBM

Tabla 4. Métricas recomendadas por escenario junto con las tres mejores combinaciones de Modelo, Clase y Algoritmo.

3.6. Resumen de la contribución

En esta tesis se realiza una contribución significativa, mediante la publicación de tres artículos en revistas de impacto, que posibilitan el avance de la investigación en el ámbito de la protección de aplicaciones web mediante técnicas de detección de anomalías.

Como parte de este trabajo, se proporciona a la comunidad científica la primera revisión sistemática de literatura sobre el uso de técnicas de detección de anomalías y aprendizaje automático aplicadas a la detección y prevención de ataques web.

Por otra parte, se ha conseguido desarrollar una metodología experimental específica que permite la comparación objetiva de diferentes herramientas de protección web; con el objeto de verificar la aplicabilidad de esta nueva metodología, se compararon diversas herramientas de protección WAF y RASP**,** en función de las puntuaciones obtenidas en la métrica F-Score.

Además, se ha puesto a disposición de la comunidad científica el dataset SR-BH 2020 (Sureda Riera et al., 2022), primer dataset multietiqueta destinado al entrenamiento de modelos de aprendizaje automático para la protección de aplicaciones y servicios web, el cual informa de la clasificación CAPEC de los ataques presentes en una determinada petición web.

A partir de las observaciones que constituyen el dataset SR-BH 2020, se ha desarrollado una nueva técnica que permite la extracción de características relevantes para el entrenamiento de los modelos de aprendizaje automático,

mediante el cálculo de la media de la suma de los valores ASCII de los caracteres de cada uno de los campos que componen una petición web.

Se han diseñado, entrenado y evaluado diversos modelos de clasificación multietiqueta que proporcionan información acerca del tipo de ataque o ataques que una aplicación web está sufriendo; mediante el uso de estos modelos, el operador de seguridad recibe una alerta en la que se especifica la clasificación o clasificaciones CAPEC del ataque o ataques. Véase la Figura 6 para más detalle.

```
In [18]: %%time
         prediccion([[5.30460883087438,
          1.043082871047808,
          0.623192182505229,
          5.216494102681128,
          0.903506237388453,
          -1.2130669975644175,
          -5.68336728338642,
          -0.9974627487855351,
          -0.5428089419695353]])

         CPU times: user 68.6 ms, sys: 0 ns, total: 68.6 ms
         Wall time: 21 ms

Out[18]: ([0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          ['272 - Protocol Manipulation', '88 - OS Command Injection'])
```

Figura 6. Clasificación CAPEC proporcionada por un modelo al detectar un ataque.

4.Artículos publicados

4.1. Artículo 1

sustainability

MDPI

Article

Prevention and Fighting against Web Attacks through Anomaly Detection Technology. A Systematic Review

Tomás Sureda Riera [1], Juan-Ramón Bermejo Higuera [2], Javier Bermejo Higuera [2], José-Javier Martínez Herraiz [1] and Juan-Antonio Sicilia Montalvo [2,*]

1 Computer Science Department, University of Alcalá, Alcalá de Henares, 28805 Madrid, Spain; tomas.sureda@uah.es (T.S.R.); josej.martinez@uah.es (J.-J.M.H.)
2 Escuela Superior de Ingeniería y Tecnología (ESIT), Universidad Internacional de la Rioja (UNIR), Logroño, 26006 La Rioja, Spain; juanramon.bermejo@unir.net (J.-R.B.H.); javier.bermejo@unir.net (J.B.H.)
* Correspondence: juanantonio.sicilia@unir.net

Received: 25 May 2020; Accepted: 12 June 2020; Published: 17 June 2020

Abstract: Numerous techniques have been developed in order to prevent attacks on web servers. Anomaly detection techniques are based on models of normal user and application behavior, interpreting deviations from the established pattern as indications of malicious activity. In this work, a systematic review of the use of anomaly detection techniques in the prevention and detection of web attacks is undertaken; in particular, we used the standardized method of a systematic review of literature in the field of computer science, proposed by Kitchenham. This method is applied to a set of 88 papers extracted from a total of 8041 reviewed papers, which have been published in notable journals. This paper discusses the process carried out in this systematic review, as well as the results and findings obtained to identify the current state of the art of web anomaly detection.

Keywords: anomaly detection; web attacks; systematic review

1. Introduction & Motivation

Web applications have changed our way of life, allowing daily operations such as making bank transfers, booking a flight, making online purchases, etc. The detection of attacks on web applications, with the purpose of safeguarding their integrity, confidentiality and availability, has become an area of special interest.

According to Liao et al. [1], Intrusion Detection Systems (IDS) can generally be divided into three categories based on the detection principle: Signature-based Detection (SD), Anomaly-based Detection (AD) and Stateful Protocol Analysis (SPA). The characteristics of the methods are as follows:

- Signature-based detection (SD): A signature corresponds to a known pattern of attack. In order to detect possible intrusions, the patterns are compared to the captured data. Alternative names for SD are Knowledge-Based Detection or Misuse Detection.
- Anomaly-based detection (AD): An anomaly is detected when there is a deviation from usual behavior, which is represented by profiles (static or dynamic). Anomaly detection is made by comparing the normal profiles with the observed events in order to detect attacks or intrusions. AD is also called Behavior-Based Detection.
- Stateful protocol analysis (SPA): SPA relies on generic profiles for specific protocols developed by the providers. Generally, SPA network protocol models are typically based on standards of protocols from international organizations. This is also known as specification-based detection.

Current scientific literature abounds with *surveys, comparative studies and reviews* of intrusion detection using anomaly detection techniques, for example:

Jyothsna et al. [2] present an overview of the main anomaly-based technologies for network intrusion detection, along with their operational architectures, and also present a classification based on the type of processing that relates to the behavior model of the target system.

Kakavand et al. [3] provided an overview of data mining methods used by HTTP web services anomaly detection, concluding that most studies do not use public datasets that allow replication of the experiments. Those studies that do use public datasets showed high percentages of accuracy in most of the intrusion detection techniques employed, but these studies were not replicated with a different set of datasets.

Samrin and Vasumathi [4], reviewed the results of applying different anomaly detection techniques on KDD Cup 99 dataset.

None of the existing surveys, comparative studies and reviews address in depth a comprehensive review in which the techniques, results, metrics and datasets used are detailed and compared in an objective and critical manner. As the reader will see in detail in Section 4.3 , one of the biggest problems that have been detected when carrying out this systematic review lies in the fact that most of the studies reviewed do not work on public datasets that allow the validation and replication of the experimental results.

To the best of the authors' knowledge, there is currently no *systematic review* of the existing scientific literature, specifically focused on the detection and prevention of web attacks using anomaly detection techniques. Authors fully agreed with Kitchenham and Charters [5] when they state that: *"(. . .) unless a literature review is thorough and fair, it is of little scientific value. This is the main rationale for undertaking systematic reviews. A systematic review synthesises existing work in a manner that is fair and seen to be fair."*

Considering the lack of systematic reviews in this area, it has been decided to undertake a systematic review on using anomaly detection techniques in web attack detection, adopting a formal and systematic procedure for the conduction of the bibliographic review, with the definition of explicit protocols for obtaining information. This systematic review was done following the guidelines of Kitchenham et al. [5–9].

This paper makes the following contributions:

- An extensive and rigorous review of the existing scientific literature on the use of anomaly detection techniques in web attack detection.
- Identification and classification of the papers reviewed according to the types of datasets, techniques, metrics, results, etc. of each one of them.
- The results and metrics obtained by the different anomaly detection techniques studied in the papers reviewed are detailed.
- Identification of opportunities to improve research aimed at the prevention and detection of web attacks through the use of anomaly detection techniques. These opportunities include: generation of publicly available datasets that allow replication and validation of the experimental work, incorporation of metrics such as F-Score, Area Under the Curve (AUC) and Precision that allow complementing the usual metrics in this type of research, better definition of the attacks analyzed in each study since, as will be seen in Section 5, most of the studies reviewed do not detail the types of attack that are attempted to be detected.

The rest of this paper is structured as follows: Section 2 presents the background: related works and research, definitions, overview and theories on anomaly detection technologies. Section 3 presents the planning of the systematic review, the research questions, as well as the method that has been used to carry out the selection and review of the papers of interest. Section 4 presents the results obtained. In Section 5 the findings are discussed. Finally, Section 6 details the conclusions and makes recommendations for future work.

2. Background

Anomaly detection algorithms have broad applications in business, scientific, and security domains where isolating and acting on the results of outlier detection is critical. Firstly, we provide an overview of related work in the area of study, as well as some definitions and classification of anomaly detection technology:

2.1. Related Work

Patel et al. [10] worked on a systematic review of IDS in cloud environments, focusing mainly on the requirements that an Intrusion Detection and Prevention System (IDPS) should meet in order to be deployed in a cloud computing environment. This paper does not specify the use of any systematic methodology for searching bibliographic sources, nor the definition of specific protocols for obtaining information.

Raghav, Chhikara and Hasteer [11] analyzed in a systematic review the approaches of Intrusion Prevention System (IPS) in a cloud computing environment. Again, this systematic review does not indicate the use of any specific methodology for information gathering and does not pose a set of initial questions to be answered.

In 2007, Patcha and Park [12] conducted a survey of anomaly detection techniques, detailing the existing techniques at that time, but without referring to sections such as the datasets used in the studies reviewed, or the metrics used in the validation of the experiments.

In 2009, Chandola, Banerjee and Kumar [13] conducted a survey of the studies carried out on detection of anomalies in a wide range of knowledge domains. Despite being a great work, it is too general and does not include important aspects in the field of study of web attack prevention by detecting anomalies, such as the datasets used, metrics, etc.

In 2018, Jose et al. [14], provide an overview of various aspects of anomaly based host intrusion detection systems.

Fernandes et al. [15] reviewed the most important aspects pertaining to anomaly detection, covering an overview of a background analysis as well as a core study on the most relevant techniques, methods, and systems within the area. They also discussed the description of an IDS and its types.

Kwon et al. [16], investigated deep learning techniques employed for anomaly-based network intrusion detection; however, a review of the datasets is missing, as they only describe the KDD Cup 1999 and NSL-KDD datasets, the former being heavily criticized in several studies [17–20].

In 2018, Ieracitano et al. [21] propose an innovative statistical analysis driven optimized deep learning system for intrusion detection, extracting optimized and more correlated features using big data visualization and statistical analysis methods, followed by a deep autoencoder (AE) for potential threat detection. Specifically, a preprocessing module eliminates the outliers and converts categorical variables into one-hot-encoded vectors. In 2020, Ieracitano et al. [22] combine traditional data analysis and statistical techniques incorporating advances in Machine Learning (ML). Specifically, Deep Learning (DL) technology is employed in conjuction with statistical analysis. In both studies, the NSL-KDD dataset was used.

Khraisat et al. [23] presented a taxonomy of contemporary IDS, a comprehensive review of recent works and an overview of the datasets commonly used. They also presented an overview of evasion techniques used by attackers.

Ahmed, Naser Mahmood and Hu [24] provide an overview of different network anomaly detection techniques, as well as various alternative datasets to KDD Cup 1999 and NSL-KDD. However, datasets such as CSIC 2010 are not included in the work. The metrics used for the validation of the various studies reviewed are also not listed in this study.

The present work aims to remedy the weaknesses of the above-mentioned studies, through a systematic review of the available literature, strictly following the principles of a methodology widely accepted by the scientific community as proposed by Kitcheham and Charters [5], carrying

out an analysis of publicly available datasets, metrics used for the evaluation of results obtained and discussion of techniques used by the various studies reviewed.

2.2. Anomaly Detection Definition

Kotu and Deshpande [25] define anomaly detection as *"the process of finding outliers in a given dataset"*. Outliers are the data objects that stand out amongst other data objects and do not conform to the expected behavior in a dataset. An outlier is a data object that is markedly different from the other objects in a dataset. Hence, an outlier is always defined in the context of other objects in the dataset.

2.3. Types of Anomaly Detection Algorithms

It is common to divide the anomaly detection algorithms according to their purpose. The main categories are listed below: [26,27]:

- Supervised algorithms: Supervised algorithms model the relationships between input data and prediction. They try to predict the output values that will be obtained when feeding the model with new input data. This prediction is based on the relationships learned from the tagged training data. Examples of supervised algorithms are Nearest Neighbor, Naive Bayes, Decision Trees, Linear Regression, Support Vector Machines (SVM), Neural Networks.
- Unsupervised algorithms: As there is no tagged training data on which the algorithm can perform its learning, these algorithms perform pattern detection on the input data. Examples of unsupervised algorithms are association rules and k-means.
- Semi-supervised algorithms: Semi-supervised algorithms use little tagged data and a lot of untagged data as part of the training set. These algorithms try to explore the structural information contained in the unlabeled data to generate predictive models that work better than those that only use labeled data. Common examples of unsupervised algorithms are: Generative models, Low-density separation and Graph-based methods.
- Reinforcement algorithms: The objective is the development of a system (called agent) that is intended to improve its efficiency by performing a certain task based on the interaction with its environment, receiving rewards that allow it to adapt its behavior. As the agent receives rewards, it must develop the right strategy (called policy) that leads it to obtain positive rewards in all possible situations. Common examples of reinforcement algorithms are: Q-Learning, Temporal Difference (TD) and Deep Adversarial Networks.

There are different auxiliary techniques used in the process of detecting anomalies in a given dataset [28–30].

- Feature extraction: N-grams, Bag of Words, multi-features information entropy prediction model.
- Dimensionality reduction: Principal Component Analysis (PCA), Random Projection, Diffusion Maps.
- Parameter estimation: Limited-memory Broyden—Fletcher—Goldfarb—Shanno (L-BFGS) algorithm.
- Regular expression generator: Simple (SREG) and Complex (CREG) Expression Generator.

2.4. Advantages and Disadvantages of Anomaly Detection Algorithms

According to García-Teodoro et al. [31], signature-based schemes provide very good detection results for specified, well-known attacks, but they are not capable of detecting new intrusions, even if they are built as minimum variants of already known attacks. On the contrary, anomaly detection algorithms are capable of detecting previously unseen intrusion events. However, the rate of false positive (FP, events erroneously classified as attacks) in anomaly-based systems is usually higher than in signature-based ones.

3. Review Methodology

This systematic review has been conducted following the guidelines outlined by Kitchenham et al [5–9]. The methodology includes: development of a review protocol, managing the review, analysis and reporting of results, and discussion of findings.

3.1. Preparing the Review

In the generation of the review protocol, we considered the definition and details of the research questions, the bibliographic databases to be included in the search, as well as the methods used to identify and evaluate the papers susceptible to inclusion in our work. In order to carry out the review, we identified the main studies, applying the inclusion and exclusion criteria to them and synthesizing the results. In order to reduce investigator bias, the review protocol was drafted by one of the authors, reviewed by the other authors and finalized by a discussion among all the authors. The online databases were searched widely and their studies are reported. In total, the initial search returned 8041 articles.

3.2. Research Questions

The main objective of our work was to analyze and classify the available scientific literature focused on the detection of web attacks using anomaly detection techniques. For proper planning of the review, a set of research questions were generated. The method for generating the set of research questions is as follows: first, each of the authors contributes those they consider appropriate for the objective of the paper; then, all the proposals contributed are discussed, and finally, the most relevant questions are chosen by agreement of all the authors. In Table 1 we detail the research questions details.

Table 1. Research questions details.

Research question	Details
(1) **What is the current state of web anomaly detection?**	
What kind of attacks are attempted to be detected or prevented?	An overview of the types of web attacks that most concern the scientific community is achieved.
What web anomaly detection methods are used? How often are they cited in specialized literature?	An overview of the status of the different methods and techniques for detecting anomalies in web applications is achieved. The different studies that evaluate and compare the different anomaly detection techniques are explored.
Advantages and disadvantages of different methods and techniques	
(2) **Systematic reviews of anomaly detection in web applications**	
What studies and research have been done	The different studies are quantified and tabulated by each method or technique
Identification of the strengths and lacks of these works	A critical review of the studies provided is made
(3) **Key areas of interest in the different studies carried out in the detection of anomalies in web applications**	
Featured areas, number of studies carried out in each area and main findings	An objective view of the number of studies is obtained for each subarea that helps identify key areas for future research
Evolution of the number of studies carried out in each area over time	A time-based count shows how the key area has evolved over time

3.3. Information Sources

As recommended by [5–9], a wide search in electronic sources was made to increase the probability of finding relevant articles. In our case, the following databases were searched:

- IEEE Xplore (https://ieeexplore.ieee.org/Xplore/home.jsp)
- ScienceDirect (https://www.sciencedirect.com/)
- Springer (https://link.springer.com/)
- Wiley Interscience (https://onlinelibrary.wiley.com/)
- ACM Digital Library (https://dl.acm.org/dl.cfm)

3.4. Search Criteria

An exhaustive search has been carried out on the different online resources; the search sequence included the keywords *"anomaly detection" and ("web" or "network")*. The search covered the papers published in the period from January 1993 to December 2019. The search terms have been searched in the title and abstract of the publication, whenever possible. This phase returned 8,041 results. Table 2 shows the search strategy in the different online resources.

Table 2. Search queries.

Resource	Search Query	Years	Content Type	# Results
ieeexplore.ieee.org	In title and abstract: "anomaly detection" AND (web OR network)	1993–2019	Conferences, Journals, Early Access Articles, Books	4063
sciencedirect.com	In title and abstract: "anomaly detection" AND (web OR network)	1993–2019	Journals, Books	528
link.springer.com	In title: "anomaly detection" AND (web OR network)	1993–2019	Chapter, Conference Paper, Article	962
onlinelibrary.wiley.com	In title and abstract: "anomaly detection" AND (web OR network)	1993–2019	Journals, Books	31
acm.org	In abstract: "anomaly detection" AND (web OR network)	1993–2019	Articles, Proceedings	2457
			Total Results:	**8041**

3.5. Criteria for Inclusion and Exclusion of Papers

Papers were considered for inclusion in the review if their field of study was web anomaly detection. This systematic review includes only quantitative studies written in English. In a first stage, duplicate results have been detected and eliminated. A publication is considered to be duplicated if its Digital Object Identifier (DOI) is equal to the DOI of another publication. In a second phase, irrelevant documents were manually excluded based on their level of relevance. In that case, the number of irrelevant documents is significant, as research articles on the detection of anomalies in biology, medicine, social networks and study disciplines other than computer security are difficult to distinguish from the detection of web anomalies in an online database search. In a later stage, a selection based on title and abstract was done. Papers that do not contain *"web"* or *"http"* in the title or abstract were rejected. Finally, the selected documents were fully read to select a definitive list of documents according to the inclusion/exclusion criteria.

As shown in Figure 1, our search returned over 8041 total papers, which were narrowed down to 6906 after removing duplicates papers. Subsequently, based on their relevance, 1124 papers were selected, from which 189 papers were further selected based on their titles and abstracts. Then, these 189 papers were read entirely to select a final list of 106 papers based on the inclusion and exclusion criterion.

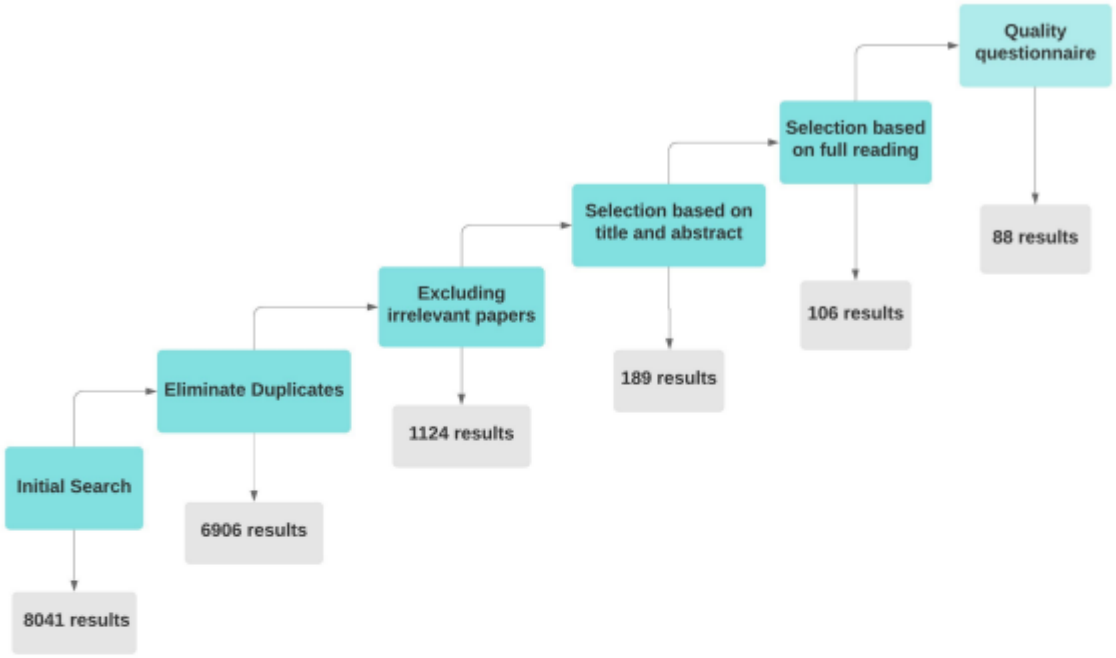


Figure 1. Papers Selection Process.

3.6. Quality Assurance

After applying the inclusion/exclusion criteria to select the relevant papers, an analysis of the quality of the remaining papers was carried out. Following the guidelines cited by [5–9], all studies were evaluated for bias, internal and external validity of results.

A questionnaire Appendix A was made to evaluate the quality of each paper to be included in the systematic review. Each team member evaluated the studies using the quality questionnaire. Only papers that passed the evaluation of the quality questionnaire by unanimity of the five team members were included in the systematic review. After evaluating the papers with the questionnaire, a total of 88 papers conform to this systematic review.

3.7. Quality Evaluation Results

In the last 5 years (2015 to 2019) the interest of the scientific community in anomaly detection techniques applied to web intrusion detection has increased; proof of this is that 67% of the studies analysed (59 out of 88 studies) are concentrated in this period. Figure 2 shows the evolution of the number of papers selected by year. Similarly, Table 3, shows the number of studies selected in each year.

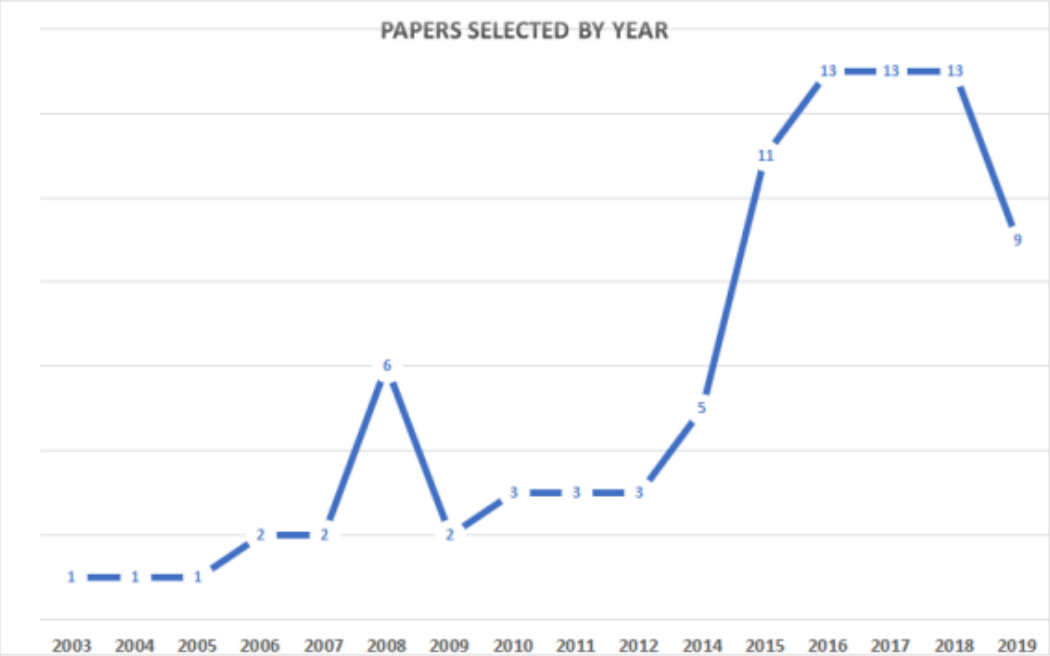


Figure 2. Papers selected by year.

Table 3. Papers selected by year.

Year	Selected by Title and Abstract	Selected after Complete Reading	Selected after Quality Questionnaire
1993	0	0	0
1994	0	0	0
1995	0	0	0
1996	0	0	0
1997	0	0	0
1998	0	0	0
1999	0	0	0
2000	0	0	0
2001	0	0	0
2002	0	0	0
2003	2	1	1
2004	1	1	1
2005	2	2	1
2006	5	4	2
2007	5	3	2
2008	11	6	6
2009	5	3	2
2010	7	4	3
2011	4	4	3
2012	6	4	3
2013	5	1	0
2014	14	5	5
2015	20	13	11
2016	25	14	13
2017	26	16	13
2018	20	14	13
2019	31	11	9
Total	**189**	**106**	**88**

3.8. Information Retrieval

The data retrieval form presented in Appendix B gives guidelines for the retrieval of data from all studies covered in this systematic literature review. It also includes details of the main study itself and the information needed to target the research questions. The entire paper has been read to collect the necessary data and the specific information has been extracted from each document: source, authors, title, year of publication and responses to the research questions.

4. Results

This study aims to investigate the available literature according to the research questions mentioned in Table 1. 29 (32.95%) of the 88 studies included in our systematic review of the literature were published in journals specializing in computer and network security, data science, etc., while 59 (67.05%) were published in leading conferences and workshops in the same or similar areas. Table 4 lists the total of studies selected by year, grouped by conferences and journals, while Figure 3 summarizes the percentage of studies published in journals and conferences.

Table 4. Studies selected by year, grouped by conferences and journals.

Year	Conferences	Journals	Total Studies
2003	1	0	1
2004	0	1	1
2005	0	1	1
2006	2	0	2
2007	2	0	2
2008	5	1	6
2009	2	0	2
2010	3	0	3
2011	1	2	3
2012	3	0	3
2014	4	1	5
2015	7	4	11
2016	10	3	13
2017	11	2	13
2018	8	5	13
2019	0	9	9
Total	**59**	**29**	**88**

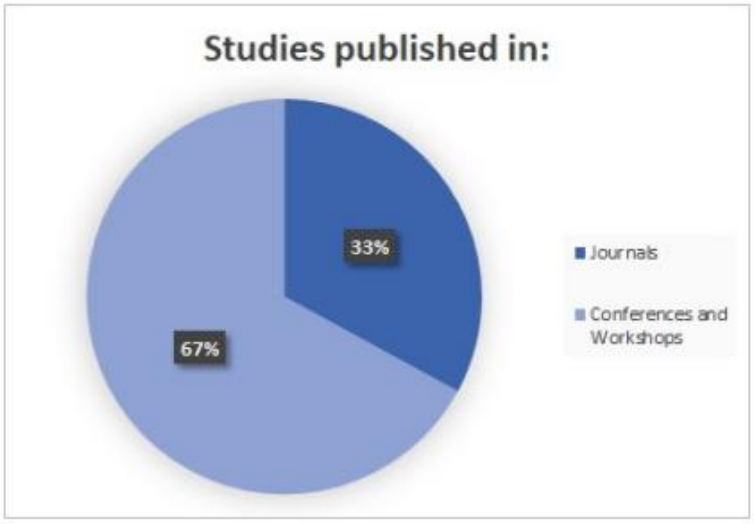


Figure 3. % of studies published in journals and conferences.

4.1. Specific Attack Detection/Prevention

In our research, we have detected that some of the studies reviewed are focused on protecting web servers against specific kinds of attacks, mainly on DDoS and Injection Attacks. A list of the most studied types of attacks is presented below; in addition, Table 5, details the specific attacks, the number of studies dealing with each particular attack, as well as a list of the corresponding citations.

Table 5. Detail of attacks.

Attack	Number of Studies	Citations
DDoS	11	[32–42]
Injection	10	[43–52]
Botnets	2	[53,54]
Defacement	2	[55,56]
Other Attacks	62	[57–118]

4.1.1. DDoS Attacks

Denial of Service (DoS) attacks are a form of attack that seeks to make a network resource unavailable due to overloading the resource or machine with an overwhelming number of packets, thereby crashing or severely slowing the performance of the resource. Distributed Denial of Service (DDoS) is a large scale DoS attack which is distributed in the Internet. In a first phase, the attacker proceeds to identify and exploit vulnerabilities in one or more networks for the installation of malware programs in multiple computers in order to obtain control of them remotely. At a later stage, these compromised computers are exploited for the mass sending of attack packets to the target(s) that will usually be located outside the original network of infected computers. These attacks occur without the knowledge of the compromised hosts [119].

Thang and Nguyen [32] proposed a framework to detect DDoS attacks; this framework was based on using an on-line scanning process to detect certain traits of DDoS attack and building a dynamic blacklist. Tripathi and Hubballi [33] proposed the use of chi-square test in order to detect slow rate denial of service attacks against HTTP/2 protocol. In [34], Najafabadi et al. proposed a detection method for the application layer DDoS attacks that worked extracting instances of user behaviors requesting resources from HTTP web server logs and using Principal Component Analysis (PCA) in order to detect anomalous behavior instances. Zolotukhin and Kokkonen [35] focused on detection of application-layer DoS attacks that utilize encrypted protocols by applying an anomaly-detection-based approach to statistics extracted from network packets headers using the stacked autoencoder algorithm. Shirani, Azgomi and Alrabaee [36] proposed the detection of DDoS attacks on Web Services using time series and applying the ARIMA model. Tripathi, Hubballi and Singh [37] used Hellinger distance between two probability distributions generated in training and testing phases to detect Slow HTTP DoS attacks. Wang et al. [38] proposed a sketch-based anomaly detection scheme for application layer DDoS attacks. The scheme utilizes the divergence of sketches in two consecutive detection cycles to detect the occurrence of an anomaly, designing a variant of Hellinger Distance to measure the divergence for mitigating the impact of network dynamics. Wang et al. [39] proposed multifeatures information entropy prediction model in order to prevent flooding App-DDoS attacks; for asymmetric attacks, a second-order Markov detection model was proposed. Xie and Tang [40] proposed a Web user browsing behavior model to detect DDoS attacks based on Hidden Markov Model. Markov states represent the click-behavior of users while hyperlink among pages is represented by different states. Lin et al. [41] proposed a new statistical model to detect DDoS attacks called Rhythm Matrix (RM), based on the packet size and the interarrival time of consecutive HTTP-request packets in a flow that indicated the users' behaviour when opening and browsing web pages. RM characterized the user access trajectory fragments distribution, including the order of visiting pages and the time spent on each page. Change-rate abnormality in the RM was used to detect DDoS attacks, and further identify the malicious hosts according to their drop points in the RM.

4.1.2. Injection Attacks

Injection flaws allow attackers to relay malicious code through an application to another system. These attacks include calls to the operating system via system calls, the use of external programs via shell commands, as well as calls to backend databases via SQL (i.e., SQL injection) [120]. SQL Injection (SQLI) constitutes an important class of attacks against web applications. By leveraging insufficient input validation, an attacker could obtain direct access to the database underlying an application [121].

Kozik, Choraś and Holubowicz [43] used token extraction of HTTP request, as well as an evolutionary-based token alignment unsupervised algorithm to detect SQLI and Cross Site Scripting (XSS) attacks. Wang et al. [44] proposed a new algorithm called FCERMining (Frequent Closed Episode Rules Mining) for mining frequently closed episode rules, dealing with big data on Spark to find the valid rules quickly. They made some experiments with the SQLMAP map tool in order to test the proposed method against SQLI attacks. Yuan et al. [45] presented a comprehensive three-step approach aimed at detecting and preventing SQLI attacks: Firstly, an ensemble clustering model is applied to separate anomalies from normal samples. In the second phase, word2vec algorithm is used to get the semantical presentations of anomalies. Finally, another multi-clustering approach clusters anomalies into specific types. Kozik, Choraś and Holubowicz [49] proposed an algorithm for SQL injection attack detection using a modified Linear Discriminant Analysis (LDA), including dimensionality reduction using Singular Value Decomposition (SVD), and an adaptation of Simulated Annealing for LDA projection vector computation.

4.1.3. Botnets Attacks

A bot is a compromised computer that can carry out the commands of its master, and bots are networked to form a botnet with a topology chosen by their master [122]. Differences botnet than other types of attacks is the existence of Command and Control (C&C) that work in giving orders from botmaster to bot. Bots always hide while looking for an unattended target, when bot find the target they will report to the botmaster [123].

Yu, Guo and Stojmenovic [53] established a four-parameter semi-Markov model to represent browsing behavior. Based on this model, they found that it was impossible to detect mimicking attacks based on statistics if the number of active bots of the attacking botnet is sufficiently large (though it is hard for botnet owners to satisfy the condition to carry out a mimicking attack most of the time). They concluded that mimicking attacks could be discriminated from genuine flash crowds using second order statistical metrics, defining a new correntropy metric. Sakib and Huang [54] proposed the detection of HTTP-based C&C traffic using statistical features based on client generated HTTP request packets and DNS server generated response packets. They applied three different anomaly detection methods: Chebyshev's Inequality, One-class Support Vector Machines (OCSVM) and Nearest Neighbor based Local Outlier Factor.

4.1.4. Defacement

In the web defacement attack the invader changes the visual appearance of the webpage. The business competitor, insurgent and extremist groups defame the reputation of the organizations and mislead public through these types of attacks, modifying the content of home page. Web defacement can be broadly categorized into Text Defacement and Image Defacement. [124].

Davanzo, Medvet and Bartoli [56] proposed a test framework for a web defacement monitoring service, working with different algorithms aimed at producing an item's (a document downloaded from a specific URL) binary classification. The algorithms evaluated were: kth nearest, Local Outlier Factor, Hotelling's T-Square, Parzen windows, Support Vector Machines and Domain Knowledge aggregator. Best results were obtained with Domain Knowledge, Support vector Machines, Parzen windows and Hotelling's T-Square. Medvet and Bartoli [55] considered the problem of corruption in

the learning data, concerning a Web Site Defacement detection system, presenting a procedure for detecting whether a learning set is corrupted.

4.1.5. Other Attacks

This group includes all those studies in which the type of attack studied is not clearly specified, either because it makes use of non-publicly accessible datasets and does not provide information on the type of attack it is trying to detect, or because it does not try to detect a specific type of attack but any web request which is considered anomalous, etc.

4.2. Current Status of Anomaly Detection

An anomaly detection process implies the use of different strategies and different techniques to achieve the final objective: clustering algorithms, classification, dimensionality reduction, use of auxiliary techniques, etc. The main algorithms and techniques detected in the different studies analyzed are detailed below.

4.2.1. Clustering Algorithms

Clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups (clusters) [125]. Depending on the comparison of the new data received against the model generated by the clustering algorithm, it is determined whether it is an anomaly point (the way to do this varies depending on the type of clustering algorithm used, and can be based on distance or probabilities). The clustering algorithms most used are:

- **K-Means:** K-means is an unsupervised classification (clustering) algorithm that groups objects into k groups based on their characteristics. Clustering is done by minimizing the sum of distances between each object and the centroid of its group or cluster. The quadratic distance is usually used. The k-means algorithm solves an optimization problem. The function to optimize (minimize) is the sum of the quadratic distances of each object to the centroid of its cluster. [126]

  The objects are represented with real vectors of n dimensions $(x_1, x_2, \ldots, x_n)$ and the k-means algorithm constructs k groups where the sum of distances of the objects is minimized, within each group $S = \{S_1, S_2, \ldots, S_k\}$, to its centroid. The problem can be formulated as follows [126]:

$$\min_{s} E(\mu_i) = \min_{s} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \quad (1)$$

  where S is the set of data whose elements are the objects x_j represented by vectors, where each of its elements represent a characteristic or attribute. We will have k groups or clusters with their corresponding centroid μ_i [126].

  In each update of the centroids, from the mathematical point of view, we impose the necessary end condition on the function $E(\mu_i)$ which, for the quadratic function (1) is:

$$\frac{\partial E}{\partial \mu_i} = 0 \Rightarrow \mu_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

  and the average of the elements of each group is taken as a new centroid.
- **Gaussian Mixture Model:** Gaussian mixture models are a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models, in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Since subpopulation assignment is not known, this constitutes

a form of unsupervised learning. The Gaussian Mixture function is formed by several Gaussians, individually identified by $k \in \{1, \ldots, K\}$, where K is the number of groups formed in the data set. Each Gaussian k is composed of mean μ (defines its center), Σ covariance (defines its width), a mixture probability π (defines the size of the Gaussian function). The mixing coefficients are probabilistic and must meet this condition:

$$\sum_{k=1}^{K} \pi_k = 1$$

In general, the Gaussian density function is defined by:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

where x represents the data points, D is the number of dimensions of each data point. μ is the mean and Σ is the covariance.

- **Mahalanobis Distance:** The Mahalanobis Distance is a multivariate distance metric that measures the distance between a point (vector) and a distribution. The most common use for Mahalanobis Distance is to find multivariate outliers, which indicates unusual combinations of two or more variables. The formal definition is:

$$D^2 = (x-m)^T \cdot C^{-1} \cdot (x-m)$$

where D^2 is the square of the Mahalanobis Distance, x is the vector of observations, m is the vector of mean values of independents variables, C^{-1} is the inverse covariance matrix of independent variables.

- **Affinity Propagation:** Affinity Propagation does not require the number of clusters to be determined before running the algorithm. The data points can be seen as a network where all the data points send messages to all other points [127]. The subject of the messages is the determination of the points being an exemplar. The exemplars are points that explain the other data points "better" and are the most significant of their aggregation. All data points want to determine collectively the data points that are an exemplar to them. These messages are saved in two matrices:

  - *Responsibility Matrix R*. In this matrix, $r(i,k)$ reflects how well point k is adjusted to be an exemplar for point i.
  - *Availability Matrix A*. $a(i,k)$ reflects how accurate it would be for point i to select point k as an exemplar.

Let $(x_1, x_2, \ldots, x_n)$ be a set of data points, with no internal structure assumptions, and let s be a function that measures the degree of similarity between any two points, such that $s(x_i, x_j) > s(x_i, x_k) \iff x_i$ is more similar to x_j than to x_k.

Similarity matrix(S) gives us information about the similarity between two data points:

$$s(i,k) = -\|x_i - x_k\|^2$$

that is defined as the negative of the euclidean distance between the two instances. The greater the distance between any two instances, smaller is the similarity between them. The diagonal of s $s(i,i)$ represents the input preference, i.e., the probability that a given input will become an exemplar. When the same value is set for all entries, it controls how many classes the algorithm produces. A value close to the lowest possible similarity produces fewer classes, while a value close to or

greater than the highest possible similarity produces many classes. It is usually initialized with the median similarity of all pairs of entries.

The algorithm proceeds by alternating two message passing stages, to update the Responsibility Matrix and the Availability Matrix. Both matrices are initialized with zeros and can be viewed as log probability tables. These updates are performed on an iterative basis:

First, Responsibility updates are sent:

$$r(i,k) \leftarrow s(i,k) - \max_{k's.t.k' \neq k} \{a(i,k') + s(i,k')\}$$

Then, availability is updated per:

$$a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k))\right)$$

for $i \neq k$ and

$$a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k))$$

The iterations are performed until either the cluster boundaries remain unchanged over a number of iterations, or after some predetermined number of iterations. The exemplars whose sum of *responsibility and availability* is positive are obtained: $(r(i,i) + a(i,i)) > 0$

- **DBSCAN:** Density-based spatial clustering of applications with noise (DBSCAN) [128] is a density-based clustering algorithm because it finds a number of groups (clusters) starting with a given density distribution of the corresponding nodes. Clustering happens based on two parameters: *Neighbourhood*, cutoff distance of a point from core point for it to be considered a part of a cluster. Commonly referred to as ε. *Minimum points*, minimum number of points required to form a cluster. Commonly referred to as minPts.

  There are three types of points after the DBSCAN clustering is complete: *Core*, this is a point which has at least minPts points within distance ε from itself. *Border*, this is a point which has at least one Core point at a distance ε. *Noise*, this is a point which is neither a *Core* nor a *Border*. It has less than minPts points within distance ε from itself.

  DBSCAN can be summarized in following steps: The algorithm begins with an arbitrary point that has not been visited. The neighborhood of this point is limited, and if it contains specific points, a cluster starts on it. Otherwise, the point is labeled as noise. Note that the point in question may belong to another neighborhood than the specific one in the corresponding cluster. If a point is included in the dense part of a cluster, its neighborhood is also part of the cluster. Thus, all points in that neighborhood are added to the cluster, as are the neighborhoods of these points that are sufficiently dense. This process continues until a densely connected cluster is completely built. Then, a new point not visited is visited and processed in order to discover another cluster or noise.
- **Nearest Neighbor based Local Outlier Factor:** The Local Outlier Factor (LOF) [129] is based on the concept of a local density, where the locality is given by the k-nearest neighbours. The density is estimated by the distance between close neighbours. If an object's density is compared with the densities of its neighbours, regions with similar density values and points with density values far below the values obtained by its neighbours will be identified. These points are considered outliers. The steps to calculate the LOF are detailed below:

- Calculate distance between the pair of observations.
- Find the kth nearest neighbor observation; calculate the distance between the observation and k-Nearest neighbor.
- Calculate the reachability distance between object p and o:

$$reach\text{-}dist_k(p,o) = max\{k\text{-}distance(o), d(p,o)\}$$

- Calculate Local Reachability Density (LRD): the most optimal distance in any direction from the neighbor to the individual point. The local reachability density of an object p is the inverse of the average reachability distance based on the MinPts (minimum number of objects) nearest neighbors of p.

$$LRD_{MinPts}(x) = \frac{1}{\left(\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p,o)}{|N_{MinPts}(p)|}\right)}$$

- Calculate Local Outlier Factor: It is the average of the ratio of the local reachability density of p and those of $p'sMinPts\text{-}nearestneighbors$; captures the degree to which we call p an outlier.

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

- **Expectation–Maximization:** The Expectation–Maximization (EM) algorithm is a way to find maximum likelihood estimates for model parameters when the data is incomplete, has missing data points, or has unobserved (hidden) latent variables. It is an iterative way to approximate the maximum likelihood function [130,131]. The basics steps for the algorithm are:

  - An initial guess is made for the model's parameters and a probability distribution is created (E-step).
  - Until stability is reached, do:
    - Newly observed data is added to the model.
    - The probability distribution from the E-step is tweaked to include the new data (M-step).

  Formally:

  Given the statistical model which generates a set X of observed data, a set of unobserved latent data or missing values Z, and a vector of unknown parameters θ, along with a likelihood function $L(\theta; X, Z) = p(X, Z|\theta)$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by maximizing the marginal likelihood of the observed data. [132]

$$L(\theta; X) = p(X|\theta) = \int p(X, Z\theta) dZ$$

  The EM algorithm finds the MLE by iteratively calculating the [132]:

  - Expectation Step (E-Step): Define $Q(\theta|\theta^{(t)})$ as the expected value of the log likelihood function of θ, with respect to the current conditional distribution of Z given X and the current estimates of the parameters $\theta^{(t)}$

$$Q(\theta|\theta^{(t)}) = E_{Z|X,\theta^{(t)}}[logL(\theta; X, Z)]$$

- Maximization step (M step): Find the parameters that maximize:

$$\theta^{(t+1)} = arg_\theta \max Q(\theta|\theta^{(t)})$$

4.2.2. Classification Algorithms

The idea behind the classification algorithms is very simple: it is about predicting the target class by analyzing the training dataset. The training dataset is used to get better boundary conditions which could help to determine each target class. When boundary conditions are determined, target class can be predicted. The classification of new data as anomalous or not, depends on the class in which it is classified.

All classification algorithms can be generalized as algorithms that receive a training set and learn a classification function of the form $f : R^n \rightarrow \{+1, -1\}$. This function is applied to news inputs and its value represents the class to which the input is classified [133].

The classification algorithms most used are:

- **One Class Support Vector Machine:** The problem addressed by One Class Support Vector Machine (OCSVM) is novelty detection [134]. The idea of novelty detection is to detect rare events, i.e., events that happen rarely, and hence, with very little samples. The problem is then, that the usual way of training a classifier will not work. Here the idea is to find a function that is positive for regions with high density of points, and negative for small densities.

  Consider a dataset:
  $\Omega = \{(x_1, y_2), (x_2, y_2), \ldots, (x_n, y_n)\}$; $x_i \in R^d$ in a multi-dimensional space where x_i is the i –th input data point and $y_i \in \{-1, 1\}$ is the i –th output pattern, indicating the class membership.

  SVMs can create a non-linear decision boundary by projecting the data through a non-linear function ϕ to a space with a higher dimension. This means that data points which can't be separated by a straight line in their original space I are lifted to a feature space F where there can be a straight hyperplane that separates the data points of one class from another. When that hyperplane would be projected back to the input space I, it would have the form of a non-linear curve.

  OCSVM separates all the data points from the origin (in feature space F) and maximizes the distance from this hyperplane to the origin. This results in a binary function which captures regions in the input space where the probability density of the data lives.
- **Hidden Markov Model:** A Hidden Markov Model (HMM) is a statistical model in which the system to be modeled is assumed to be a Markov process of unknown parameters. The objective is to determine the hidden parameters of that string from the observable parameters. In a normal Markov model, the state is directly visible to the observer, so the transition probabilities between states are the only parameters. In a hidden Markov model, the state is not directly visible, but only the variables influenced by the state are visible. Each state has a probability distribution over the possible output symbols. Consequently, the symbol sequence generated by an HMM provides some information about the state sequence [135].

  Formally, a HMM is a quintuple (S, V, π, A, B), characterized by the following elements [136]:

  - $S = \{S_1, S_2, \cdots, S_N\}$ is the set of states, where N is the number of states. The triplet (S, π, A) represents a Markov chain; the states are hidden and never observable directly.
  - $V = \{V_1, V_2, \cdots, V_M\}$ is the discrete set of possible symbol observations, where M represents the number of observations.

- $\pi : S \rightarrow [0,1] = \{\pi_1, \pi_2, \cdots, \pi_N\}$ is the initial probability distribution on states. It gives the probability of starting in each state. It can be expected that:

$$\sum_{s \in S} \pi(s) = \sum_{i=1}^{N} \pi_1 = 1$$

- $A = (a_{ij})_{i \in S, j \in S}$ is the transition probability of moving from state S_i to state S_j. It can be expected that $a_{ij} \in [0,1]$ for each S_i and S_j, and that $\sum i \in S^{a_{ij}} = 1$ for each S_j.
- $B = (b_{ij})_{i \in V, j \in S}$ is the emission probability that symbol v_i is seen in state S_i.

The model makes two assumptions:

- *The Markov assumption:* represents the memory of the model, so current state depends only on the previous state. Formally:

$$P(q_t | q_1^{t-1}) = P(q_t | q_{t-1})$$

- *The independence assumption:* the output observation at time t is dependent only on the current state and it is independent of previous observations and states. Formally:

$$P(o_t | o_1^{t-1}, q_1^t) = P(o_t | q_t)$$

- **K-Nearest Neighbors:** The K-Nearest Neighbors (KNN) algorithm classifies new objects according to the outcome of the closest object or the outcomes of several closest objects in the feature space of the training set [137]. An object is classified by a majority vote of its neighbors, with the new object being assigned to the class that is most common among its k nearest neighbors (k is a positive integer, and typically small). The neighbors are taken from a set of objects for which the correct classification is known. In the classification phase, k is a user-defined constant, and a new object with given features is classified by assigning to it the label that is most frequent among the k training samples nearest to that new object. With continuous features, Euclidean distance is used as distance metric, while with categorical features the Hamming distance is used. Finally, the input x gets assigned to the class with the largest probability.
- **Naive Bayes:** The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors [138–140]. Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifier assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors, i.e., *class conditional independence*. Formally:

$$P(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)}$$

where:

- $P(C_k|x)$ is the posterior probability of class given predictor.
- $p(C_k)$ is the prior probability of class.
- $p(x|C_k)$ is the likelihood which is the probability of predictor given class.
- $p(x)$ is the prior probability of predictor.

As the denominator does not depend on C and the values of the features x_i are given, the denominator is constant. The numerator is equivalent to the joint probability model $p(C_k, x_1, x_2, \ldots, x_n)$ and, using the chain rule:

$$\begin{aligned}
&p(C_k, x_1, x_2, \cdots, x_n) = \\
&= p(x_1, x_2, \cdots, x_n, C_k) \\
&= p(x_1|x_2, \cdots, x_n, C_k)p(x_2, \cdots, x_n, C_k) \\
&= p(x_1|x_2, \cdots, x_n, C_k)p(x_2|x_3, \cdots, x_n, C_k) \\
&p(x_3, \cdots, x_n, C_k) \\
&= \cdots \\
&= p(x_1|x_2, \cdots, x_n, C_k)p(x_2|x_3, \cdots, x_n, C_k) \cdots \\
&p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k)
\end{aligned}$$

Supposing that all features in x are independent of each other, depending on the category C_k, then:

$$p(x_i|x_{i+1}, \cdots, x_n, C_k) = p(x_i|C_k)$$

Thus, the joint model can be expressed as:

$$\begin{aligned}
&p(C_k|x_1, \cdots, x_n) \propto p(C_k, x_1, \cdots, x_n) \\
&= p(C_k)p(x_1|C_k)p(x_2|C_k)p(x_3|C_k) \cdots \\
&= p(C_k)\prod_{i=1}^{n} p(x_1|C_k)
\end{aligned}$$

where $\propto$ denotes proportionality.

4.2.3. Neural Network

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes [141]. The connections between neurons are modeled as weights. A positive weight reflects an excitatory connection, while a negative weight results in an inhibitory connection. There is a *linear combination* that modifies all inputs applying the corresponding weights and proceeding to the sum of modified inputs. There is also, an activation function that controls the amplitude of the output. When the neural network receives a new anomalous data, it will have difficulty in processing it, as it is trained to process normal data, so it will generate a high mean square error (MSE).

- **Stacked Auto-encoder:** Autoencoder is a kind of unsupervised learning structure that owns three layers: input layer, hidden layer, and output layer. The structure of a stacked auto-encoder consists of several hidden layers of auto-encoders in a neural network. The output of each hidden layer is connected to the input of the next layer. The hidden layers are trained by an unsupervised algorithm and then tuned by a supervised method. Stacked autoencoder mainly consists of three steps [142]:
  - Train the first autoencoder by input data and obtain the learned feature vector.
  - The feature vector of the former layer is used as the input for the next layer, and this procedure is repeated until the training completes.
  - After all the hidden layers are trained, backpropagation algorithm (BP) [143] is used to minimize the cost function and update the weights with labeled training set to achieve fine-tuning.

- **Word2vec:** Word2vec is a two-layer neural net that processes text by transforming words into vectors. Its input is a text corpus and its output is a set of vectors: feature vectors that represent words in that corpus. Word2vec tries to group vectors of similar words together in vectorspace [144,145]. There are two different architectures capable of representing words in a distributed way: *continuous bag of words (CBOW)* and *continuous skip-gram*. Word2vec is capable of using any of them. The word prediction is done in a different way depending on the selected architecture: in the CBOW architecture, the prediction is done based on a window of context words, without being influenced by the order of those contextual words. In the skip-gram architecture, the surrounding context words are predicted from the current word, with the nearby words having more weight in the context than those that are distant.

4.2.4. Feature Selection and Extraction

Features are the specific variables that are used as input to an algorithm. Features can be selections of raw values from input data or can be values derived from that data. Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps. In feature selection and extraction models, an anomaly is defined in terms of the redundancy present in the model, i.e., the semantic information is modelled in multiple ways. The most commonly used algorithms and techniques for selecting and extracting features are listed below.

- **N-Grams:** This is a consecutive sequence of n elements that constitute a text sample; based on a probabilistic language model, the next element in the sequence is predicted in the form of an $(n-1)$ order *Markov Model*. An N-gram model predicts x_i based on $x_{i-(n-1)}, \cdots, x_{i-1}$, i.e.,: $P(x_i|x_{i-(n-1)}, \cdots, x_{i-1})$. Formally [146,147]: Given a sequence of tokens $S = (s_1, s_2, \cdots, s_{N+(n-1)})$ over the token alphabet $\mathcal{A}$, where N and n are positive integers, an *n-gram* of the sequence S is any *n-long* subsequence of consecutive tokens. The *ithn-gram* of S is the sequence $s_i, s_{i+1}, \cdots, s_{i+n-1}$.
- **Bag Of Words:** Bag Of Words (BOW) algorithm encodes words of the text (that represent categorical features) into real-valued vectors, making a list of unique words in the text corpus called *vocabulary*. Each sentence or document can be represented as a vector with a value of 1 if the word is present in the vocabulary, or 0 otherwise. Another representation can be done by counting the number of times each word appears in the document, using the *Term Frequency-Inverse Document Frequency (TF-IDF)* technique [148,149].

  - Term Frequency (TF): $TF = TD/ND$ where TD is the number of times term t appears in a document and ND is the number of terms in the document.
  - Inverse Document Frequency (IDF): $IDF = log(N/n)$,
    where N is the number of documents and n is the number of documents a term t has appeared in. The IDF of a rare word is high, whereas the IDF of a frequent word is likely to be low.
  - Term Frequency-Inverse Document Frequency (TF-IDF): $TF\text{-}IDF = TF \cdot IDF$

4.2.5. Attribute Character Distribution

The attribute character distribution model captures the concept of a "normal" or "regular" query parameter by looking at its character distribution. The approach is based on the observation that attributes have a regular structure, are mostly human-readable, and almost always contain only printable characters. In case of attacks that send binary data, a completely different character distribution can be observed. Characters in regulars attributes are drawn using the corresponding ASCII table values [98].

- **Idealized Character Distribution:** In [98] approach, the Idealized Character Distribution (ICD) is obtained during the training phase from normal requests sent to web application. The IDC is calculated as the mean value of all character distributions. During the detection phase, the probability that the character distribution of a sequence is an actual sample drawn from its ICD is evaluated. For that purpose Chi-Square metric is used. Let $D_{chisq}(Q)$ be the Chi-Square metric for a sequence Q where N indicates the length of Q, ICD the distribution established for all the samples, σ the standard deviation from the ICD, and h the distribution of the sequence that is being tested Q, then the value of $D_{chisq}(Q)$ is computed as:

$$D_{chisq}(Q) = \sum_{n=0}^{N} \frac{1}{\sigma^2}[ICD_n - h(Q_n)]^2$$

  In [87], Kozik et al. group the characters for which the decimal value in ASCII table belongs to the followings ranges: $< 0,31 >$, $< 32,47 >$, $< 48,57 >$, $< 58,64 >$, $< 65,90 >$, $< 91,96 >$, $< 97,122 >$, $< 123,127 >$, $< 128,255 >$.

4.2.6. Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables [150]. In this case, an anomaly will be detected depending on the distance between a new data and the standard deviation of the training data set.

- **Principal Component Analysis:** This is the most commonly used technique of dimensionality reduction; it works by linearly reducing the existing data to a lower dimensionality space, trying to preserve the maximum variance of the data in the lower dimensionality space. [151,152]. Principal Component Analysis (PCA) is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on [153]. The process of obtaining PCA from a given dataset can be summarized as:

  - Take a whole dataset of $d + 1$ dimension ignoring the label field, so the dataset becomes d *dimensional*.
  - Compute the mean of every dimension of the whole d *dimension* dataset and represent it in a matrix A.
  - Compute the covariance matrix of A. The result would be a square matrix of $d \times d$ *dimensions*.
  - Compute *Eigenvectors* and corresponding *Eigenvalues*.
  - Sort the *Eigenvectors* by decreasing *Eigenvalues* and choose k *Eigenvectors* with the largest *Eigenvalues* to form a $d \times k$ *dimensional matrix* W.
  - Transform the samples onto the new subspace.

- **Linear Discriminant Analysis:** Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant to find a linear combination of features that characterizes or separates two or more classes of objects or events. LDA attempts to express one dependent variable as a linear combination of other features or measurements [154–156]. The goal of an LDA is to project a n *dimensional* feature space onto a smaller subspace k where $k \leq n - 1$. The process of obtaining LDA from a given dataset can be summarized as:

  - Compute the d-*dimensional* mean vectors for the different classes from the dataset.
  - Compute the scatter matrices.

- Compute the eigenvectors $(e_1, e_2, \cdots, e_d)$ and corresponding eigenvalues $(\lambda_1, \lambda_2, \cdots, \lambda_d)$ for the scatter matrices.
- Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix W, where every column represents an eigenvector.
- Use W matrix to transform the samples onto the new subspace.

- **Diffusion Map:** Unlike others popular dimensionality reduction techniques like PCA and LDA, Diffusion Maps are non-linear and focus on discovering the underlying manifold, i.e.,: lower-dimensional constrained "surface" upon which the data is embedded [157–159]. It achieves dimensionality reduction by re-organising data according to parameters of its underlying geometry. A diffusion map embeds data in (transforms data to) a lower-dimensional space, such that the Euclidean distance between points approximates the diffusion distance in the original feature space. The dimension of the diffusion space is determined by the geometric structure underlying the data, and the accuracy by which the diffusion distance is approximated.

4.2.7. Statistical Techniques and Probability Distribution

In addition to the algorithms and techniques described above, there are a number of statistical techniques commonly used in the studies reviewed. These techniques are listed below:

- **Chebyshev's Inequality.** Let K is any positive real number greater than 1. Chebyshev's Inequality says that *at least* $1 - \frac{1}{K^2}$ of data from a sample must fall within K standard deviations from the mean [160]. In a normal distribution, 68% of the data is one standard deviation from the mean, 95% is two standard deviations from the mean, and approximately 99% is three standard deviations from the mean. If data set is not normally distributed then the use of Chebyshev's Inequality provides a way to, knowing only the mean and standard deviation of the sample, estimate the worst scenario in which the data is distributed: i.e.,: for *any distribution, at least* 75% of the data must be between two standard deviations from the mean. For example:
  - If $\sigma = 2$, then $1 - (1/2^2) = 3/4 = 75\%$ of the data values of any distribution must be within two standard deviations of the mean.
  - If $\sigma = 3$, then $1 - (1/3^2) = 8/9 = 89\%$ of the data values of any distribution must be within three standard deviations of the mean.
  - If $\sigma = 4$, then $1 - (1/4^2) = 15/16 = 93.75\%$ of the data values of any distribution must be within four standard deviations of the mean.

- **Pearson's Chi Square Test.** Pearson's chi-squared test χ^2 is a statistical test applied to sets of categorical data to evaluate how likely it is that any observed difference between the sets arose by chance. The chi-square test belongs to the so-called goodness-of-fit or contrast tests, which aim at deciding whether the hypothesis that a given sample comes from a population with a probability distribution fully specified in the null hypothesis can be accepted, allowing the comparison of the goodness of fit, the independence of the variables and the level of homogeneity of a distribution. The comparisons are based on the comparison of observed frequencies (empirical frequencies) in the sample with those that would be expected (theoretical or expected frequencies) if the null hypothesis were true. Thus, the null hypothesis is rejected if there is a significant difference between the observed and expected frequencies. [161]. To calculate the statistic, the procedure is as follows:
  - Calculate the chi-squared test statistic.
  - Determine the degrees of freedom *(df)* of that statistic.

- Select a desired level of confidence.
- Compare χ^2 to the critical value from the chi-squared distribution with df degrees of freedom and the selected confidence level.
- The difference between the observed and expected frequencies of a distribution is evaluated using the χ^2 statistic. If the difference between observed and expected frequencies is large, the null hypothesis H_0 is false and may be rejected, i.e., this distribution does not fit the theoretical distribution. The alternative hypothesis H_1 can be accepted.

- **Bayesian Probability.** Bayesian Probability theory provides a mathematical framework for performing inference, or reasoning, using probability. It is most often used to judge the relative validity of hypotheses in the face of noisy, sparse, or uncertain data, or to adjust the parameters of a specific model [162]. The combined probability of two events, A and B, can be expressed as $P(AB) = P(A|B)P(B) = P(B|A)P(B)$. Assuming that one of the events is the hypothesis H and the other is data D, it is possible to judge the relative certainty of the hypothesis in relation to the data: According to Bayes' rule:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}$$

Another way of interpreting Bayes' rule is by taking into account the acquired learning. That is, the transformation from $P(H)$ to $P(H|D)$ reflects the level of learning about the validity of the hypothesis from the data.

On Table 6, the different clustering algorithms found in the studies analyzed are listed. Table 7 lists the main classification algorithms. Finally, in Table 8, we detail the main auxiliary techniques used on different papers analyzed.

Table 6. Detail of clustering algorithms used.

Algorithm	Number of Studies	Citations
K-Means	7	[35,60,67,73,74,88,91]
Gaussian Mixture Model	4	[45,93,94,163]
Mahalanobis Distance	4	[82,93,102,113]
Affinity Propagation	3	[58,91,103]
DBSCAN	2	[32,35]
Convolutional Autoencoder	2	[71,115]
Nearest Neighbor based Local Outlier Factor	2	[54,56]
Expectation Maximization	2	[93,163]
Single-linkage clustering algorithm	1	[35]
Fuzzy C-Means	1	[35]
Self-Organizing Maps (SOM)	1	[35]
Deep Learning Enabled Subspace Spectral Ensemble Clustering (DEP-SSEC)	1	[45]
Subspace Spectral Ensemble Clustering (SSEC)	1	[45]
Min-Max-Tree	1	[69]
Graph-based segmentation	1	[72]
Outlier Gaussian Mixture	1	[79]
Birch	1	[88]
Mean shift	1	[88]
Subspace Weighted Ensemble Clustering	1	[94]
Query based projected clustering	1	[106]
Infinite Bounded Generalized Gaussian mixture model (InBGG)	1	[116]
Session Feature Similarity (SFAD)	1	[42]
Lambda Cut	1	[42]

Table 7. Main classification algorithms used.

Algorithm	Number of Studies	Citations
Markov Models [1]	15	[39,40,48,53,63,66,84,88,98,104,105,107,109, 111,116]
One-Class Support Vector Machine (OCSVM)	11	[45,54,56,63,64,77,81,90,94,103,114]
Decision Tree [2]	9	[43,47,65,66,70,85,91,101,103]
K-Nearest Neighbors (K-NN)	5	[56,83,91,103,112]
Neural Network [3]	2	[86,89]
Random Forest	2	[75,117]
Isolation Forest	2	[59,114]
Needleman-Wunsch Algorithm	2	[72,74]

[1] Markov Models include: Hidden Markov Model, Hidden Semi - Markov Model, Semi-Markov Model, Continuous Time Markov Chains, Markov Random Field, Event—driven Hidden Semi Markov Model, Markov Chain Monte Carlo, Second Order Markov. [2] Decision Tree Models include: Reduced Error Pruning Tree (REPTree), Decision Stump, Information Gain, C4.5 decision tree, XGBoost, SPAM tree, Modified DPS Pruning. [3] Neural Network Models include: Extreme Learning Machine, C-LSTM, Convolutional Neural Network, Recurrent Neural Network, Long Short Term Memory, Deep Neural Network, Softmax Classifier.

Table 8. Detail of auxiliary techniques used.

Technique	Type	Number of Studies	Citations
N-Grams	Feature Selection	12	[57,59–61,69,77,82,90,93, 102,113,118]
Bag-Of-Words (BOW)	Feature Selection	2	[48,115]
Filter Based subset Evaluation (FBSE)	Feature Selection	1	[75]
Uniformed Conditional Dynamic Mutual Information (UCDMIFS)	Feature Selection	1	[78]
Multifeatures Information Entropy Prediction Model	Feature Selection	1	[39]
Genetic Algorithm (GA)	Optimization & Searching	4	[43,81,85,97]
Principal Component Analysis (PCA)	Dimensionality Reduction	6	[34,57,60,91,113,118]
Random Projection (RP)	Dimensionality Reduction	1	[57]
Diffusion Map (DM)	Dimensionality Reduction	1	[57]
Sample Entropy	Dimensionality Reduction	1	[60]

4.3. Datasets

One of the biggest problems in conducting experiments to evaluate web anomaly detection techniques is the lack of a suitable framework to ensure the reproducibility of the experiments and the validity of the conclusions reached; one of the main components of such a framework should be one or more datasets with updated normal and attack records. One of the attempts to establish an adequate framework was the DARPA/MIT Lincoln Lab framework in 1998 and 1999 [164] and the KDD Cup dataset [165] derived from them. However, these frameworks suffer from significant limitations and have been repeatedly criticized [166]. The lack of publicly available data is explained by the data's sensitive nature: the inspection of network traffic can reveal highly sensitive information of an organization. Due to the lack of public data, researchers are forced to assemble their own datasets, generally with no access to appropriately sized networks: activity found in a small laboratory network cannot be generalized to larger scale network [167].

The public datasets that have been used in the different studies to carry out the proposed experiments are detailed below:

- **DARPA:** Created in 1998 by Lincoln Laboratory, Massachusetts Institute of Technology (MIT), promoted by DARPA and the Air Force Research Laboratory. After some time, due to the quickly changing of technology, another version of the dataset was created in 1999 (including novel attacks and a Windows NT target) [168]. DARPA98 and DARPA99 consist of raw tcpdump data, allowing testing of 244 labeled instances of 58 different attacks on four operating systems (SunOS, Solaris, Linux, and Windows NT) [164]. These datasets have been widely criticized by several academic papers [18,19], mainly due to the use of artificial data: customized software was used to synthesize typical user behavior and usual network traffic to generate a small, isolated network as if it were part of an Air Force network. According to McHugh [18], the dataset suffers from flaws in traffic data collection as there is no statistical evidence of similarity to typical Air Force network traffic (mainly with respect to the false alarm rate), attack taxonomy and distribution, and evaluation criteria. In the work of Mahoney et al. [19], numerous irregularities were found, including the fact that, due to the way the data had been generated, all malicious packets had a TTL of 126 or 253, while most normal traffic had a TTL of 127 or 254.
- **KDD Cup 99:** This is a transformed version of the DARPA dataset containing 41 features appropriate for machine learning classification algorithms. The data set can be obtained in three different versions: a complete training set, a 10% version of the training set and a test data set. Records duplication on both training and test sets can produce skewed results for more common cases [17]. Based on the works of McHugh [18] and Mahoney and Chan [19], the archival authority of Irvine KDD Archive, University of California, discourage the use of DARPA and KDD Cup 99 data sets [20].
- **NSL-KDD:** Created in 2009 by the Information Security Center of Excellence (ISCX), University of New Brunswick (UNB), in order to solve the problem of duplicate records found in KDD Cup 99 [17]; this duplication of records could cause biased results by the learning algorithms, as well as the lack of learning of infrequent records. After applying the cleanup operations, the recordset of 4,900,000 and 2,000,000 in the KDD Cup 99 training and test data set was reduced to 125,973 and 22,544 in the new NSL-KDD training and test data sets, respectively.
- **UNSW-NB15:** This dataset was created on by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours; has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms [169,170]. The number of records is 175,341 and 82,332 in the training and testing sets respectively. The simulation period was 16 h on 22 January 2015 and 15 h on 17 February 2015 [171].
- **Kyoto 2006:** Both KDD Cup '99 dataset and NSL-KDD dataset do not reflect real data flow in computer network since they are generated by simulation over the virtual network. The Kyoto 2006+ data set is built on real three year-traffic data from November 2006 to August 2009. This data set is captured using honeypots, darknet sensors, e-mail server and web crawler [172].
- **ISCX:** Shiravi et al. [173] devised a systematic approach to be able to generate datasets to analyse and evaluate intrusion detection systems, mainly through the use of anomaly detection techniques. It is intended that researchers will be able to generate datasets from a set of profiles that can be combined to create a diverse set of dataset. From this work, the ISCX (Information Security Centre of Excellence) dataset emerged. This dataset consists of simulated traffic for one week, each record consists of 11 different features. The dataset is labeled, containing a description of the legitimate network traffic and attacks.
- **CSIC-2010:** The CSIC 2010 dataset contains the generated traffic targeted to an e-commerce Web application developed at Spanish Research National Council (CSIC) . In this web application, users can buy items using a shopping cart and register by providing some personal information. The dataset was generated automatically and contains 36,000 normal requests and more than 25,000 anomalous requests; the requests are labeled as normal or anomalous [174].

- **ECML/PKDD 2007:** As part of the 18th European Conference on Machine Learning (ECML) and the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)—ECML/PKDD 2007 Discovery Challenge, a dataset was provided containing 35,006 requests classified as normal traffic and 15,110 requests classified as attacks. The dataset was generated by collecting real traffic which was then processed to mask parameter names and values, replacing them with random values [175].

Table 9 summarizes the public datasets that have been used in the analyzed works while Figure 4 provides an overview of the percentage of datasets used in the different studies reviewed.

Table 9. Public datasets used.

Dataset	Number of Studies	Citations
CSIC 2010	13	[43,59,64,65,72,74,80,85–87,93,114,117]
KDD-Cup 99	8	[58,78,80,81,91,97,106,116]
DARPA	4	[77,82,100,102]
NSL-KDD	2	[75,80]
UNSW-NB15	2	[75,115]
Kyoto 2006	2	[80,116]
ISCX	1	[116]
ECML/PKDD 2007	1	[114]

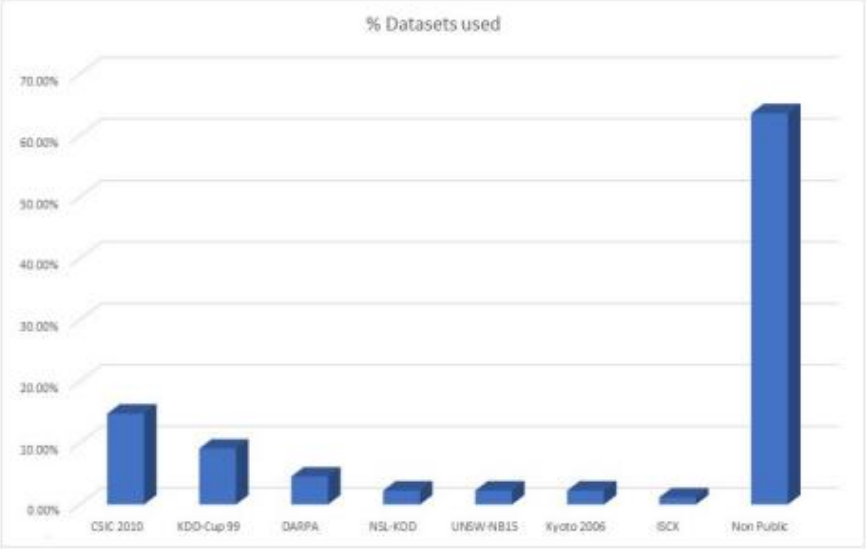


Figure 4. % of datasets used.

4.4. Metrics

This section details the most commonly used metrics to evaluate the different experiments carried out in the works that have been reviewed. A summary is given in Table 10.

- **Accuracy:** Accuracy (ACC) is ratio of payloads correctly identified divided by the total generated payloads.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

- **False Alarm Rate, False Positive Rate:** False Alarm Rate (FAR) or False Positive Rate (FPR), it's the probability that a false alarm will be raised: that a positive result will be given when the true value is negative.

$$FAR = \frac{FP}{FP + TN}$$

- **True Negative Rate, Specificity:** True Negative Rate (TNR) or Specificity measures the proportion of actual negatives that are correctly identified.

$$TNR = \frac{TN}{FP + TN}$$

- **True Positive Rate, Recall, Sensitivity, Detection Rate:** True Positive Rate (TPR), Recall, Sensitivity or Detection Rate (DR) measures the proportion of actual positives that are correctly identified.

$$TPR = \frac{TP}{TP + FN}$$

- **Precision, Positive Predictive Value:** Precision or Positive Predictive Value (PPV) is the ratio of the number of malicious payloads correctly detected divided by the number of total malicious payloads.

$$PPV = \frac{TP}{TP + FP}$$

- **False Negative Rate:** False Negative Rate (FNR) is the proportion of positives that yield negative test outcomes with the test, i.e., the conditional probability of a negative test result given that the condition being looked for is present.

$$FNR = \frac{FN}{FN + TP}$$

- **F1-Score:** F1 score is a measure of test's accuracy, considering both the precision and recall. The F1-Score, also called F-measure or F-Score, is the weighted harmonic mean of two measures: precision (P) and recall (R) [176–178] .

$$F1 - Score = \frac{1}{\alpha \cdot \frac{1}{P} + (1 - \alpha) \cdot \frac{1}{R}}$$

- **Classification error:** Classification error (CE) depends on the number of samples incorrectly classified (false positives plus false negatives) and is evaluated by the formula:

$$CE = \frac{f}{n} \cdot 100$$

where f is the number of sample cases incorrectly classified, and n is the total number of sample cases.

- **Matthews Correlation Coefficient:** The Matthews Correlation Coefficient (MCC) is used in machine learning as a measure of the quality of binary (two-class) classifications [179]. MCC is a correlation coefficient between the observed and predicted binary classifications; it returns a value between −1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than

random prediction and −1 indicates total disagreement between prediction and observation.It's equivalent to *phi coefficient*.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- **Area Under the Curve:** A Receiver Operating Characteristic Curve or ROC curve is a graph that shows the performance of a classification model at all classification thresholds, representing the true positive rate (TPR) against the false positive rate (FPR). The AUC measures the entire two-dimensional area below the full ROC curve, ranging from 0 (model with 100% incorrect predictions) to 1 (model with 100% correct predictions), representing degrees of separability and thus indicating a model's ability to distinguish between classes. [180].

Table 10. Metrics used to evaluate experiments.

Metric	Number of Studies	Citations
FPR / FAR	61	[32,33,35,36,38–46,48–50,52,55,56,58,60,62–67,69,71,72,74,75,79,81–83,85–87,91,94–102,104,105,107–113,116–118]
TPR/Recall/ Sensitivity/DR	49	[33,35–42,45–50,52,53,60,65–67,69–72,74–76,79–83,85–87,89,91,93,94,97,98,100–102,105–115,117,118]
Accuracy	24	[35,36,41,47,48,50,53,59,60,64,66,75,80,81,88,89,92,99,100,103,105,113,114,116]
F-Score	12	[47,59,62,65,71,76,80,89,113–115,118]
Precision	11	[41,43,59,65,66,76,80,89,113–115]
ROC/AUC	10	[34,50,55,71,72,77,83,90,107,115]
FNR	4	[55,56,95,100]
TNR/Specificity	3	[59,93,114]
CE	1	[163]
MCC	1	[71]

5. Discussion

This section details the different findings after careful review of the different studies evaluated.

- **Attacks:** 11 (12.5%) of the 88 studies reviewed focus on prevention and mitigation of SQLI attacks. The same amount and percentage applies to prevention and mitigation of denial of service (DDoS) attacks. Both types of attacks are included as high risk in various attack classification systems such as OWASP Top Ten Project [181], Common Attack Pattern Enumeration and Classification(CAPEC) [182], Common Weakness Enumeration (CWE) [183], OWASP Automated Threat Handbook Web Applications [184], etc. Only 2 of the studies reviewed target the detection of Botnet attacks (2.27%) and two more focus on the detection of Defacement attacks (2.27%). Also, the type of attack is not clearly specified in the vast majority of studies reviewed: 62 out of 88 (70.45%). Figure 5 presents an overview of the percentage of studies focused on the prevention or detection of specific attacks.

  Due to the wide variety of web attacks that currently exist, it seems necessary to make efforts aimed at specifying clearly and in detail the types of attacks studied in the new papers that will be published from now on. In addition, it is suggested to use recognized resources (e.g., OWASP Top Ten, CAPEC, etc.) to be able to determine the types of attacks with the highest prevalence and thus be able to focus on their study and detection.

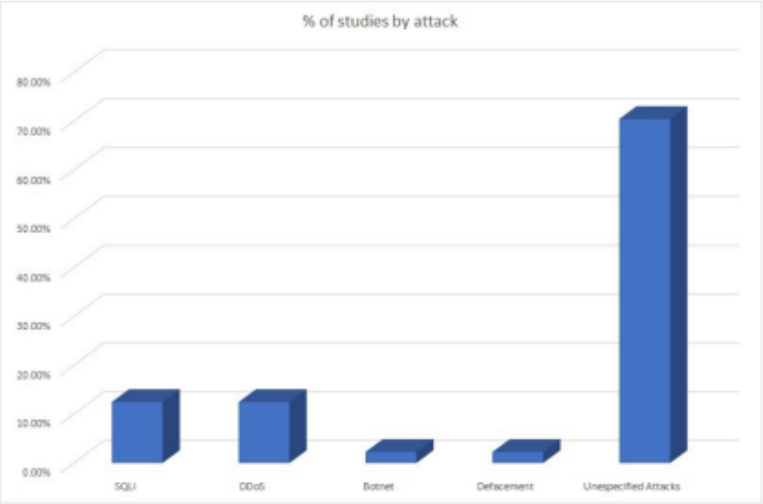


Figure 5. % of studies grouped by attack.

- **Algorithms and techniques used:**
  - CLUSTERING ALGORITHMS: 27 (30.68%) of the 88 studies reviewed used some kind of clustering algorithm. 7 (25.93%) of these 27 studies used more than one type of clustering algorithm: for example, Zolotukhin et al. [35] used 5 clustering algorithms, namely: K-Means, DBSCAN, single-linkage clustering, Self-Organizing Map (SOM), Fuzzy C-Means. The K-Means clustering algorithm is the most used: it is included in 7 of the 27 studies, although in 3 of them it is combined with other clustering algorithms. Gaussian Mixture Model (GMM) is used in 4 different studies, always in combination with other clustering algorithms: Yuan et al. [45] combine GMM with deep learning enabled subspace spectral ensemble clustering (DEP-SSEC) and Subspace Spectral Ensemble Clustering (SSEC). In [163], Kozik, Choraś and Holubowicz combine GMM with Expectation Maximisation (EM). Betarte et al. [93] combine GMM with Mahalanobis Distance and Expectation Maximisation (EM). In [94], Lin et al. combine GMM with Subspace Weighted Ensemble Clustering (SWEC).
  - CLASSIFICATION ALGORITHMS: 47 (53.41%) of the 88 studies reviewed propose some kind of classification algorithm (including those that propose the use of neural networks). Markov-type models represent a 31.91% (15 of 47 studies): Hidden Markov Model (HMM), Semi-Markov Model, Continuous Time Markov Chains, Markov Chain Monte Carlo, Markov Random Field, etc. In 13 (27.66%) of the 47 studies reviewed, the use of decision tree algorithms is proposed, Random Forest and Isolation Forest: Reduced Error Pruning Tree (REPTree), Decision Stump (DS), Isolation Forest, Information Gain (IG), XGBoost, etc. The use of Support Vector Machine (SVM) is proposed in 11 (23.40%) of the 47 studies, although 9 (19.15%) of them are of the One-Class Support Vector Machine (OCSVM) type, while the K-Nearest Neighbors (K-NN) algorithm is proposed in 5 (10.64%) of the 47 studies. Combining different classification algorithms is common: Wang and Zhang [103] introduced Information Gain based attribute selection, and then K-NN and OCSVM were used to detect anomalies. Zhang, Lu and Xu [63] propose a multi-model approach: First, the web request is partitioned into 7 fields: method, web resource, HTTP version, headers and headers inputs values are inspected by a probability distribution model, attribute sequence is inspected by HMM and attribute value is inspected by OCSVM. If one of the algorithms detects the request as anomalous, it is classified as anomalous. Kim and Cho [89] proposed a C-LSTM neural network to model the spatial and temporal information contained in traffic data. More complex characteristics are extracted from the data, combining Convolutional

Neural Network (CNN) and Recurrent Neural Network (RNN)—used to reduce frequency variation in the spatial information—, Long Short Term Memory (LSTM)—reduces temporal variation—and a Deep Neural Network (DNN)—used as a classifier, mapping the data in a more defined space. The detection of anomalies is done through the use of a SOFTMAX function.

- FEATURE SELECTION AND EXTRACTION: 17 of the 88 studies reviewed propose some kind of classification Feature Selection algorithm. 12 (70.59%) of these 17 studies are based on the N-grams algorithm for selection and feature extraction. 2 (11.76%) are based on the Bag of Words (BOW) algorithm. Vartouni, Kashi and Teshnehlab [59] propose an anomaly detection model in which the characteristics of web requests are extracted using an n-grams model based on character distribution. Subsequently, a dimensionality reduction is performed by means of SAE and finally, a classification task is performed by means of Isolation Forest. Zolotukhin et al. [60] propose an anomaly detection model in which n-grams are applied to extract the characteristics of web requests. Sample Entropy is used to capture the degree of dispersion or concentration of the web request parameter distribution in a given time interval. Unsupervised models are then used to generate a normal pattern. Using PCA, a dimensionality reduction is performed. Once PCA is applied, SVDD is applied to the transformed vectors of the training set, to build a hypersphere containing all the data in one category. To detect anomalies in the parameters of a web request, k-means is used. The detection of intrusions inside the user agent is carried out by DBSCAN. Asselin, Aguilar-Melchor and Jakllari [61] propose the use of n-grams to extract all the bi-grams from the URLs of a web server and generate a harmonic average of the probabilities that a given bi-gram is part of a normal request. New requests are classified as normal or abnormal depending on the probability of their bi-grams. Ren et al. [48] propose a model in which by means of BOW the extraction of characteristics of the web requests is carried out. Then, the detection of anomalies is done by HMM.
- DIMENSIONALITY REDUCTION: Only in 7 (7.95%) of the studies reviewed is some kind of dimensionality reduction technique applied. Principal Component Analysis (PCA) is applied in 6 of these 7 studies although in one of them it is combined with Random Projection (RP) and Diffusion Map (DM) and, in another one, it is combined with Sample Entropy. The remaining study applies Linear Discriminant Analysis (LDA) as a dimensionality reduction technique. Juvonen, Sipola and Hämäläinen [57] present the results from three methods that can be used for web log dimensionality reduction in order in order to allow the analysis and detection of anomalies: random projection, principal component analysis and diffusion maps. Based on the experimental results, they propose that RP and DM should be used together. RP methodology is efficient for daily analysis of huge amounts of traffic, while DM produces better visualizations and more accurate analysis of smaller amounts of data when needed. PCA falls in between of the other methods, but does not seem to offer any major advantages. In [49], Kozik, Choraś and Holubowicz estimated LDA transformation vector with Simulated Annealing approach in order to increase the effectiveness of SQL injection attack detection.

The use of dimensionality reduction techniques is recommended as an area for improvement in new research work. As previously mentioned, less than 8% of the papers reviewed incorporate this type of techniques, which allow the reduction of the number of characteristics of complex data samples, thus generating simpler models that avoid their overfitting to the training data and, therefore, a low performance with real data.

- **Datasets:** Only 26 (29.55%) of the 88 papers include public datasets in their experiments. The rest (70.45%) include private, synthetic or unspecified datasets, making it impossible to replicate the experiments and verify the results. Of the 26 studies that include public datasets, 12 of them

include datasets that belong to the DARPA and KDD families, which have been widely criticized in several studies [17–20].

The creation of public datasets, including new types of attacks, appears to be a major area to which new research efforts should be directed. Of considerable concern is the fact that the results obtained by 70% of the papers reviewed cannot be audited, replicated or validated because there is no access to the data on which these results are based. The availability of public data endorsed by the scientific community would allow further progress in the prevention of web attacks while avoiding problems associated with data privacy.

- **Metrics:** While most of the studies reviewed use FPR (61 of 88), TPR/Recall/Sensitivity/DR (49 of 88) and Accuracy (24 of 88), and which represent 69.32%, 55.68%, and 27.27% respectively, only 12 of them use F-Score (13.64%), 11 use precision (12.5%), and 10 use ROC/AUC (11.36%) as metrics to evaluate the experiments performed with the different algorithms analyzed. This fact is noteworthy because F-Score, accuracy and ROC/AUC are widely used metrics in work related to vulnerability detection, tool comparison, etc. [185].

The comparison of the results of the different techniques applied in the studies reviewed is difficult since there is no single public dataset widely accepted by the scientific community that allows for the replication of results, with most of the studies reviewed being based on non-publicly accessible datasets; because of this, the authors have decided to analyse the most representative results of the studies that are based on publicly accessible datasets.

Nguyen, Nguyen and Vu [77] combine an n-gram model for feature extraction and OCSVM for anomaly detection, based on the DARPA dataset, obtaining an AUC of 0.91425 for generic attacks, an AUC of 0.9912 for Shell-Code attacks and an AUC of 0.9831 for Traditional Polymorphic Attacks (CLET). Jamdagni et al. [102] propose an anomaly detection model based on Geometrical Structure Anomaly Detection (GSAD): this is a pattern recognition technique used in image processing. GSAD analyzes the correlations between various payload characteristics, using Mahalanobis Distance Map (MDM) to calculate the difference between normal and abnormal traffic, based on the DARPA dataset. The results obtained are 100% True Positive Rate (TPR) and 0.087% False Positive rate (FPR). Angiulli, Argento and Furfaro [82] identify anomalous packets in the DARPA dataset, by dividing the payload into segments of equal length, using n-grams to learn the byte sequences that usually appear in each chunk. Using a semi-supervised approach, a model is built that associates the protocol-packet length pair. This model is used to classify incoming traffic. Anomaly detection is carried out using Mahalanobis distance to determine whether a sequence of n-grams is unusual or not. A TPR of 100% and an FPR of 0.588% are obtained, but only for FTP traffic.

Wang et al. [58] use the Affinity Propagation (AP) algorithm by which they learn a subject's behavior through dynamic clustering of the data flow. It automatically tags the data and adapts to normal changes in behavior while detecting anomalies. Its study is based on the KDD-Cup 99 dataset and obtains a TPR of 98.4% and an FPR of 1.01%. Kaur and Bansal [81] implement the genetic algorithm (GA) to form clusters of normal and abnormal traffic in the training data set. The resulting clusters are used to generate normal and abnormal data partitions in the test data set. Finally, a classification of attack types is performed using SVM. The study is based on the KDD-Cup 99 dataset and obtains a TPR of 99.46%, an FPR of 1.96% and an Accuracy of 92.09%. Li et al. [97] propose an anomaly detection system based on Transductive Confidence Machines for K-NN (TCM-KNN). GA is used to reduce the size of the training dataset for the TCM-KNN model. Based on the KDD-Cup 99 dataset, it obtains a TPR of 99.38% and an FPR of 3.87%, complementing the TCM-KNN model with CHC (A genetic Algorithm called heterogeneous recombination and cataclysmic mutation used as search strategy).

Kamarudin et al. [75] propose a web anomaly detection model based on two stages: pre-processing and data mining. Pre-processing adopts the Hybrid Feature Selection (HFS) technique. In this phase, the characteristics are extracted. In the data mining phase, the classification is done by the LogitBoost algorithm. This study is based on the NSL-KDD and UNSW-NB15 datasets, obtaining a TPR of 89.75%, a FPR of 8.22% and a 90.33% of Accuracy in the NSL-KDD dataset; in the UNSW-NB15 dataset, the results obtained are as follows TPR: 99.1%, FPR: 0.18%, Accuracy: 99.45%.

Moustafa, Misra and Slay [79] propose a 4-step methodology: 1- Collect attack data by crawling websites and represent this data as feature vectors, 2- Extract features by association rules (ARM), 3- Use the extracted features to simulate web attacks, 4- Use a new Outlier Gaussian Mixture (OGM) algorithm to detect attacks. The data is obtained from the UNSW-NB15 dataset and from a non-publicly accessible dataset; a TPR of 95.56% and an FPR of 4.43% are obtained with the UNSW-NB15 dataset data. The metrics obtained with the data from the private dataset are TPR of 97.28% and FPR of 2.72%.

Alhakami et al. [116] propose a model in which patterns of activities (both normal and abnormal) are learned through Bayesian-based MCMC inference for infinite bounded generalized Gaussian mixture models. Unlike classical clustering methods, this approach does not need to specify the number of clusters, takes into account uncertainty by introducing prior knowledge for the parameters of the model, and allows to solve problems related to over and under evaluation. To obtain better cluster performance, weights of characteristics, model parameters and number of clusters are estimated simultaneously and automatically. The model evaluation data come from the KDD-Cup 99, Kyoto 2006 and ISCX datasets. The following metrics are obtained: Accuracy 83.49%, 87.41%, 90.4% FPR: 16.84%, 14.24%, 9.79% in the KDD-Cup 99, Kyoto 2006 and ISCX datasets respectively.

Alrawashdeh and Purdy [80] propose a rapid activation Adaptive Linear Function (ALF) that increases the speed of convergence and the accuracy of deep learning networks in real-time applications. Accuracy levels are as follows: 96.57%, 98.59%, 99.96% and 98.4% in the CSIC-2010, KDD-Cup 99, NSL-KDD and Kyoto 2006 datasets respectively.

Vartouni, Kashi and Teshnehlab [59] propose a model in which the characteristics of the web requests are extracted through an n-grams model based on character distribution. Subsequently, a dimensionality reduction is performed by means of SAE and finally a classification task is carried out by means of Isolation Forest. The evaluation data come from the dataset CSIC-2010; the metrics obtained are TPR: 88.34%, Precision: 80.29%, Accuracy: 88.32%, Specificity: 88.32%, F-Score: 84.12%.

Parhizkar and Abadi [64] propose a model in which, in the training phase, an initial set of One-Class Support Vector Machine is extracted (OCSVM) on the basis of legitimate requests. This initial set of OCSVM is then pruned using the ABC BeeSnips algorithm, with the aim of finding a "quasi-optimal" subset. In the detection phase, the outputs of the OCSVM present in the subset are combined to classify a new request as normal or abnormal. Based on the CSIC-2010 dataset, the metrics obtained are as follows: TPR: 95.9%, FPR: 2.82%, Accuracy: 96.54%.

Betartte et al. [93] try to improve the detection of ModSecurity through two approaches: the first, by building a One-Class Classification model. The second, by using N-grams for anomaly detection. In the One-Class Classification model, Gaussian Mixture Model (GMM) is used as the probability density distribution of the training data. Expectation Maximisation (EM) is used to estimate GMM parameters and the number of clusters. In the use of n-grams, Mahalanobis distance is used to detect the anomalies. The model evaluation data come from the CSIC-2010 dataset; for N-grams = 3, it obtains a TPR of 96.1% and a specificity of 99.5%.

Moradi Vartouni, Teshnehlab and Sedighian Kashi [114] propose a model that uses stacked auto-encoder (SAE) and deep belief network as feature learning methods in which only normal data is used in the learning phase. Subsequently, OCSVM, Isolation Forest and Elliptic Envelope are used as classifiers. The model is based on the datasets CSIC-2010 and ECML/PKDD 2007; the best values obtained are DR: 89.48%, Specificity: 89.11%, F-Score: 85.35% in the CSIC-2010 dataset and TPR: 89.75%, Specificity: 78.25%, F-Score: 84.93% in the ECML/PKDD 2007 dataset.

Kozik and Choraś [65] propose a model in which, firstly, an extraction of the tokens and the data between them is performed, to later transform them into a characteristics vector by distributing characters in different ranges of the ASCII table, thus achieving a reduction in dimensionality. The classification is carried out by means of a set of One Class Classifiers, specifically Decision Stump (DS) and RepTree. The data come from the dataset CSIC-2010 and the results obtained are TPR: 98.3%, FPR: 1.9%, Precision: 94.6%, F-Score: 96.4%.

Choraś and Kozik [72] propose a model in which they use a graphical approach to build a set of regular expressions to model HTTP requests. An attempt is made to group similar web requests and represent them using a single pattern. The measure of dissimilarity between components is done by the Needleman-Wunsch algorithm. The data come from the dataset CSIC-2010, obtaining a TPR of 94.46% and an FPR of 4.5%.

Kozik et al. [87] present a model in which a character pattern extraction method is used in web requests, based on the ICD (Ideal Character Distribution) proposed by Kruegel [186], but applied to different ranges of the ASCII table in order to reduce the dimensionality. The data come from the dataset CSIC-2010 and the results obtained are TPR: 86.6%, FPR: 1.8% for 1% of the dataset (300 samples), TPR: 95.6%, FPR: 5.8% for 10% of the dataset (3000 samples), TPR: 96.9%, FPR: 6.8% for 20% of the dataset (6000 samples), TPR: 97.7%, FPR: 8.1% for 100% of the dataset (32,400 samples).

Kozik, Choraś and Hołubowicz [43] propose a model in which the genetic algorithm is used to determine the valid subset of extracted tokens and their correct order. The data between tokens is classified by assigning the distribution of characters to different intervals in the ASCII table. RepTree and DS are used as classifiers. The data come from the extended CSIC-2010 dataset *(CSIC-2010+)* using new data samples collected during penetration tests carried out in a web-based Geographic Information System (GIS)-system. The total number of records from the original dataset was increased by 3.6%, adding 2.08% of normal samples (around 1500 requests) as well as 7.9% of new attacks (around 2000 anomalous requests). This results in a Precision level of 98%.

In [74], Kozik, Choraś and Hołubowicz propose the processing of web requests in order to extract vectors of constant length characteristics. Then k-means is applied in order to group web requests of similar characteristics. Finally, through multiple sequence alignment (MSA), an analysis of the request structure is performed. The data come from the dataset CSIC-2010+, obtaining a TPR of 92.7% and a FPR of 6.4%.

Kozik and Choraś [85] propose a model in which a genetic algorithm (GA) is used to realign HTTP payload sequences and extract their structure. The number of characters falling into different ranges of the ASCII table is counted. A hybrid classification technique combining REPTree and AdaBoost is used. The data come from the dataset CSIC-2010+, obtaining a TPR of 91.5% and an FPR of 0.7%.

In [86], Kozik et al. propose a model in which, after extracting the characteristics of web requests by distributing characters that fall within a given range of the ASCII table, Extreme Learning Machine is applied to classify the data as normal or abnormal. The data come from the dataset CSIC-2010+, obtaining a TPR of 94.98% and an FPR of 0.79%.

In [117], Kozik and Choraś propose a modified version of the Random Forest algorithm as a classifier. The study uses data from the dataset CSIC-2010+ and the results obtained are TPR: 93.5% and FPR: 0.5%.

Please note that the results of studies using data from the DARPA and KDD-Cup 99 datasets should be taken with caution, as these datasets have been widely criticized by the scientific community [17–20].

Table 11 provides a summary of the results of the different studies based on public datasets.

The authors recommend selecting the appropriate result validation metrics based on the type of vulnerability scenario to be protected (Business-Critical Applications, Heightened-Critical Applications, Best Effort, Minimum Effort) as recommended in [185].

Table 11. Metrics by study and dataset.

Study	TPR	FPR	Precision	Accuracy	Specificity	F-Score	AUC	Dataset
Nguyen, Nguyen and Vu [77]							91.42% [1] 99.12% [2] 98.31% [3]	DARPA
Angiulli, Argento and Furfaro [82]	100% [4]	0.59% [4]						DARPA
Jamdagni et al. [102]	100%	0.087%						DARPA
Wang et al. [58]	98.4%	1.01%						KDD-Cup 99
Kaur and Bansal [81]	99.46%	1.96%						KDD-Cup 99
Li et al. [97]	99.38%	3.87%						KDD-Cup 99
Kamarudin et al. [75]	89.75% 99.1%	8.22% 0.18%		90.33% 99.45%				NSL-KDD UNSW - NB15
Moustafa, Misra and Slay [79]	95.56% 97.28%	4.43% 2.72%						UNSW - NB15 Private Dataset
Alhakami et al. [116]		16.84% 14.24% 9.79%		83.49% 87.41% 90.4%				KDD-Cup 99 Kyoto 2006 ISCX
Alrawashdeh and Purdy [80]				96.57% 98.59% 99.96% 98.4%				CSIC 2010 KDD-Cup 99 NSL-KDD Kyoto 2006
Vartouni, Kashi and Teshnehlab [59]	88.34%		80.29%	88.32%	88.32%	84.12%		CSIC 2010
Parhizkar and Abadi [64]	95.9%	2.82%		96.54%				CSIC 2010
Betartte et al. [93]	96.1% [5]				99.5% [5]			CSIC 2010
Moradi Vartouni, Teshnehlab and Sedighian Kashi [114]	89.48% 89.75%				89.11% 78.25%	85.35% 84.93%		CSIC 2010 ECML/PKDD 2007
Koziz and Choraś [65]	98.3%	1.9%	94.6%			96.4%		CSIC 2010
Choraś and Kozik [72]	94.46%	4.5%						CSIC 2010
Kozik et al. [87]	86.6% [6] 95.6% [7] 96.9% [8] 97.7% [9]	1.8% [6] 5.8% [7] 6.8% [8] 8.1% [9]						CSIC 2010
Kozik, Choraś and Hołubowicz [43]			98%					CSIC 2010 +
Kozik, Choraś and Hołubowicz [74]	92.7%	6.4%						CSIC 2010 +
Kozik and Choraś [85]	91.5%	0.7%						CSIC 2010 +
Kozik et al. [86]	94.98%	0.79%						CSIC 2010 +
Kozik and Choraś [117]	93.5%	0.5%						CSIC 2010 +

[1] For generic attacks. [2] For Shell Code. [3] For CLET. [4] Only for FTP traffic. [5] For N-grams = 3. [6] 1% of dataset (300 samples). [7] 10% of dataset (3000 samples). [8] 20% of dataset (6000 samples). [9] 100% of dataset (32,400 samples).

6. Conclusions

In this work, a systematic review of the available literature on the detection of web attacks using anomaly detection techniques has been carried out, following the guidelines provided by Kitchenham et al. [5–9]. One of the major drawbacks detected in this systematic review is the unavailability of a standardized, updated and correctly labeled dataset, which allows the verification of the experimental results obtained in the different studies. It is worrying that only 29.55% of the experimental results obtained in the studies reviewed are based on public datasets; of these, approximately 50% are based on datasets that are strongly criticized by the scientific community, so it seems clear that more research efforts are needed to allow for the creation and validation of a public dataset, maintained by the scientific community, that incorporates a sufficient number of normal and abnormal requests, so as to allow for the replication and validation of studies conducted on that dataset.

A small number of studies in which dimensionality reduction techniques are applied have also been detected. Dimensionality reduction allows the analysis of a larger amount of data in a shorter period of time, simplifying the complexity of sample spaces with many dimensions while preserving their information. If PCA is used, the use of robust methods is recommended, as the PCA method is highly sensitive to outliers. If an observation has an anomaly in one of its variables, the variance in this direction will be artificially high. Since PCA tries to identify the directions with the highest variance, the subspace created will have been over-guided in this direction.

Most of the studies reviewed apply grouping algorithms using K-means and GMM classification with Markov and SVM type models. A combination of two or more clustering and/or classification algorithms is common.

A reduced use of classic metrics is detected in works related to vulnerability detection such as F-Score, accuracy and ROC/AUC; however metrics such as FPR, DR/Accuracy and TPR are widely used. Although these last metrics may be valid, the authors believe that more research efforts should be made in this area, in order to establish a concrete methodology that facilitates the choice of particular metrics depending on the type of study being conducted.

In the review of the studies carried out, it was found that most of them do not clearly specify the type of attack they are trying to prevent, although there is a small number that investigate DDoS, injection, botnets and defacement attacks. Further research efforts may be needed to generate studies that investigate the prevention and detection of other types of attacks.

In general, high statistical performances are observed in the different papers that incorporate deep learning techniques, but in those that report on the datasets used, it is detected that the results depend largely on the datasets, with Accuracy's percentages decreasing as the dataset becomes more recent. Therefore, in addition to encouraging the generation and use of public datasets that allow the replication and validation of experiments as indicated above, it should be further analyzed whether deep learning really improves intrusion detection systems.

Author Contributions: Conceptualization, T.S.R., J.-R.B.H., J.B.H. and J.-A.S.M.; methodology, T.S.R., J.-R.B.H., J.B.H. and J.-A.S.M.; investigation, T.S.R.; resources, T.S.R.; writing—original draft preparation, T.S.R.; validation, J.-R.B.H., J.B.H., J.-J.M.H. and J.-A.S.M. writing—review and editing, T.S.R., J.-R.B.H., J.B.H., J.-J.M.H. and J.-A.S.M.; visualization, J.-R.B.H., J.B.H., J.-J.M.H. and J.-A.S.M.; supervision, J.-R.B.H., J.B.H., J.-J.M.H. and J.-A.S.M.; project administration, J.-J.M.H.; funding acquisition, J.-J.M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Appendix Quality Assurance Form

Appendix A.1. Section 1

Does the study refer to the use of anomaly detection to improve security and prevent attacks in web environments?

- This section includes: intrusion detection, DDoS attacks, IDS/IPS improvement, WAF, RASP, detection and prevention of any attack classified in OWASP Top Ten.
- The study should focus on the detection of anomalies using different techniques: probability distribution models, Markov models, one-class SVM, clustering models, data minig and, in general, any automatic learning technique.

The first section sets out the basis for the initial selection of the study. If the answer to this section is positive, we will move on to the next section. Otherwise, the study will not be included in the systematic review.

Appendix A.2. Section 2

Is the study quantitative?

- Check whether the study provides any kind of empirical measure that allows the objective interpretation of the data obtained.

Only quantitative studies are to be included in the systematic review, so if the answer to this section is yes, we will move on to the next section. Otherwise, the study will not be included in the systematic review.

Appendix A.3. Section 3

Please indicate for each of the following questions the score among the following possible values: (1)—Yes , (−1)—No, (0.5)—Partially, (0)—Not applicable.

1. Are the research objectives clearly defined?
2. Is the choice of techniques used correctly justified?
3. Are the techniques used correctly detailed?
4. Is the measurement of the variables involved correct?
5. Are data collection methods correctly described?
6. Are objective indicators applied for the analysis of the results obtained?
7. Are the objective indicators applied adequate, correctly described and justified?
8. Are all research objectives adequately addressed?
9. Are negative results specified?
10. Are problems of validity or trust of the results obtained adequately discussed?
11. Is there a clearly defined relationship between objectives, data obtained, interpretation and conclusions?

This section details the quality assessment questions for the study itself. Each of the questions has 4 possible answers with an associated score: Yes (1 point), No (−1 point), Partially (0.5 points), Not Applicable (0 points). Each study can obtain from 0 to 11 points (if the score obtained by a study is negative, it will be left at 0); the second quartile (11/2 = 5.5) is taken as the cut-off point, in such a way that those studies whose score is lower than 5.5 points will not be included in the systematic review.

Appendix B. Appendix Data Retrieval Form

Data Item	Description
Study identifier	Unique ID for the study
Date of data retrieval	
Bibliographic data	Title of the study, name(s) of the author(s), publication year
Type of study	Journal, conference, workshop, etc.
Study objective	What is the principal objective of the study, what research areas the study is focused on
Study classification	Case study, experiment, survey, comparative analysis, etc.
Type of anomaly detection technique used	
What statistical indexes are used and its values	Values of the statistical indexes used for the objective validation of the results of the study experiments
Type of dataset	It refers specifically to the type of dataset used for the development of the experiment or the validation of the proposed model. (Publicly available, from private institution, synthetically generated, etc.)
Study findings	Major findings or conclusions from the study

References

1. Liao, H.J.; Richard Lin, C.H.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
2. Jyothsna, V. A Review of Anomaly based Intrusion Detection Systems. *Int. J. Comput. Appl.* **2011**, *28*, 26–35. [CrossRef]
3. Kakavand, M.; Mustapha, N.; Mustapha, A.; Abdullah, M.T.; Riahi, H. A Survey of Anomaly Detection Using Data Mining Methods for Hypertext Transfer Protocol Web Services. *JCS* **2015**, *11*, 89–97. [CrossRef]
4. Samrin, R.; Vasumathi, D. Review on anomaly based network intrusion detection system. In Proceedings of the 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Mysuru, India, 15–16 December 2017; pp. 141–147. [CrossRef]
5. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering Version 2.3*; Technical Report; Keele University: Keele, UK; University of Durham: Durham, UK, 2007.
6. Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.* **2007**, *80*, 571–583. [CrossRef]
7. Budgen, D.; Brereton, P. Performing Systematic Literature Reviews in Software Engineering. In Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, 20–28 December 2006; Association for Computing Machinery: New York, NY, USA; pp. 1051–1052. [CrossRef]
8. Kitchenham, B.; Pearl Brereton, O.; Budgen, D.; Turner, M.; Bailey, J.; Linkman, S. Systematic literature reviews in software engineering—A systematic literature review; *Inf. Softw. Technol.* **2009**, *51*, 7–15. [CrossRef]
9. Kitchenham, B.; Brereton, P. A Systematic Review of Systematic Review Process Research in Software Engineering. *Manuscr. Publ. Inf. Softw. Technol.* **2013**, *55*, 2049–2075. [CrossRef]
10. Patel, A.; Taghavi, M.; Bakhtiyari, K.; Celestino Júnior, J. An intrusion detection and prevention system in cloud computing: A systematic review. *J. Netw. Comput. Appl.* **2013**, *36*, 25–41. [CrossRef]
11. Raghav, I.; Chhikara, S.; Hasteer, N. Article: Intrusion Detection and Prevention in Cloud Environment: A Systematic Review. *Int. J. Comput. Appl.* **2013**, *68*, 7–11.
12. Patcha, A.; Park, J.M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.* **2007**, *51*, 3448–3470. [CrossRef]
13. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly Detection: A Survey. *ACM Comput. Surv.* **2009**, *41*. [CrossRef]

14. Jose, S.; Malathi, D.; Reddy, B.; Jayaseeli, D. A Survey on Anomaly Based Host Intrusion Detection System. *J. Phys. Conf. Ser.* **2018**. [CrossRef]
15. Fernandes, G.; Rodrigues, J.J.P.C.; Carvalho, L.F.; Al-Muhtadi, J.F.; Proença, M.L. A comprehensive survey on network anomaly detection. *Telecommun. Syst.* **2019**, *70*, 447–489. [CrossRef]
16. Kwon, D.; Kim, H.; Kim, J.; Suh, S.C.; Kim, I.; Kim, K.J. A survey of deep learning-based network anomaly detection. *Clust. Comput.* **2019**, *22*, 949–961. [CrossRef]
17. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A Detailed Analysis of the KDD CUP 99 Data Set. In Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; IEEE Press: Piscataway, NJ, USA, 2009; pp. 53–58.
18. McHugh, J. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 262–294. [CrossRef]
19. Mahoney, M.V.; Chan, P.K. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection BT—Recent Advances in Intrusion Detection. In *Recent Advances in Intrusion Detection*; Vigna, G., Kruegel, C., Jonsson, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 220–237.
20. Brugger, T. KDD Cup '99 dataset (Network Intrusion) considered harmful. *KDnuggets News* **2007**, *7*, 15.
21. Ieracitano, C.; Adeel, A.; Gogate, M.; Dashtipour, K.; Morabito, F.C.; Larijani, H.; Raza, A.; Hussain, A. Statistical Analysis Driven Optimized Deep Learning System for Intrusion Detection BT. In *Advances in Brain Inspired Cognitive Systems*; Ren, J., Hussain, A., Zheng, J., Liu, C.L., Luo, B., Zhao, H., Zhao, X., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 759–769.
22. Ieracitano, C.; Adeel, A.; Morabito, F.C.; Hussain, A. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing* **2020**, *387*, 51–62. [CrossRef]
23. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
24. Ahmed, M.; Naser Mahmood, A.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [CrossRef]
25. Kotu, V.; Deshpande, B. Chapter 13—Anomaly Detection. In *Data Science*, 2nd ed.; Kotu, V., Deshpande, B., Eds.; Morgan Kaufmann: Burlington, MA, USA, 2019; pp. 447–465. [CrossRef]
26. Hodge, V.J.; Austin, J. A Survey of Outlier Detection Methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
27. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
28. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
29. Pudil, P.; Novovičová, J. Novel Methods for Feature Subset Selection with Respect to Problem Knowledge BT—Feature Extraction, Construction and Selection: A Data Mining Perspective. In *Feature Extraction, Construction and Selection. The Springer International Series in Engineering and Computer Science*; Liu, H., Motoda, H., Eds.; Springer: Boston, MA, USA, 1998; Volume 453; pp. 101–116._7. [CrossRef]
30. Hu, H.; Zahorian, S.A. Dimensionality reduction methods for HMM phonetic recognition. In Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, USA, 14–19 March 2010; pp. 4854–4857. [CrossRef]
31. García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* **2009**, *28*, 18–28. [CrossRef]
32. Thang, T.M.; Nguyen, K.V. FDDA: A Framework For Fast Detecting Source Attack In Web Application DDoS Attack. In Proceedings of the Eighth International Symposium on Information and Communication Technology, Nha Trang, Vietnam, 7–8 December 2017; Association for Computing Machinery: New York, NY, USA, 2017; SoICT 2017; pp. 278–285. [CrossRef]
33. Tripathi, N.; Hubballi, N. Slow Rate Denial of Service Attacks against HTTP/2 and Detection. *Comput. Secur.* **2018**, *72*, 255–272. [CrossRef]
34. Najafabadi, M.M.; Khoshgoftaar, T.M.; Calvert, C.; Kemp, C. User Behavior Anomaly Detection for Application Layer DDoS Attacks. In Proceedings of the 2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, USA, 4–6 August 2017; pp. 154–161. [CrossRef]

35. Zolotukhin, M.; Hämäläinen, T.; Kokkonen, T.; Siltanen, J. Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. In Proceedings of the 2016 23rd International Conference on Telecommunications (ICT), Thessaloniki, Greece, 16–18 May 2016; pp. 1–6. [CrossRef]
36. Shirani, P.; Azgomi, M.A.; Alrabaee, S. A method for intrusion detection in web services based on time series. In Proceedings of the 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, NS, Canada, 3–6 May 2015; pp. 836–841. [CrossRef]
37. Tripathi, N.; Hubballi, N.; Singh, Y. How Secure are Web Servers? An Empirical Study of Slow HTTP DoS Attacks and Detection. In Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 454–463. [CrossRef]
38. Wang, C.; Miu, T.T.N.; Luo, X.; Wang, J. SkyShield: A Sketch-Based Defense System Against Application Layer DDoS Attacks. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 559–573. [CrossRef]
39. Wang, Y.; Liu, L.; Si, C.; Sun, B. A novel approach for countering application layer DDoS attacks. In Proceedings of the 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 25–26 March 2017; pp. 1814–1817. [CrossRef]
40. Xie, Y.; Tang, S. Online Anomaly Detection Based on Web Usage Mining. In Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, Shanghai, China, 21–25 May 2012; pp. 1177–1182. [CrossRef]
41. Lin, H.; Cao, S.; Wu, J.; Cao, Z.; Wang, F. Identifying Application-Layer DDoS Attacks Based on Request Rhythm Matrices. *IEEE Access* **2019**, *7*, 164480–164491. [CrossRef]
42. Xiao, R.; Su, J.; Du, X.; Jiang, J.; Lin, X.; Lin, L. SFAD: Toward effective anomaly detection based on session feature similarity. *Knowl.-Based Syst.* **2019**, *165*, 149–156. [CrossRef]
43. Kozik, R.; Choraś, M.; Hołubowicz, W. Evolutionary-based packets classification for anomaly detection in web layer. *Secur. Commun. Netw.* **2016**, *9*, 2901–2910. [CrossRef]
44. Wang, L.; Cao, S.; Wan, L.; Wang, F. Web Anomaly Detection Based on Frequent Closed Episode Rules. In Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, NSW, Australia, 1–4 August 2017; pp. 967–972. [CrossRef]
45. Yuan, G.; Li, B.; Yao, Y.; Zhang, S. A deep learning enabled subspace spectral ensemble clustering approach for web anomaly detection. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3896–3903. [CrossRef]
46. Bronte, R.; Shahriar, H.; Haddad, H. Information Theoretic Anomaly Detection Framework for Web Application. In Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, USA, 10–14 June 2016; Volume 2, pp. 394–399. [CrossRef]
47. Luo, Y.; Cheng, S.; Liu, C.; Jiang, F. PU Learning in Payload-based Web Anomaly Detection. In Proceedings of the 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC), Shanghai, China, 18–19 October 2018; pp. 1–5. [CrossRef]
48. Ren, X.; Hu, Y.; Kuang, W.; Souleymanou, M.B. A Web Attack Detection Technology Based on Bag of Words and Hidden Markov Model. In Proceedings of the 2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Chengdu, China, 9–12 October 2018; pp. 526–531. [CrossRef]
49. Kozik, R.; Choraś, M.; Hołubowicz, W. Hardening Web Applications against SQL Injection Attacks Using Anomaly Detection Approach. In *Image Processing & Communications Challenges 6*; Choraś, R.S., Ed.; Springer International Publishing: Cham, Switzerland, 2015; pp. 285–292.
50. Maggi, F.; Robertson, W.; Kruegel, C.; Vigna, G. Protecting a Moving Target: Addressing Web Application Concept Drift. In *Recent Advances in Intrusion Detection*; Kirda, E., Jha, S., Balzarotti, D., Eds.; Springer:Berlin/Heidelberg, Germany, 2009; pp. 21–40.
51. Valeur, F.; Vigna, G.; Kruegel, C.; Kirda, E. An Anomaly-Driven Reverse Proxy for Web Applications. In Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France, 23–27 April 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 361–368. [CrossRef]
52. Guangmin, L. Modeling Unknown Web Attacks in Network Anomaly Detection. In Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology, Busan, Korea, 11–13 November 2008; Volume 2, pp. 112–116. [CrossRef]
53. Yu, S.; Guo, S.; Stojmenovic, I. Fool Me If You Can: Mimicking Attacks and Anti-Attacks in Cyberspace. *IEEE Trans. Comput.* **2015**, *64*, 139–151. [CrossRef]

54. Sakib, M.N.; Huang, C. Using anomaly detection based techniques to detect HTTP-based botnet C C traffic. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6. [CrossRef]
55. Medvet, E.; Bartoli, A. On the Effects of Learning Set Corruption in Anomaly-Based Detection of Web Defacements. In *Detection of Intrusions and Malware, and Vulnerability Assessment*; Hämmerli, M.B., Sommer, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 60–78.
56. Davanzo, G.; Medvet, E.; Bartoli, A. Anomaly detection techniques for a web defacement monitoring service. *Expert Syst. Appl.* **2011**, *38*, 12521–12530. [CrossRef]
57. Juvonen, A.; Sipola, T.; Hämäläinen, T. Online anomaly detection using dimensionality reduction techniques for HTTP log analysis. *Comput. Netw.* **2015**, *91*, 46–56. [CrossRef]
58. Wang, W.; Guyet, T.; Quiniou, R.; Cordier, M.O.; Masseglia, F.; Zhang, X. Autonomic Intrusion Detection. *Know.-Based Syst.* **2014**, *70*, 103–117. [CrossRef]
59. Vartouni, A.M.; Kashi, S.S.; Teshnehlab, M. An anomaly detection method to detect web attacks using Stacked Auto-Encoder. In Proceedings of the 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Kerman, Iran, 28 February–2 March 2018; pp. 131–134. [CrossRef]
60. Zolotukhin, M.; Hämäläinen, T.; Kokkonen, T.; Siltanen, J. Analysis of HTTP requests for anomaly detection of web attacks. In Proceedings of the 2014 World Ubiquitous Science Congress: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, DASC 2014, Dalian, China, 24–27 August 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 406–411. [CrossRef]
61. Asselin, E.; Aguilar-Melchor, C.; Jakllari, G. Anomaly detection for web server log reduction: A simple yet efficient crawling based approach. In Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS), Philadelphia, PA, USA, 17–19 October 2016; pp. 586–590. [CrossRef]
62. Zhang, S.; Li, B.; Li, J.; Zhang, M.; Chen, Y. A Novel Anomaly Detection Approach for Mitigating Web-Based Attacks Against Clouds. In Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 3–5 November 2015; IEEE Computer Society: Piscataway, NJ, USA, 2015; pp. 289–294. [CrossRef]
63. Zhang, M.; Lu, S.; Xu, B. An Anomaly Detection Method Based on Multi-models to Detect Web Attacks. In Proceedings of the 2017 10th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 9–10 December 2017; Volume 2, pp. 404–409. [CrossRef]
64. Parhizkar, E.; Abadi, M. OC-WAD: A one-class classifier ensemble approach for anomaly detection in web traffic. In Proceedings of the 2015 23rd Iranian Conference on Electrical Engineering, Tehran, Iran, 10–14 May 2015; pp. 631–636. [CrossRef]
65. Kozik, R.; Choras, M. Adapting an Ensemble of One-Class Classifiers for a Web-Layer Anomaly Detection System. In Proceedings of the 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC, Krakow, Poland, 4–6 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 724–729. [CrossRef]
66. Cao, Q.; Qiao, Y.; Lyu, Z. Machine learning to detect anomalies in web log analysis. In Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 13–16 December 2017; pp. 519–523. [CrossRef]
67. Yu, J.; Tao, D.; Lin, Z. A hybrid web log based intrusion detection model. In Proceedings of the 2016 4th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2016, Beijing, China, 17–19 August 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 356–360. [CrossRef]
68. Threepak, T.; Watcharapupong, A. Web attack detection using entropy-based analysis. In Proceedings of theInternational Conference on Information Networking, Phuket, Thailand, 10–12 February 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 244–247. [CrossRef]
69. Swarnkar, M.; Hubballi, N. Rangegram: A novel payload based anomaly detection technique against web traffic. In Proceedings of the 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), Kolkata, India, 15–18 Decemnber 2015; pp. 1–6. [CrossRef]
70. Xu, H.; Tao, L.; Lin, W.; Wu, Y.; Liu, J.; Wang, C. A model for website anomaly detection based on log analysis. In Proceedings of the 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems, Shenzhen, China, 27–29 November 2014; pp. 604–608. [CrossRef]
71. Park, S.; Kim, M.; Lee, S. Anomaly Detection for HTTP Using Convolutional Autoencoders. *IEEE Access* **2018**, *6*, 70884–70901. [CrossRef]

72. Choraś, M.; Kozik, R. Machine learning techniques applied to detect cyber attacks on web applications. *Log. J. IGPL* **2014**, *23*, 45–56. [CrossRef]
73. Tharshini, M.; Ragavinodini, M.; Senthilkumar, R. Access Log Anomaly Detection. In Proceedings of the 2017 Ninth International Conference on Advanced Computing (ICoAC), Chennai, India, 14–16 December 2017; pp. 375–381.
74. Kozik, R.; Choraś, M.; Hołubowicz, W. Packets tokenization methods for web layer cyber security. *Log. J. IGPL* **2016**, *25*, 103–113. [CrossRef]
75. Kamarudin, M.H.; Maple, C.; Watson, T.; Safa, N.S. A LogitBoost-Based Algorithm for Detecting Known and Unknown Web Attacks. *IEEE Access* **2017**, *5*, 26190–26200. [CrossRef]
76. Yu, Y.; Liu, G.; Yan, H.; Li, H.; Guan, H. Attention-Based Bi-LSTM Model for Anomalous HTTP Traffic Detection. In Proceedings of the 2018 15th International Conference on Service Systems and Service Management (ICSSSM), Hangzhou, China, 21–22 July 2018; pp. 1–6.
77. Nguyen, X.N.; Nguyen, D.T.; Vu, L.H. POCAD: A novel pay load-based one-class classifier for anomaly detection. In Proceedings of the 2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), Danang, Vietnam, 14–16 September 2016; pp. 74–79. [CrossRef]
78. Lu, L.; Zhu, X.; Zhang, X.; Liu, J.; Bhuiyan, M.Z.A.; Cui, G. One Intrusion Detection Method Based On Uniformed Conditional Dynamic Mutual Information. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA , 1–3 August 2018; pp. 1236–1241. [CrossRef]
79. Moustafa, N.; Misra, G.; Slay, J. Generalized Outlier Gaussian Mixture technique based on Automated Association Features for Simulating and Detecting Web Application Attacks. *IEEE Trans. Sustain. Comput.* **2018**, 1. [CrossRef]
80. Alrawashdeh, K.; Purdy, C. Fast Activation Function Approach for Deep Learning Based Online Anomaly Intrusion Detection. In Proceedings of the 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), Omaha, NE, USA, 3–5 May 2018; pp. 5–13. [CrossRef]
81. Kaur, R.; Bansal, M. Multidimensional attacks classification based on genetic algorithm and SVM. In Proceedings of the 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 14–16 October 2016; pp. 561–565. [CrossRef]
82. Angiulli, F.; Argento, L.; Furfaro, A. Exploiting N-Gram Location for Intrusion Detection. In Proceedings of the 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), Vietri sul Mare, Italy, 9–11 November 2015; pp. 1093–1098. [CrossRef]
83. Hiremagalore, S.; Barbará, D.; Fleck, D.; Powell, W.; Stavrou, A. transAD: An Anomaly Detection Network Intrusion Sensor for the Web. In *Information Security*; Chow, S.S.M., Camenisch, J., Hui, L.C.K., Yiu, S.M., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 477–489. [CrossRef]
84. Favaretto, M.; Spolaor, R.; Conti, M.; Ferrante, M. You Surf so Strange Today: Anomaly Detection in Web Services via HMM and CTMC. In *Green, Pervasive, and Cloud Computing*; Au, M.H.A., Castiglione, A., Choo, K.K.R.;, Palmieri, F., Li, K.C., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 426–440. [CrossRef]
85. Kozik, R.; Chorás, M. The http content segmentation method combined with adaboost classifier for web-layer anomaly detection system. *Adv. Intell. Syst. Comput.* **2017**, *527*, 555–563. [CrossRef]
86. Kozik, R.; Choraś, M.; Holubowicz, W.; Renk, R. Extreme Learning Machines for Web Layer Anomaly Detection. In *Image Processing and Communications Challenges 8*; Choraś, R.S., Ed.; Springer International Publishing: Cham, Switzerland, 2017; pp. 226–233. [CrossRef]
87. Kozik, R.; Choraś, M.; Renk, R.; Holubowicz, W. Patterns Extraction Method for Anomaly Detection in HTTP Traffic. In *Proceedings of the International Joint Conference*; Herrero, Á., Baruque, B., Sedano, J., Quintián, H., Corchado, E., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 227–236. [CrossRef]
88. Shi, Y.; Wang, S.; Zhao, Q.; Li, J. A Hybrid Approach of HTTP Anomaly Detection. In *Web and Big Data*; Springer International Publishing: Cham, Switzerland, 2017; pp. 128–137. [CrossRef]

89. Kim, T.Y.; Cho, S.B. Web traffic anomaly detection using C-LSTM neural networks. *Expert Syst. Appl.* **2018**, *106*, 66–76. [CrossRef]
90. Jin, X.; Cui, B.; Li, D.; Cheng, Z.; Yin, C. An improved payload-based anomaly detector for web applications. *J. Netw. Comput. Appl.* **2018**, *106*, 111–116. [CrossRef]
91. Wang, W.; Liu, J.; Pitsilis, G.; Zhang, X. Abstracting massive data for lightweight intrusion detection in computer networks. *Inf. Sci.* **2018**, *433–434*, 417–430. [CrossRef]
92. Liu, T.; Zhang, L. Application of Logistic Regression in WEB Vulnerability Scanning. In Proceedings of the 2018 International Conference on Sensor Networks and Signal Processing (SNSP), Xi'an, China, 28–31 October 2018; pp. 486–490. [CrossRef]
93. Betarte, G.; Gimenez, E.; Martinez, R.; Pardo, A. Improving Web Application Firewalls through Anomaly Detection. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 779–784. [CrossRef]
94. Li, B.; Yuan, G.; Shen, L.; Zhang, R.; Yao, Y. Incorporating URL embedding into ensemble clustering to detect web anomalies. *Future Gener. Comput. Syst.* **2019**, *96*, 176–184. [CrossRef]
95. Yun, Y.; Park, S.; Kim, Y.; Ryou, J. A Design and Implementation of Profile Based Web Application Securing Proxy. In *Information Security Practice and Experience*; Chen, K., Deng, R., Lai, X., Zhou, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 248–259. [CrossRef]
96. Bolzoni, D.; Etalle, S. Boosting Web Intrusion Detection Systems by Inferring Positive Signatures. In *On the Move to Meaningful Internet Systems: OTM 2008*; Meersman, R., Tari, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 938–955. [CrossRef]
97. Li, Y.; Guo, L.; Tian, Z.H.; Lu, T.B. A Lightweight Web Server Anomaly Detection Method Based on Transductive Scheme and Genetic Algorithms. *Comput. Commun.* **2008**, *31*, 4018–4025. [CrossRef]
98. Kruegel, C.; Vigna, G.; Robertson, W. A multi-model approach to the detection of web-based attacks. *Comput. Netw.* **2005**, *48*, 717–738. [CrossRef]
99. Cho, S.; Cha, S. SAD: Web session anomaly detection based on parameter estimation. *Comput. Secur.* **2004**, *23*, 312–319. [CrossRef]
100. Yamada, A.; Miyake, Y.; Takemori, K.; Studer, A.; Perrig, A. Intrusion Detection for Encrypted Web Accesses. In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), Niagara Falls, ON, Canada, 21–23 May 2007; Volume 1, pp. 569–576. [CrossRef]
101. Yan, C.; Qin, Z.; Shi, Y. Sequence Analysis and Anomaly Detection of Web Service Composition. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12–14 December 2008; Volume 3; pp. 1043–1048. [CrossRef]
102. Jamdagni, A.; Tan, Z.; Nanda, P.; He, X.; Liu, R.P. Intrusion Detection Using GSAD Model for HTTP Traffic on Web Services. In Proceedings of the 6th International Wireless Communications and Mobile Computing Conference, Caen, France, 15 January 2010; Association for Computing Machinery: New York, NY, USA; pp. 1193–1197. [CrossRef]
103. Wang, W.; Zhang, X. High-Speed Web Attack Detection through Extracting Exemplars from HTTP Traffic. In Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, Taiwan, 21–24 March 2011; Association for Computing Machinery: New York, NY, USA; pp. 1538–1543. [CrossRef]
104. Kruegel, C.; Vigna, G. Anomaly detection of Web-based attacks. In Proceedings of the ACM Conference on Computer and Communications Security, Washington, DC, USA, 27–30 October 2003; ACM Press: New York, NY, USA; pp. 251–261. [CrossRef]
105. Rahnavard, G.; Najjar, M.S.A.; Taherifar, S. A method to evaluate Web Services Anomaly Detection using Hidden Markov Models. In Proceedings of the 2010 International Conference on Computer Applications and Industrial Electronics, Kuala Lumpur, Malaysia, 5–8 December 2010; pp. 261–265. [CrossRef]
106. Das, D.; Sharma, U.; Bhattacharyya, D.K. A Web Intrusion Detection Mechanism based on Feature based Data Clustering. In Proceedings of the 2009 IEEE International Advance Computing Conference, Patiala, India, 6–7 March 2009; pp. 1124–1129. [CrossRef]
107. Li, X.; Xue, Y.; Malin, B. Detecting Anomalous User Behaviors in Workflow-Driven Web Applications. In Proceedings of the 2012 IEEE 31st Symposium on Reliable Distributed Systems, Irvine, CA, USA, 8–11 October 2012; pp. 1–10. [CrossRef]
108. Le, M.; Stavrou, A.; Kang, B.B. DoubleGuard: Detecting Intrusions in Multitier Web Applications. *IEEE Trans. Dependable Secur. Comput.* **2012**, *9*, 512–525. [CrossRef]

109. Xie, Y.; Yu, S.Z. Light-weight detection of HTTP attacks for large-scale Web sites. In Proceedings of the 2008 11th IEEE Singapore International Conference on Communication Systems, Guangzhou, China, 19–21 November 2008; pp. 1182–1186. [CrossRef]
110. Sriraghavan, R.G.; Lucchese, L. Data processing and anomaly detection in web-based applications. In Proceedings of the 2008 IEEE Workshop on Machine Learning for Signal Processing, Cancun, Mexico, 16–19 October 2008; pp. 187–192. [CrossRef]
111. Fan, W.K.G. An adaptive anomaly detection of WEB-based attacks. In Proceedings of the 2012 7th International Conference on Computer Science Education (ICCSE), Melbourne, VIC, Australia, 14–17 July 2012; pp. 690–694. [CrossRef]
112. Kirchner, M. A framework for detecting anomalies in HTTP traffic using instance-based learning and k-nearest neighbor classification. In Proceedings of the 2010 2nd International Workshop on Security and Communication Networks (IWSCN), Karlstad, Sweden, 26–28 May 2010; pp. 1–8. [CrossRef]
113. Kakavand, M.; Mustapha, A.; Tan, Z.; Yazdani, S.F.; Arulsamy, L. O-ADPI: Online Adaptive Deep-Packet Inspector Using Mahalanobis Distance Map for Web Service Attacks Classification. *IEEE Access* **2019**, *7*, 167141–167156. [CrossRef]
114. Moradi Vartouni, A.; Teshnehlab, M.; Sedighian Kashi, S. Leveraging deep neural networks for anomaly-based web application firewall. *IET Inf. Secur.* **2019**, *13*, 352–361. [CrossRef]
115. Li, J.; Fu, Y.; Xu, J.; Ren, C.; Xiang, X.; Guo, J. Web application attack detection based on attention and gated convolution networks. *IEEE Access* **2019**, 1. [CrossRef]
116. Alhakami, W.; ALharbi, A.; Bourouis, S.; Alroobaea, R.; Bouguila, N. Network Anomaly Intrusion Detection Using a Nonparametric Bayesian Approach and Feature Selection. *IEEE Access* **2019**, *7*, 52181–52190. [CrossRef]
117. Kozik, R.; Choraś, M. Protecting the application layer in the public domain with machine learning methods. *Log. J. IGPL* **2018**, *27*, 149–159. [CrossRef]
118. Jin, L.; Wang, X.J.; Zhang, Y.; Yao, L. Anomaly Detection in the Web Logs Using Unsupervised Algorithm. In *Human Centered Computing*; Tang, Y., Zu, Q., Rodríguez García, J.G., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 393–405. [CrossRef]
119. Bhattacharyya, D.K.; Kalita, J.K. *DDoS Attacks: Evolution, Detection, Prevention, Reaction, and Tolerance*; CRC Press: Boca Raton, FL, USA, 2016.
120. OWASP Foundation. Injection Flaws | OWASP, Available online: https://owasp.org/www-community/Injection_Flaws (accesed on 15 May 2020).
121. Wei, K.; Muthuprasanna, M.; Kothari, S. Preventing SQL injection attacks in stored procedures. In Proceedings of the Australian Software Engineering Conference (ASWEC'06), Sydney, NSW, Australia, 18–21 April 2006; pp. 8–198. [CrossRef]
122. Leonard, J.; Xu, S.; Sandhu, R. A Framework for Understanding Botnets. In Proceedings of the 2009 International Conference on Availability, Reliability and Security, Fukuoka, Japan, 16–19 March 2009; pp. 917–922.
123. Hadianto, R.; Purboyo, T.W. A Survey Paper on Botnet Attacks and Defenses in Software Defined Networking. *Int. J. Appl. Eng. Res.* **2018**, 13, *483–489.*
124. Gurjwar, R.K.; Sahu, D.R.; Tomar, D.S. An approach to reveal website defacement. *Int. J. Comput. Sci. Inf. Secur.* **2013**, *11*, 73.
125. Cluster analysis—Wikipedia. Available online: https://en.wikipedia.org/wiki/Cluster_analysis#Definition. (accesed on 9 February 2020).
126. Unioviedo. kmeans. Available online: https://www.unioviedo.es/compnum/labs/new/kmeans.html. (accessed on 17 May 2020).
127. Frey, B.J.; Dueck, D. Clustering by Passing Messages Between Data Points. *Science* **2007**, *315*, 972–976. [CrossRef] [PubMed]
128. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters a Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; KDD'96; pp. 226–231.

129. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*; Association for Computing Machinery: New York, NY, USA, 2000; pp. 93–104. [CrossRef]
130. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Stat. Soc. Ser. B (Methodol.)* **1977**, *39*, 1–38.
131. Gupta, M.R.; Chen, Y. Theory and Use of the EM Algorithm. *Found. Trends Signal Process.* **2011**, *4*, 223–296. [CrossRef]
132. Rahul, A.E.; Narukulla, S. Introduction to Data Mining and Machine Learning Algorithms. *Int. J. Res. Eng. Sci. Manag.* **2018**, *1*.
133. Duda, R.O.; Hart, P.E. *Pattern Classification and Scene Analysis*; Duda, R.O., Hart, P.E., Eds.; Wiley: New York, NY, USA, 1973.
134. Schölkopf, B.; Williamson, R.; Smola, A.; Shawe-Taylor, J.; Platt, J. Support Vector Method for Novelty Detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1999; pp. 582–588.
135. Franzese, M.; Iuliano, A. Hidden Markov Models. In *Encyclopedia of Bioinformatics and Computational Biology*; Ranganathan, S., Gribskov, M., Nakai, K., Schönbach, C.B.T.E.O.B.; Eds.; Academic Press: Oxford, UK, 2019; pp. 753–762. [CrossRef]
136. Rabiner, L.; Juang, B. An introduction to hidden Markov models. *IEEE ASSP Mag.* **1986**, *3*, 4–16. [CrossRef]
137. Altman, N.S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *Am. Stat.* **1992**, *46*, 175–185. [CrossRef]
138. Maron, M.E. Automatic Indexing: An Experimental Inquiry. *J. ACM* **1961**, *8*, 404–417. [CrossRef]
139. Domingos, P.; Pazzani, M. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Mach. Learn.* **1997**, *29*, 103–130. [CrossRef]
140. Webb, G.I.; Boughton, J.R.; Wang, Z. Not So Naive Bayes: Aggregating One-Dependence Estimators. *Mach. Learn.* **2005**, *58*, 5–24. [CrossRef]
141. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef] [PubMed]
142. Liu, G.; Bao, H.; Han, B. A Stacked Autoencoder-Based Deep Neural Network for Achieving Gearbox Fault Diagnosis. *Math. Probl. Eng.* **2018**, *2018*, 5105709. [CrossRef]
143. Puig-Arnavat, M.; Bruno, J.C. Artificial Neural Networks for Thermochemical Conversion of Biomass. *Recent Adv. Thermo-Chem. Convers. Biomass* **2015**, 133–156. [CrossRef]
144. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
145. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2013; pp. 3111–3119.
146. Tomović, A.; Janičić, P.; Kešelj, V. n-Gram-based classification and unsupervised hierarchical clustering of genome sequences. *Comput. Methods Programs Biomed.* **2006**, *81*, 137–153. [CrossRef]
147. Tauritz, D. *Applications of n-grams*; Technical Report; Department of Computer Science, University of Missouri-Rolla: Rolla, MI, USA, 2002.
148. Manning, C.D.; Raghavan, P.; Schütze, H. Scoring, term weighting, and the vector space model. In *Introduction to Information Retrieval*; Manning, C.D., Schütze, H., Raghavan, P., Eds.; Cambridge University Press: Cambridge, UK, 2008; pp. 100–123. [CrossRef]
149. Stephen, R. Understanding inverse document frequency: On theoretical arguments for IDF. *J. Doc.* **2004**, *60*, 503–520. [CrossRef]
150. Roweis, S.T.; Saul, L.K. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* **2000**, *290*, 2323 LP – 2326. [CrossRef]
151. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *Philos. Mag. J. Sci.* **1901**, *2*, 559–572. [CrossRef]
152. Hotelling, H. Relations Between Two Sets of Variates. *Biometrika* **1936**, *28*, 321–377. [CrossRef]
153. Jolliffe, I.T. *Principal Component Analysis*; Springer Series in Statistics; Springer: New York, NY, USA, 2002. [CrossRef]

154. Fisher, R.A. The Use of Multiple Measurements in Taxonomic Problems. *Ann. Eugen.* **1936**, *7*, 179–188. [CrossRef]
155. McLachlan, G.J. *Discriminant Analysis and Statistical Pattern Recognition*; John Wiley & Sons: Hoboken, NJ, USA, 2004; Volume 544.
156. Rao, C.R. The Utilization of Multiple Measurements in Problems of Biological Classification. *J. R. Stat. Soc. Ser. B (Methodol.)* **1948**, *10*, 159–203. [CrossRef]
157. Coifman, R.R.; Lafon, S.; Lee, A.B.; Maggioni, M.; Nadler, B.; Warner, F.; Zucker, S.W. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 7426–7431. [CrossRef] [PubMed]
158. Coifman, R.R.; Lafon, S. Diffusion maps. *Appl. Comput. Harmon. Anal.* **2006**, *21*, 5–30. [CrossRef]
159. Delaporte, J.; Herbst, B.M.; Hereman, W.; der Walt Stéfan, V. An introduction to diffusion maps. In Proceedings of the 19th Symposium of the Pattern Recognition Association of South Africa (PRASA 2008), Cape Town, South Africa , 27–28 November 2008.
160. Steliga, K.; Szynal, D. On Markov-type inequalities. *Int. J. Pure Appl. Math.* **2010**, *58*, 137–152.
161. Pearson, K.X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1900**, *50*, 157–175. [CrossRef]
162. Olshausen, B.A. *Bayesian Probability Theory*; The Redwood Center for Theoretical Neuroscience, Helen Wills Neuroscience Institute at the University of California at Berkeley: Berkeley, CA, USA, 2004.
163. Kozik, R.; Choraś, M.; Renk, R.; Holubowicz, W. Semi-unsupervised Machine Learning for Anomaly Detection in HTTP Traffic. In Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, Wroclaw, Poland, 25–27 May 2015; Springer International Publishing: Cham, Switzerland, 2016; pp. 767–775. [CrossRef]
164. Lichman, M. *1999 DARPA Intrusion Detection Evaluation Dataset*; MIT Lincoln Laboratory: Cambridge, MA, USA, 2000.
165. Hettich, S.; Bay, S.D. The UCI KDD Archive. Available online: http://kdd.ics.uci.edu (accessed on 15 March 2020).
166. Sommer, R.; Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, USA, 16–19 May 2010; pp. 305–316. [CrossRef]
167. Sommer, R. *Viable Network Intrusion Detection: Trade-Offs in High-Performance Environments*; VDM Verlag: Saarbrücken, DEU, 2008.
168. Siddique, K.; Akhtar, Z.; Khan, F.A.; Kim, Y. KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research. *Computer* **2019**, *52*, 41–51. [CrossRef]
169. Moustafa, N. The UNSW-NB15 data set description.
170. Moustafa, N.M.; Slay, J. The significant features of the UNSW-NB15 and the KDD99 Data sets for Network Intrusion Detection Systems. In Proceedings of the 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), Kyoto, Japan, 5 November 2015. [CrossRef]
171. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
172. Singh, R.; Kumar, H.; Singla, R. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Syst. Appl.* **2015**, *42*, 8609–8624. [CrossRef]
173. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. [CrossRef]
174. Torrano-Gimenez, C.; Perez-Villegas, A.; Alvarez, G. A Self-learning Anomaly-Based Web Application Firewall. In *Computational Intelligence in Security for Information Systems*; Advances in Intelligent and Soft Computing; Herrero, Á., Gastaldo, P., Zunino, R., Corchado, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 85–92.
175. Raïssi, C.; Brissaud, J.; Dray, G.; Poncelet, P.; Roche, M.; Teisseire, M. Web Analyzing Traffic Challenge: Description and Results. In Proceedings of the ECML/PKDD'2007 Discovery Challenge, Warsaw, Poland, 17 September 2007; p. 6.
176. Van Rijsbergen, C. *Information Retrieval*, 2nd ed.; Butterworth-Heinemann: Newton, MA, USA, 1979.

177. Díaz, G.; Bermejo, J.R. Static analysis of source code security: Assessment of tools against SAMATE tests. *Inf. Softw. Technol.* **2013**, *55*, 1462–1476. [CrossRef]
178. Bermejo Higuera, J.R. Metodología de Evaluación de Herramientas de Análisis Automático de Seguridad de Aplicaciones Web Para su Adaptación en el ciclo de vida de Desarrollo. Ph.D. Thesis, Universidad Nacional Educación a Distancia (UNED), Madrid, Spain, 2013.
179. Matthews, B.W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Et Biophys. Acta (BBA) - Protein Struct.* **1975**, *405*, 442–451. [CrossRef]
180. Swets, J.A. *Signal Detection Theory and ROC Analysis in Psychology and Diagnostics: Collected Papers*; Scientific Psychology Series; Lawrence Erlbaum Associates, Inc.: Hillsdale, NJ, USA, 1996.
181. OWASP Foundation. OWASP Top Ten, 2017.
182. MITRE Corporation. CAPEC—Common Attack Pattern Enumeration and Classification (CAPEC), 2011.
183. MITRE Corporation. CWE—Common Weakness Enumeration.
184. OWASP Foundation. OWASP Automated Threats to Web Applications. Available online: https://owasp.org/www-project-automated-threats-to-web-applications/ (accessed on 3 March 2020).
185. Antunes, N.; Vieira, M. On the Metrics for Benchmarking Vulnerability Detection Tools. In Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Rio de Janeiro, Brazil, 22–25 June 2015; pp. 505–516. [CrossRef]
186. Kruegel, C.; Toth, T.; Kirda, E. Service Specific Anomaly Detection for Network Intrusion Detection. In *Proceedings of the 2002 ACM Symposium on Applied Computing*; Association for Computing Machinery: New York, NY, USA, 2002; pp. 201–208. [CrossRef]

4.2. Artículo 2

Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.020976 Published Online: 14 June 2022

ARTICLE

Systematic Approach for Web Protection Runtime Tools' Effectiveness Analysis

Tomás Sureda Riera[1,*], Juan Ramón Bermejo Higuera[2], Javier Bermejo Higuera[2], Juan Antonio Sicilia Montalvo[2] and José Javier Martínez Herráiz[1]

[1]Computer Science Department, University of Alcalá, Madrid, 28801, Spain

[2]Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja, Logroño, 26006, Spain

*Corresponding Author: Tomás Sureda Riera. Email: tomas.sureda@uah.es

Received: 22 December 2021 Accepted: 09 March 2022

ABSTRACT

Web applications represent one of the principal vehicles by which attackers gain access to an organization's network or resources. Thus, different approaches to protect web applications have been proposed to date. Of them, the two major approaches are Web Application Firewalls (WAF) and Runtime Application Self Protection (RASP). It is, thus, essential to understand the differences and relative effectiveness of both these approaches for effective decision-making regarding the security of web applications. Here we present a comparative study between WAF and RASP simulated settings, with the aim to compare their effectiveness and efficiency against different categories of attacks. For this, we used computation of different metrics and sorted their results using F-Score index. We found that RASP tools scored better than WAF tools. In this study, we also developed a new experimental methodology for the objective evaluation of web protection tools since, to the best of our knowledge, no method specifically evaluates web protection tools.

KEYWORDS

Web Application Firewall (WAF); Runtime Application Self Protection (RASP); F-Score; web attacks; experimental methodology

1 Introduction

Most computer security attacks against organizations involve an attempt to exploit a number of web application vulnerabilities [1].

As per the 2016 Internet Security Threat Report (ISTR) made by Symantec [2], 1.1 M of web application attacks are blocked every day and 78% of websites have some type of vulnerability, including 15% with critical vulnerabilities. The 2016 Web Applications Security Statistics Report published by WhiteHat Security [3] stated, "Web application attacks represent the greatest threat to an organization's security. Web app attacks represented 40% of breaches in 2015."

Attacks on web applications are significantly increasing year after year. The most prevalent attacks include SQL injection attacks (SQLi) and Cross Site Scripting (XSS) attacks [1].

The Open Web Application Security Project publishes the OWASP Top Ten document, currently in its 2021 version, which identifies the most critical risks in web applications of organizations. The top three in these lists are occupied by Broken authentication, Cryptographic Failures and Injection attacks [4].

Web Application Firewall (WAF) tools are one of the most widely used methods for web application protection. A WAF is deployed between the application and the requesting user to inspect the incoming traffic at the application layer of the OSI model and look for attack patterns, eventually blocking incoming malicious traffic. WAF devices perform a deep packet inspection of incoming network traffic and check the incoming traffic against a database of signatures or rules. These devices are independent of the programming language of the web application as they act before the malicious traffic can execute the code [5]. However, studies have proposed different ways to circumvent the protection provided by a WAF [6–8]: Normalization method, HTTP Parameter Pollution, HTTP Parameter Fragmentation, logical requests AND/OR, replacing SQL functions that get WAF signatures with their synonyms, using comments, case changing, and triggering a Buffer Overflow/WAF Crashing. Most of these techniques focus on *protocol-level evasion* that attemp to exploit the little differences in how WAFs see traffic and how backend web servers and applications see it.

Gartner [9] defined Runtime Application Self-Protection (RASP) tools as "a security technology that is built or linked into an application or application runtime environment, and is capable of controlling application execution and detecting and preventing real-time attacks." These tools combine real-time contextual awareness of the factors that have led to the application of current behavior [10,11]. These tools "embed" the security in the application server to be protected, intercepting all calls to the system to check whether they are secure. Thus, RASP tools depend entirely on the programming language of the application to protect.

On the other hand, Nicolett [12] reported that the WAF and RASP technologies play a fundamental role in protecting web applications, shielding the vulnerable parts of the application until they are mitigated through code correction by the developers' team.

Both technologies work in the application layer filtering the HTTP protocol. WAF is a black box technology. It intercepts the calls and responses to the logic of the application to be protected, performing a syntactic analysis to detect attacks. Thus, it does not require access to the logic of the application. On the other hand, RASP is a white box technology that installs an agent in the application server to examine the assignment of values to the variables in the process stack. The RASP may even have to examine the code of the application to decide if a payload is an attack attempt or not.

Referring to the principles of defense in depth, it is advisable to incorporate additional layers of defense into an organization's computer environment. Given the recent emergence of RASP tools, it is necessary to compare the two types of protection tools objectively. Such a comparison will allow us to determine whether these tools can be complementary and establish an additional defense layer, improving an organization's security.

However, to our best knowledge, there is no specific methodology that allows us to evaluate different web protection tools.

In this paper, we analyzed and evaluated the performance of different WAF and RASP tools in protecting a web application (simulated by a benchmark) against different types of attacks. We here aimed to determine the best suitable tool based on objective criteria through the collection and subsequent tabulation of false positives, false negatives, true positives and true negatives. The

different tools were sorted according to their effectiveness by using the F-Score index [13]. We have also developed a new experimental methodology to evaluate the different tools analyzed in this paper; the proposed methodology can be generalized for the evaluation of other tools that work in the network layer such as IDS and IPS. This methodology allows the use of any other automatic vulnerability scanning tool in addition to the one used in this work. It also includes different categories of vulnerabilities for analysis and different test benches in addition to those used in this work.

The main contributions of this study are:

- Analysis of the performance of different WAF and RASP tools protecting web applications.
- Using the F-Score indicator to present the results obtained by different web protection tools. The tools that show greater efficiency in the detection of attacks minimizing the presence of false positives obtain a higher score in the F-Score indicator.
- Development of a new experimental methodology to evaluate web protection tools.

2 Related Work

In Section 2.1, we present a review of the studies carried out on WAF technologies. In Section 2.2, we review studies on RASP technology, including studies on self-protecting, self-healing and self-adaptive systems. Finally, in Section 2.3, we present a review of the studies that evaluated and compared web protection tools.

2.1 Web Application Firewall

In this study, Byrne [14] highlighted the role played by the WAF in the in-depth defense of an IT system. He stated that the WAF detection methods evolved from a static to a dynamic approach because of their use of machine-learning techniques.

Schmitt et al. [15] pointed out that an attacker can remotely determine whether the request has been blocked (shorter response time) or not by a WAF by measuring the time used to return a response to a legitimate request and another malicious request. Based on this, the attacker can adjust the construction of the payloads to evade the protection of the WAF.

Holm et al. [16] estimated the effectiveness of web application firewalls in preventing of SQL injection attacks carried out by professional penetration testers. They considered the presence or absence of four conditions: the use of an experienced operator that monitors the WAF, tuning the WAF using an automated black box tool, using an experienced professional to tune the WAF, and spending significant effort to tune the WAF. They measured the effectiveness using 16 different operational scenarios that were judged by a panel of 49 experts. The authors did not use any objective measures like false positive, false negative, etc.

Different studies have attempted to improve the effectiveness of WAF solutions. Moosa [17] proposed a model based on an Artificial Neural Network (ANN) that can effectively protect against SQL injection attacks. This proposed model has a training phase and a working phase. In the training phase, the ANN was exposed to a set of normal and malicious data so as to train the ANN. In the working phase, the trained ANN was integrated into the WAF to protect the web application.

In an attempt to improve the effectiveness of the WAF rule sets, Auxilia et al. [18] proposed a Negative Security Model that monitors requests for anomalies, unusual behavior, and common web application attacks.

Applebaum et al. [19] conducted a survey showing the development of WAFs based on machine-learning methods.

2.2 RASP and Self-Protected Systems

Many studies examined self-protected, self-healing and self-adaptive systems.

Weyns et al. [20] conducted a literature review and showed that self-adaptive systems improved the flexibility, reliability and performance of the system. They also pointed out the lack of empirical systematic studies as well as the lack of industrial evidence in support of such systems.

Yuan et al. [21] proposed a taxonomy to classify self-protection systems and research approaches. The proposed taxonomy consisted of nine dimensions, which were grouped into two different categories. The first category, Research Positioning, characterized the objectives and the investigation of self-protection, i.e., the aspects of "WHAT." The second category, Technique Characterization, described the "HOW" aspects of self-protection research.

Yuan et al. [22] redefined this taxonomy was redefined. Their taxonomy contained 14 dimensions that were grouped into three categories: The first category, Approach Positioning, characterized the objectives and attempts of self-protection research: the "WHAT" aspects. The second category, Approach Characterization, classified the "HOW" aspects of self-protection research. The third category, Approach Quality, evaluated self-protection research.

Keromytis [23] characterized the general architecture of a self-healing system. In the proposed model, the system enters a self-diagnostic state on detecting anomalous behaviors in order to identify the fault and extract as much information as possible about the causes and impact on the system. Subsequently, the system attempts to self-adapt generating candidate fixes, which are tested in the self-testing phase in order to determine the fix that best suits the optimum state of the system; once the fix has been found, it is deployed on the protected system.

According to Lane [24], most RASP tools can learn from the applications they are protecting. The author differentiates between passive and active learning. The process helps to refine detection rules by adapting generic ones to match specific requests and add fraud detection capabilities. Another model used a security positive approach (whitelisting); this way, the RASP tool knows how API calls are made and what certain lines of code look like, blocking unknown patterns.

Taint analysis is one of the techniques used in RASP solutions. Haldar et al. [25] analyzed the externals to the web application data, marking them as untrusted (tainted data).

Zeller et al. [26] proposed a system with a self-protecting layer to protect it against attacks aimed at misusing business logic rules. Yin et al. [27] suggested a scheme that automatically added the injection filtering instruction into the existing web program using static code analysis technology.

Another study used RASP technology to design and implement the runtime self-protection framework of smart contracts [28].

2.3 Evaluation and Comparison of Protection Tools

The Web Application Firewall Evaluation Criteria [29] attempts to develop a standardized criterion for the evaluation of WAFs in order to provide users with a tool that allows an educated decision when selecting a WAF. However, it only describes a series of characteristics that must be taken into account in choosing a WAF.

The SANS Institute [30] compared the HP Application Defender's RASP solution and an unnamed WAF and reviewed their respective preventive and detective capabilities. However, this report lacked the metrics to establish an objective comparison between the two solutions compared within it.

Razzaq et al. [31] presented a critical analysis of different WAFs. However, their analysis was limited to "*defense mechanism*": security policy control, monitoring, blocking, response filtering, attack prevention, authentication, website cloaking, deep inspection, session protection and overall security performance. These criteria also included the management interface.

3 Materials and Methods

Section 3.1 presents the categories of vulnerabilities against which the various WAF/RASP protection tools will be analyzed. In Section 3.2, we present a review of the studies carried out on the generation of test cases capable of revealing vulnerabilities in a web application. In Section 3.3, we present the metrics used in our study and discuss the reasons for using this metrics in Section 3.4. In Section 3.5, we present the WAF and RASP tools that were evaluated and compared. In Section 3.6, we provide an explanation of how the different tools have been configured and the reason for doing it in that specific way. We also provide a detail of the categories of vulnerabilities covered by each tool. Finally, in Section 3.7, we explain the different attack detection methods used by each tool.

3.1 Categories of Vulnerabilities Analyzed

- Command Injection: Classified by Mitre with the code CWE-78 [32]. These attacks are produced usually by the lack of validation of user input data or calls to unsafe functions that execute commands from the operating system on which the web application is run. This vulnerability may allow an attacker to execute unexpected commands directly in the operating system.
- SQL Injection: Classified by Mitre with code CWE-89 [32]. This attack is caused by the lack of validation of the entries entered by the user, which are interpreted as part of the SQL query.
- Path Traversal: Classified by Mitre with code CWE-22 [32]. These vulnerabilities are exploited by entering user input of special characters such as ".." and "/". By not validating the entries, the use of these characters allows the application to escape from the restricted directory, where it is supposed to run, to other directories of the operating system.
- Cross-Site Scripting (XSS): Classified by Mitre with code CWE-79 [32]. This vulnerability occurs when the JavaScript code is injected, avoiding the Same Origin Policy, which states that scripts in a domain should not be able to access resources or execute code in a different domain. Code injection is possible due to the lack of validation of user inputs.
- Remote File Inclusion (RFI): Classified by Mitre with code CWE-98 [32]. This vulnerability allows the attacker to link to files located on other servers; thus, the attacker can obtain and run on untrusted code.
- Open Redirect: Classified by Mitre with code CWE-601 [32]. This vulnerability allows redirection to an external site through the data entered in a non-validated entry of a form, which can lead to different phishing attacks.

3.2 Test Cases

In the improvement of computer security research, it is essential to generate representative sets of test cases capable of efficiently revealing various vulnerabilities in an application. These sets of test

cases will typically be used in automated vulnerability scanning tools, allowing the generation and launch of a large number of attacks against a system to be audited. In this section, we will review different studies about the generation of test cases.

Shahriar et al. [33] proposed *MUSIC*, a tool that automatically generates mutants for applications written in JSP. This mutation-based testing approach uses nine mutation operators that inject SQL Injection Vulnerabilities (SQLIV) into the application source code. This approach forces the generation of an adequate test data set containing effective test cases capable of revealing SQLIV.

Gu et al. [34] proposed the generation of test data, mutating captured protocol messages and introducing regular expressions into the approach for constructing test case templates.

3.3 Metrics

- F-Score: The F-Score is also called F-measure or F1-Score; as can be seen in Eq. (1). It is the weighted harmonic mean of two measures: precision (P) and recall (R) [35].

$$F = \frac{1}{\alpha \frac{1}{P} + (1+\alpha)\frac{1}{R}} \tag{1}$$

where the weighting factor

$\alpha \in [0,\ 1]$

F-Score balanced, assigns the same weights to precision and recall, which means that

$$\alpha = \frac{1}{2}$$

Therefore, the balanced F-Score formula can be written as shown in Eq. (2).

$$F = 2\frac{PR}{P+R} \tag{2}$$

- Precision (P): It is the ratio of the number of malicious payloads correctly detected divided by the number of total malicious payloads detected. This measure is also known as Positive Predictive Value (PPV). See Eq. (3).

$$P = \frac{TP}{TP+FP} \tag{3}$$

- Recall (R): As can be seen in Eq. (4), is the ratio of the number of malicious payloads correctly detected divided by the number of known total malicious payloads. Also called True Positive Rate (TPR) or Sensitivity.

$$R = \frac{TP}{TP+FN} \tag{4}$$

The F-Score value is a compromise between Recall and Precision. It assumes values between the interval [0, 1]. The value of Recall is 0 if no malicious payload has been detected and will be 1 if all detected payloads are malicious and all malicious payloads have been detected.

- False Positive Rate (FPR): It is the ratio of incorrectly detected payloads to exploit a specific vulnerability associated with a non-vulnerable test case, divided by the number of payloads incapable of exploiting non-vulnerable test cases. In other words, FPR is the probability that a non-malicious payload is incorrectly detected as malicious (probability of false alarms). See Eq. (5).

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

- Negative Predictive Value (NPV): As it can be inferred from Eq. (6), NPV is the ratio of payloads correctly detected to be incapable of exploiting a specific vulnerability associated with a non-vulnerable test case, divided by the sum of the number of payloads incapable of exploiting non-vulnerable test cases and the number of payloads detected incorrectly as unable to exploit a vulnerable test case. That is the confidence that can be assigned to the absence of alerts before the generation of a specific attack.

$$NPV = \frac{TN}{TN + FN} \tag{6}$$

- Accuracy: It is the ratio of payloads correctly identified divided by the total generated payloads. See Eq. (7).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

3.4 Metrics Discussion

As we intend to compare an objective *intragroup* (each tool of a particular category, compared to the other tools of the same category) and an objective *intergroup* (each tool of a particular category, compared to the other tools of another category) of various web protection tools, it is essential to have indexes that allow numerical ordering of the efficiency of the various tools. Bermejo Higuera [13] and Diaz et al. [36] proposed the use of *recall*, *precision* and *F-Score* metrics to analyze the results. This proposed approach was a concrete methodology for their evaluation.

Accuracy is one of the simplest methods to evaluate a binary classification model. It classifies the web request as *attack* or *no attack* and has advantages such as easily interpretable. However, its disadvantage is that it is not robust when the data is unevenly distributed, or where there is a higher cost associated with a particular type of error.

On the other hand, the F-Score allows to take into account the costs associated with FP and FN detection as in the case of web attack detection in different criticality scenarios. As the level of criticality increases, the costs associated with FP detection will be significantly lower. In addition, F-Score allows the correction of Acuracy deviations due to class imbalance, as in the case of a dataset in which 90% of web requests are malicious. If a classifier classifies all web requests as attack, it will automatically get 90% accuracy, but it would be useless in a real scenario.

Both Precision and Recall must be examined to fully evaluate the effectiveness of a model. These metrics are closely related. When the classification threshold increases, the number of FP decreases but FN increases. In such a case, Precision increases while Recall decreases. If the classification threshold decreases, FP increases and FN decreases, and accordingly, Precision decreases while Recall increases.

The F-Score is used to combine the Precision and Recall measures into a single value so as to easily compare the combined performance of Precision and Recall between various solutions. F-Score gives equal importance to Precision and Recall, as is the *harmonic mean* of Precision and Recall. As can be seen in Eq. (8), for certain scenarios with different levels of criticality, the more general formula of F-Score F_β can be used, where β is chosen such that Recall is considered β times as important as Precision:

$$F_\beta = (1+\beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{8}$$

A common value of beta is 2, where recall has a weight 2 times higher than precision: in a high criticality scenario, it is desirable to minimize FN (undetected attacks) at the cost of increasing FP (normal requests detected as attacks).

3.5 WAF/RASP Tools to be Evaluated

The tools that we propose to analyze are detailed in Table 1.

Table 1: WAF and RASP tools to be evaluated

Tool	Technology	Commercial/Open source
ModSecurity	WAF	Open source
Imperva	WAF	Commercial
Fortify App. Defender	RASP	Commercial
Contrast	RASP	Commercial

3.5.1 ModSecurity

ModSecurity is a hybrid WAF that delegates part of its work on the web server. In the case of Apache ModSecurity delegates some of its roles through its integration with modules [37]. Apache performs the following functions:

- Decrypts SSL traffic.
- Splits the incoming connection flow into HTTP requests.
- Partially parses HTTP requests.
- Invokes ModSecurity, depending on the correct configuration context.
- Decompresses the bodies of HTTP requests as needed.
- In case Apache is configured as a reverse-proxy, it forwards the requests to the backend servers.

The functionality provided by ModSecurity could be summarized in the following points:

- Traffic parsing: The data that compose the web requests are examined by security analyzers that extract bits of data and store them for later use by the rules.
- Buffering: The request and response bodies are buffered, so that ModSecurity has access to the complete requests before passing them to the web application for processing. ModSecurity also has access to the complete responses before they are sent to the client. In this way, a reliable lock can be provided in the case of malicious traffic, although it is disadvantage includes requirement of an additional RAM for the storage of the request and response bodies.
- Full transaction log or audit log: This feature allows it to log all full HTTP traffic instead of logging only partial access log information. It records the headers and body of the request, as well as the headers and body of the response.
- Rules engine: It evaluates the transaction and carries out the actions defined in the rules in case the evaluated traffic coincides with any of the patterns defined in any of those rules.

Apache can be reconfigured to send the ModSecurity's error logs through Syslog onto a remote server acting as a Security Information and Event Management (SIEM). However, forwarded data are only the short version of the ModSecurity alert messages that appear in the Apache error_log file. To conduct proper incident response, the ModSecurity audit log files needs to be accessed [38].

3.5.2 Imperva

This is a cloud-based solution which provides characteristics such as load balancing, failover, Content Delivery Network (CDN) and DDoS protection. Imperva works by a change in the DNS of the applications to be protected, so that the traffic passes through the Imperva servers. In this way, all traffic is analyzed by the integrated WAF in the product, providing protection against the categories of OWASP Top Ten attacks. To maintain a holistic view of the security events produced, it allows the integration with different SIEM tools. It has a series of integrated WAF rules for protection against different types of attacks, allowing the user to create new rules through an editor. Imperva also provides a two-factor authentication available for any website without any changes in the application code and can implement virtual patching on many common applications such as WordPress, Joomla, Drupal, and others, to protect against known vulnerabilities and misconfiguration issues.

3.5.3 Fortify Application Defender

Fortify Application Defender is available as a SaaS solution or as an on-premise solution for applications developed in Java or .NET. It installs an agent (a jar file in the case of applications developed in Java, or an installable file in the case of applications developed in .NET) that is integrated in the application server to be protected.

This agent analyzes the requests made to the application and the response generated by the application. Depending on the configuration of the agent, it can monitor or block malicious requests.

The data are sent in real time to the servers of Fortify App. Defender through a web interface. The reports on the attack attempts that the application undergoes, detailing the type of attack in question, the severity of the attack, day and time of the attack, the request path, source IP address, status (monitor or protected), etc., will allow a more in-depth analysis of any malicious payload detected. The different categories of protective vulnerabilities can be easily activated or deactivated by setting them in monitor or protection mode.

All log data can be communicated to SIEM applications or to logs of compliance. The contextual perspective from the data flow of the applications and the execution logic allows the reconstruction of the malicious request and the traceability of the stack.

3.5.4 Contrast

It works by installing a jar or .NET file that is started with the application server to be protected, communicating with the "TeamServer," which is responsible for collecting and processing all information received. The contrast solution is available as a SaaS product or an on-premise product. It provides information about attacks produced in a web interface. It also provides extensive information about detected vulnerabilities, libraries in use and installed versions of these libraries, acting as an Interactive Application Security Testing (IAST) solution. The IAST solution uses an agent running on an application server or a library built into the code at compile time to create an instrumented version of the software. This can then detect behavior, indicating a vulnerability or an attack [39]. When a vulnerability is detected, it indicates the line of code in which it is possible to exploit the vulnerability, as well as indications to remedy the vulnerability, stack of calls, etc. As a RASP solution, it reports

exhaustively on the attacks. It indicates the blocked attacks and those that, although they have been made, have not been able to exploit the vulnerability.

3.6 Tools Configuration and Vulnerability Coverage

The only tool that could be configured is ModSecurity, because the implementation of the others is done by redirecting traffic to the provider's servers (case of Imperva) or by installing an agent (a ∗.jar or ∗.net file) in the application that will intercept the requests and responses of the application to protect, based on the policy marked from the tool's control panel (Fortify App. Defender and Contrast). The control panels of Imperva, Fortify App. Defender and Contrast are cloud-based. Because the only possible configurations in these control panels are the tool configuration in "monitor mode" or "block mode" and the activation and deactivation of different categories of vulnerabilities to be protected (SQLi, XSS, etc.), it has been opted to leave the configuration of all the tools analyzed in their default values (modifying only some ModSecurity rules to allow analysis through an automatic vulnerability scanning tool).

3.7 Attack Detection Methods

WAF and RASP tools are based on pattern analysis. WAF tools use syntactic analysis of requests and responses to check whether they match the patterns defined in their rules. This analysis is done in the application layer, without direct access to the logic of the application. In comparison, the RASP tools install an agent in the application to be protected and, thus, have access to direct interception of calls and responses inside the application. When the logic of the application, calls and responses are know, they can be compared more effectively against a system of rules.

Contrast makes use of bytecode instrumentation, a feature in Java used to help integrate programs and application features during development. It embeds an agent into an application to be protected, which will, thereafter, be directly monitored and protected from the inside out. With that agent, Contrast can directly access the application requests and responses and can easily perceive exploits and unknown threats. Owing to the instrumentation approach, Contrast can observe how an application responds when it is free of attacks, being able to gather as much information as possible before deciding to block an attack. This amount of information makes a difference when accurately detecting the different types of attacks.

The use of advanced data labeling is possible due to the bytecode instrumentation [40]. This allows the tracing of all the data that enters the application from an untrusted source. The data in an HTTP header can be labeled as "untrusted," while the data in an HTTP parameter can be labeled "cross site" because an attacker can send it through domains. Initially, these labels are applied to the complete information. The tags are updated each time the data is modified or examined. This data flow tracking only works for the most common injection-type vulnerabilities, such as SQL injection, XSS, LDAP injection, XPath injection, command injection, etc.

A summary of each tool characteristics can be seen in Table 2.

Table 2: Tools characteristics

	WAF tools		RASP tools	
	ModSecurity	Imperva	Fortify App. Defender	Contrast
How traffic is analyzed	Reverse proxy	Redirecting traffic via DNS changes	Installing an agent in the application	Installing an agent in the application
Attack detection	Comparison of patterns against a system of rules.	Comparison of patterns against a system of rules.	Comparison of patterns against a system of rules.	Comparison of patterns against a system of rules.
Where traffic analysis is done	Applic. layer	Applic. layer	Inside the application	Inside the application
Has access to the application logic	No	No	Yes	Yes
Makes use of anomaly detection	No	No	Unknown	Yes. Through bytecode instrumentation
Advanced tagging	No	No	No	Yes. Through bytecode instrumentation

4 Experimental Methodology

Since no test suite is specifically designed for the analysis of WAF and RASP solutions, the following methodology has been developed to allow the analysis of the different tools evaluated in the present work:

(i) Selection of categories of vulnerabilities against which protection tools are to be evaluated.

(ii) Selection of a benchmark with different test cases that serves as reference for the evaluation of the tools. The OWASP Benchmark[1] project was chosen for evaluating tool performance in the following vulnerability categories: command injection, SQL injection, path traversal, cross-site scripting. For the RFI and open redirect categories, the Web Application Vulnerability Scanner Evaluation Project (WAVESEP)[2] project was selected for evaluating tool performance.

(iii) As discussed in Section 3.5, the ModSecurity, Imperva, Fortify App. Defender and Contrast tools were selected.

(iv) Selection of the vulnerability scanner from which automated attacks against the benchmark will be launched. We have chosen the OWASP ZAP[3] tool.

[1] https://owasp.org/www-project-benchmark/
[2] https://github.com/sectooladdict/wavsep
[3] https://owasp.org/www-project-zap/

(v) Selection of vulnerable and non-vulnerable cases of the different vulnerability categories used to measure, respectively, True Positive (TP) and FP. A total of 238 vulnerable test cases and 134 non-vulnerable test cases were selected and broken down into six different vulnerability categories.

(vi) Generation of attacks from OWASP ZAP against the selected test cases of the two benchmarks, without any protection tool interposed.

(vii) Selection of payloads that generate an alert in OWASP ZAP (in the case of vulnerable test cases) and payloads that do not generate alerts in OWASP ZAP (in the case of non-vulnerable test cases). In this way, we identify payloads that generate alerts (TP) in vulnerable test cases and, on the other hand, payloads that do not generate alerts (True Negative-TN) in non-vulnerable test cases. In total, there were 1,341 payloads: 684 TP and 657 TN.

(viii) Interposition of the protection tool to be evaluated between the benchmark and OWASP ZAP.

(ix) Generation of attacks from OWASP ZAP against the selected test cases of the benchmark with the interposing tool.

(x) Revision of the alerts generated in OWASP ZAP and in the log's files of the protection tool in the same payloads selected in point (vii) and obtaining the indicators of TP, TN, FP and FN.

(xi) Tabulation of results and obtaining metrics, ranked using F-Score index.

An example of payload can be seen in Fig. 1, while a flowchart with the different steps to analyze the tools is shown in Fig. 2.

SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	BenchmarkTest00024=bar%27%29+AND+3085%3DCAST%28%28CHR%2858%29%7C%7CCHR%28105%29%7C%7CCHR%28122%29%7C%7CCHR%28109%29%7C%7CCHR%2858%29%29%7C%7C%28SELECT+%28CASE+WHEN+%283085%3D3085%29+THEN+1+ELSE+0+END%29%29%3A%3Atext%7C%7C%28CHR%2858%29%7C%7CCHR%28110%29%7C%7CCHR%28114%29%7C%7CCHR%28117%29%7C%7CCHR%2858%29%29+AS+NUMERIC%29+AND+%28%27XUVt%27%3D%27XUVt&foo=bar

Figure 1: A payload example

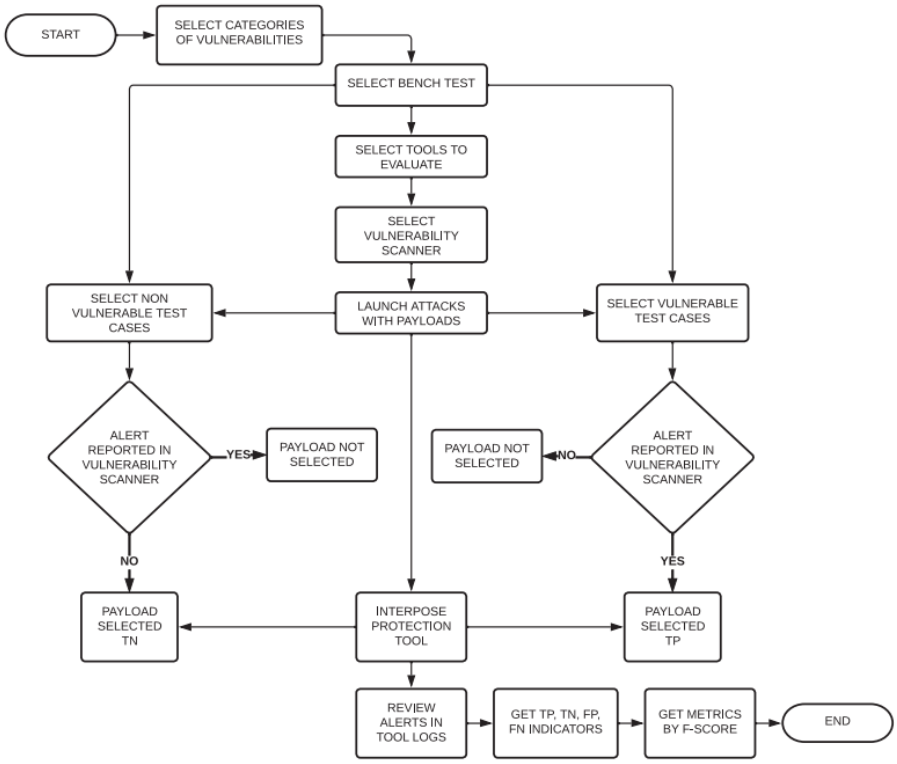


Figure 2: Steps to analyze the tools

5 Results

This section presents the results obtained by each of the evaluated solutions. The TOTAL row of each table was calculated based on the weighted average (excluding the extreme values[4]) of the results of each index of each category of vulnerabilities. Such an approach eliminated the influence caused by the different number of payloads of each category.

5.1 ModSecurity

As shown in Table 3, all categories of vulnerabilities show a high percentage of false positives, specifically in the SQL Injection and Cross-Site Scripting categories. ModSecurity systematically blocks all requests having the same number of TPs and FP, thereby causing the False Positive Rate (FPR) value to be 1. In the case of a high FN rate and a low TN rate, the value of the Negative Predictive Value is the lowest of all evaluated solutions (0.428). It also has a low level of global precision (0.507), indicative of the high number of FP. The Recall index is affected in different categories due to the number of FN, which affects the F-Score index, whose overall score is 0.541.

[4]Except in cases where it is impossible to eliminate the extremes due to the equality of results; in those cases, the weighted average will be calculated without the exclusion of extreme values.

Table 3: Results obtained by ModSecurity

Vuln.	#Payl.	#TP	#FP	#TN	#FN	Precision	Recall	FPR	NPV	Accuracy	F1-Score
Comm. injec.	156	52	52	26	26	0.5	0.667	0.667	0.5	0.5	0.571
SQLi	400	200	200	0	0	0.5	1	1	0	0.5	0.667
Path trav.	391	113	114	81	83	0.498	0.577	0.585	0.494	0.496	0.534
RFI	116	31	42	17	26	0.425	0.544	0.712	0.395	0.414	0.477
Redirect	156	33	28	36	59	0.541	0.359	0.438	0.379	0.442	0.431
XSS	122	61	61	0	0	0.5	1	1	0	0.5	0.667
Total	1341	490	497	160	194	0.507	0.655	0.684	0.428	0.487	0.541

5.2 Imperva

The results obtained by Imperva are detailed in Table 4. The Precision, Recall, FPR and F-Score in the Remote File Inclusion and Redirect categories were 0 as these categories did not contain any malicious payload. Specifically, the Precision index of this tool is the lowest of all evaluated, reaching an overall value of 0.501. The other categories contain a high number of FPs, and the categories of Path Traversal and Cross-Site Scripting have FPR as 1. The overall value of F-Score stands at 0.645.

Table 4: Results obtained by imperva

Vuln.	#Payl.	#TP	#FP	#TN	#FN	Precison	Recall	FPR	NPV	Accuracy	F1-Score
Comm. injec.	156	52	52	26	26	0.5	0.667	0.667	0.5	0.5	0.571
SQLi	400	191	190	10	9	0.501	0.955	0.95	0.526	0.503	0.657
Path trav.	391	196	195	0	0	0.501	1	1	0	0.501	0.668
RFI	116	0	0	59	57	0	0	0	0.509	0.509	0
Redirect	156	0	0	64	92	0	0	0	0.41	0.41	0
XSS	122	61	61	0	0	0.5	1	1	0	0.5	0.667
Total	1341	500	498	159	184	0.501	0.897	0.894	0.47	0.48	0.645

5.3 Fortify Application Defender

Table 5 presents the results obtained by Fortify Application Defender. No attacks were detected in the categories of Command Injection, Path Traversal and RFI. The overall Precision index of this tool was at 0.825. In the SQL Injection category, the correct detection rate TP and TN was 100%, reaching the value 1 in the Precision, Recall, Negative Predictive Value, Accuracy and F1-Score indices. It reaches a high number of TN; however, the high number of FN detected penalizes its score in the Negative Predictive Value index, which was 0.595. Its percentage of successes (Accuracy) was 0.582, and its F-Score was 0.777.

Table 5: Results obtained by fortify app. defender

Vuln.	#Payl.	#TP	#FP	#TN	#FN	Precision	Recall	FPR	NPV	Accuracy	F1-Score
Comm. injec.	156	0	0	78	78	0	0	0	0.5	0.5	0

(Continued)

Table 5 (continued)

Vuln.	#Payl.	#TP	#FP	#TN	#FN	Precision	Recall	FPR	NPV	Accuracy	F1-Score
SQLi	400	200	0	200	0	1	1	0	1	1	1
Path trav.	391	0	0	195	196	0	0	0	0.499	0.499	0
RFI	116	0	0	59	57	0	0	0	0.509	0.509	0
Redirect	156	62	0	64	30	1	0.674	0	0.681	0.808	0.805
XSS	122	59	39	22	2	0.602	0.967	0.639	0.917	0.664	0.742
Total	1341	321	39	618	363	0.825	0.803	0.058	0.595	0.582	0.777

5.4 Contrast

The results obtained by Contrast are detailed in Table 6. Due to the complete absence of FP, its Precision index was set to 1 and the False Positive Rate value was 0. In assition, the Recall index was 0.706 and its F-Score was 0.810 because of the relatively low number of FN and the high number of TPs. The Negative Predictive Value reaches the value of 0.764 due to the correct identification of TN and the low number of FN. The percentage of hits is around 83%, while the accuracy is at 0.832.

Table 6: Results obtained by contrast

Vuln.	#Payl.	#TP	#FP	#TN	#FN	Precision	Recall	FPR	NPV	Accuracy	F1-Score
Comm. injec.	156	13	0	78	65	1	0.167	0	0.545	0.583	0.286
SQLi	400	183	0	200	17	1	0.915	0	0.922	0.958	0.956
Path trav.	391	187	0	195	9	1	0.954	0	0.956	0.977	0.977
RFI	116	16	0	59	41	1	0.281	0	0.590	0.647	0.438
Redirect	156	46	0	64	46	1	0.5	0	0.582	0.705	0.667
XSS	122	42	0	61	19	1	0.689	0	0.763	0.844	0.816
Total	1341	487	0	657	197	1	0.706	0	0.764	0.832	0.81

5.5 Summary of Results and Comparison of Tools

Table 7 shows the different solutions sorted in descending order according to the score obtained by each of them in the F-Score index. In this index, the ordering was chosen because it indicates the relationship between Precision and Sensitivity (Recall) such that it objectively shows the relationship between TP, FP and TN.

This table also represents other indexes that provide additional information on the behavior of each solution. However, the FPR index has been modified in a way that its interpretation in the set of other indexes presented is more understandable by the reader. The FPR indicates the probability of false alarms so that the smaller its value is, the better is the performance of the solution evaluated, i.e., the value of the index is inversely proportional to the performance of the solution. Since the other five indexes presented have a directly proportional relationship between the value of each of the indexes and the performance of the solution, and that the possible values of each of them are in the interval [0, 1], the value of the complement of 1 according to the FPR was chosen to represent it in the tables. If, for example, the actual value of the FPR is 0.059, the value will be $1-0.059 = 0.941$.

Table 7: Results obtained by the different tools evaluated sorted by F-Score

Tool	F-Score	Precision	Recall	FPR	NPV	Accuracy
Contrast	0.810	1	0.706	1	0.764	0.832
Fortify App. Defender	0.777	0.825	0.803	0.942	0.595	0.582
Imperva	0.645	0.501	0.897	0.106	0.47	0.484
ModSecurity	0.541	0.507	0.655	0.316	0.428	0.487

ModSecurity and Imperva present the same results in several of the analyzed indexes (Precision, Accuracy and NPV), probably because request is intercepted before its arrival to the application. The comparison of the request against a set of established rules and standards, without the possibility of analyzing the calls to the system made by the application, may cause it to generate a high number of FP, which penalizes its score in Precision and, therefore, in F-Score. Imperva obtains a better score on the F-Score index (0.645) because of its high score on the Recall index (0.897). Obviously, the results obtained by WAF solutions will improve once the specific configuration applied to each solution has been modified depending on the environment and the application to be protected.

Apart from the ease of use, configuration and complementary services offered by the Imperva solution (load balancing, request accelerator, etc.), no significant differences are seen between Imperva and ModSecurity.

Contrast achieves a score of 1 in Precision, highlighting in all other indexes with respect to the other evaluated solutions, obtaining the highest score in F-Score (0.81).

Fortify App. Defender has high values in Precision (0.825) and FPR (0.942) but its score in the NPV and Accuracy indexes is penalized due to the high number of FN. However, it is still above the scores obtained by the WAF solutions. Fortify App. Defender can be considered a good option in the protection of specific vulnerabilities like SQLi, Redirect and XSS.

Compared to most security tools that suffer from an excess of FP and must be manually reviewed to rule out or confirm an attempt to exploit a particular vulnerability, RASP solutions present a clear superiority in Precision and almost total absence of FP. Hence, RASP solutions have a high degree of accuracy.

Fortify App. Defender and Contrast limit the creation of customized advanced rules by the user. Imperva allows the creation of custom rules through a proprietary scripting language. ModSecurity, on the other hand, offers the most configuration options since it allows access to the rules files, creation of new rules, etc. Obviously, the detection ratio of ModSecurity and Imperva would improve significantly when custom rules are added to these tools to adapt the execution environment of the application to protect. However, it would be a disadvantage for the other two tools to be evaluated, for this reason. Thus, it has been decided to leave all the tools in their default configuration.

By installing the Contrast agent in the application to be protected, instrumentation is added to identify vulnerabilities in the source code and subsequently detect attacks and block them from within the applications. With the interception of calls and responses from within the application, the correct attack detection rate increases significantly. RASP technology has a detailed view into the actions of the system and can help improve security accuracy. For example, RASP has insight into application logic, configuration, and data and event flows, which means it can detect attacks with high accuracy.

Contrast differentiates between attacks capable of exploiting a vulnerability (blocked) and attacks that have not been able to exploit a vulnerability (ineffective).

The tools as well as their results could be improved using machine-learning, decision trees, etc. in order to provide feedback to them according to the execution environment, thus, minimizing the detection rate of FP and FN increasing their efficiency. With ModSecurity, it is relatively easy to implement this improvement; regarding the other tools analyzed, it would be necessary for the manufacturer to provide more configuration options to the user, as well as the possibility of providing feedback to the tool (ideally through machine-learning techniques but also allowing the user to manually mark an event as FP or FN).

In Table 8, the ease of use, configurability, services offered and other features of each tool are compared.

Table 8: Tools' features

	WAF tools		RASP tools	
	ModSecurity	Imperva	Fortify App. Defender	Contrast
Installation	Hard. Manual installation	Ease. Changing DNS in order to redirect traffic to Imperva servers	Medium. Need to install java/net agent	Medium. Need to install java/net agent
Configurability	Full. Through configuration files	Limited. Through control panel	Limited. Through control panel	Limited. Through control panel
Control panel	No	Cloud	Cloud	Cloud
Tool in monitor/block mode	Modifying configuration file	Through control panel	Through control panel	Through control panel
Advanced custom rules	Through rules files. Sec-Rules language. Full control	Imperva rules proprietary scripting language. Advanced control	Point-wise protection. Limited control	Virtual patches. Limited control
Categories of vulnerabilities to protect	Modifying configuration file	Through control panel	Through control panel	Through control panel
Events analysis	Hard. No web interface for analysis	Easy. Through control panel	Easy. Through control panel	Easy. Through control panel
Common vulnerabilities and exposures (CVE) shields	No	No	No	Yes

(Continued)

Table 8 (continued)

	WAF tools		RASP tools	
	ModSecurity	Imperva	Fortify App. Defender	Contrast
Interactive application security testing (IAST)	No	No	No	Yes
Load balancer	No	Yes	No	No
Content delivery network (CDN)	No	Yes	No	No

6 Conclusions and Future Work

This paper compared different WAF and RASP protection solutions, using the OWASP ZAP vulnerability scanner against several test cases selected from two web applications. These two applications were considered benchmarks (OWASP Benchmark and Wavsep), interposing each of the solutions to be evaluated, in order to analyze their results. No additional configuration were made to the evaluated WAF-RASP solutions, except for the purpose of benchmark analysis by the vulnerability analysis tool OWASP ZAP.

The results and their analysis confirm the real degree of efficiency of the tools, obtaining precise TP, FP and FN ratios. Complementary metrics such as Accuracy, F-Score, NPV and ACC were considered to establish different rankings of the tools: TP, FP, TN, FN together. More specifically, the FPR ratio of each tool showed that the RASP tools have hardly any false positives, while the WAF tools obtain higher results between 0.68 and 0.89. Thus, the RASP tools are confirmed to be more accurate due to their white box nature.

Since there is a lack of an established methodology that specifies the steps to be followed to evaluate web application protection tools, the proposed methodology described in Section 4 can be considered for use in future research works. This methodology can be adapted on new requirements for evaluation of any protection tool against any benchmark.

It is necessary to analyze the performance of new web protection tools (e.g., AppWall) against new vulnerabilities and constantly appearing attacks (e.g., log4j). The use of the web tool evaluation methodology, defined in this work, is suitable as a starting point to achieve this goal.

The development of a feedback system for ModSecurity should be considered as part of future research work. There is evidence of a project called ModProfiler, currently discontinued and with very few references, which could be a starting point for it [41].

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Imperva (2015). WAAR 2015. Technical Report. Imperva. https://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed6.pdf.
2. Symantec (2016). Internet Security Threat Report. Technical Report. Symantec. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf.
3. WhiteHat Security (2016). Web Applications Security Statistics Report 2016. Technical Report. WhiteHat Security. https://info.whitehatsec.com/rs/675-YBI-674/images/WH-2016-Stats-Report-FINAL.pdf.
4. OWASP (2021). OWASP Top 10 2021. https://owasp.org/Top10/.
5. Clincy, V., Shahriar, H. (2018). Web application firewall: Network security models and configuration. *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 835–836. Tokyo, Japan.
6. Appelt, D., Nguyen, C. D., Panichella, A., Briand, L. C. (2018). A Machine-learning-driven evolutionary approach for testing web application firewalls. *IEEE Transactions on Reliability, 67(3),* 733–757. DOI 10.1109/TR.24.
7. OWASP (2022). SQL Injection Bypassing WAF-OWASP. https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF.
8. Garn, B., Sebastian Lang, D., Leithner, M., Richard Kuhn, D., Kacker, R. et al. (2021). Combinatorially xssing web application firewalls. *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 85–94. Porto de Galinhas, Brazil.
9. Gartner (2022). Runtime Application Self-Protection (RASP)-Gartner IT Glossary. http://www.gartner.com/it-glossary/runtime-application-self-protection-rasp.
10. Dubey, S. K. (2016). An evaluation of java applications using security requirements. *International Journal of Recent Trends in Engineering & Research Issue, 2(8),* 2455–1457.
11. Steiner, S., de Leon, D. C., Alves-Foss, J. (2017). A structured analysis of SQL injection runtime mitigation techniques. *Proceedings of the Annual Hawaii International Conference on System Sciences,* vol. 2017, pp. 2887–2895. Hawaii.
12. Nicolett, M. (2005). How to Develop an Effective Vulnerability Management Process. Technical Report ID Number: G00124126. Gartner. https://docplayer.net/5943695-How-to-develop-an-effective-vulnerability-management-process.html.
13. Bermejo Higuera, J. R. (2013). *Metodología de evaluación de herramientas de análisis automático de seguridad de aplicaciones web para su adaptación en el ciclo de vida de desarrollo (Ph.D. thesis)*. Universidad Nacional Educación a Distancia (UNED). http://e-spacio.uned.es/fez/eserv/tesisuned:IngInd-Jrbermejo/BERMEJO_HIGUERA_Juan_Ramon_Tesis.pdf.
14. Byrne, P. (2006). Application firewalls in a defence-in-depth design. *Network Security, 2006(9),* 9–11. DOI 10.1016/S1353-4858(06)70422-6.
15. Schmitt, I., Schinzel, S. (2012). *Waffle: Fingerprinting filter rules of web application firewalls. 6th USENIX Workshop on Offensive Technologies (WOOT 12)*, Bellevue, WA, USENIX Association.
16. Holm, H., Ekstedt, M. (2013). Estimates on the effectiveness of web application firewalls against targeted attacks. *Information Management & Computer Security, 21(4),* 250–265. DOI 10.1108/IMCS-11-2012-0064.
17. Moosa, A. (2010). Artificial neural network based web application firewall for sql injection. *International Journal of Computer, Electrical, Automation, Control and Information Engineering, 4(4),* 12–21.
18. Auxilia, M., Tamilselvan, D. (2010). Anomaly detection using negative security model in web application. *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, pp. 481–486. Krakow, Poland.
19. Applebaum, S., Gaber, T., Ahmed, A. (2021). Signature-based and machine-learning-based web application firewalls: A short survey. *Procedia Computer Science, 189,* 359–367. DOI 10.1016/j.procs.2021.05.105.

20. Weyns, D., Iftikhar, M. U., Malek, S., Andersson, J. (2012). Claims and supporting evidence for self-adaptive systems: A literature study. *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pp. 89–98. Zurich, Switzerland.
21. Yuan, E., Malek, S. (2012). A taxonomy and survey of self-protecting software systems. *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 109–118. Zurich, Switzerland.
22. Yuan, E., Esfahani, N., Malek, S. (2014). A systematic survey of self-protecting software systems. *ACM Transactions on Autonomous and Adaptive Systems, 8(4),* 17. DOI 10.1145/2555611.
23. Keromytis, A. D. (2013). Characterizing software self-healing systems. *Communications in Computer and Information Science, 374,* 22–33.
24. Lane, A. (2016). Understanding and Selecting RASP: Technology Overview. https://securosis.com/blog/understanding-and-selecting-rasp-technology-overview.
25. Haldar, V., Chandra, D., Franz, M. (2005). Dynamic taint propagation for java. *21st Annual Computer Security Applications Conference (ACSAC'05)*, vol. 2005, pp. 303–311. Tucson, Arizona, IEEE.
26. Zeller, S., Khakpour, N., Weyns, D., Deogun, D. (2020). Self-protection against business logic vulnerabilities. *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. pp. 174–180. New York, NY, USA, Association for Computing Machinery.
27. Yin, Z., Li, Z., Cao, Y. (2018). *A web application runtime application self-protection scheme against script injection attacks*. In: Sun, X., Pan, Z., Bertino, E. (Eds.), *Cloud computing and security*, pp. 566–577. Cham: Springer International Publishing.
28. Yang, W., Peng, J. (2020). Research on evm-based smart contract runtime self-protection technology framework. In: Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M. (Eds.), *Web, Artificial Intelligence and Network Applications*, pp. 617–627. Cham: Springer International Publishing.
29. Web Application Security Consortium (2022). The Web Application Security Consortium/Web Application Firewall Evaluation Criteria. http://projects.webappsec.org/w/page/13246985/WebApplicationFirewallEvaluationCriteria.
30. Williams, J. (2015). Protection from the Inside: Application Security Methodologies Compared. Technical Report. SANS Institute. https://techbeacon.com/sites/default/files/gated_asset/sans-application-security-methodologies-compared.pdf.
31. Razzaq, A., Hur, A., Shahbaz, S., Masood, M., Ahmad, H. F. (2013). Critical analysis on web application firewall solutions. *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, pp. 1–6. Mexico City, Mexico.
32. Mitre (2022). CWE-Common Weakness Enumeration. https://cwe.mitre.org/index.html.
33. Shahriar, H., Zulkernine, M. (2008). Music: Mutation-based SQL injection vulnerability checking. *The Eighth International Conference on Quality Software*, pp. 77–86. Oxford, UK.
34. Gu, S., Li, W., Zhao, X. (2011). Brute force vulnerability testing technology based on data mutation. *2011 IEEE Vehicular Technology Conference (VTC Fall)*, pp. 1–6. San Francisco, CA, USA.
35. Zhang, E., Zhang, Y. (2009). F-measure. *Encyclopedia of Database Systems, 1147,* pp. 50. DOI 10.1007/978-0-387-39940-9_483.
36. Díaz, G., Bermejo, J. R. (2013). Static analysis of source code security: Assessment of tools against samate tests. *Information and Software Technology, 55(8),* 1462–1476. DOI 10.1016/j.infsof.2013.02.005.
37. Ristic, I. (2010). *ModSecurity handbook*. UK: Feisty Duck.
38. ModSecurity Project (2022). ModSecurity Frequently Asked Questions (FAQ). https://github.com/SpiderLabs/ModSecurity/wiki/ModSecurity-Frequently-Asked-Questions-(FAQ).
39. Lemos, R. (2016). 4 fundamental application security testing tools best practices. https://techbeacon.com/-4-best-practices-application-security-testing-tools.

40. Contrast Security (2013). Better Application Vulnerability Detection with Advanced Data Tagging. https://www.contrastsecurity.com/security-influencers/better-application-vulnerability-detection-with-contrasts-advanced-data-tagging.
41. Ristic, I. (2008). Modprofiler: Defending Web Applications from 0-day Attacks. https://www.blackhat.com/presentations/bh-usa-08/Ristic_Shezaf/BH_US_08_No_More_Signatures_Ivan_Ristic_Ofer_Shezaf.pdf.

4.3. Artículo 3

Computers & Security 120 (2022) 102788

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

TC 11 Briefing Papers

A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques

Tomás Sureda Riera[a,*], Juan-Ramón Bermejo Higuera[b], Javier Bermejo Higuera[b], José-Javier Martínez Herraiz[a], Juan-Antonio Sicilia Montalvo[b]

[a] Computer Science Department, University of Alcalá, Ctra.Madrid-Barcelona, Km. 33.600 28805 Alcalá de Henares (Madrid), Spain
[b] Escuela Superior de Ingeniería y Tecnología (ESIT), Universidad Internacional de la Rioja (UNIR), Av. de la Paz, 137, Logroño (La Rioja) 26006, Spain

ARTICLE INFO

Article history:
Received 9 February 2022
Revised 23 April 2022
Accepted 3 June 2022
Available online 5 June 2022

Keywords:
Multi-label classification
Dataset
LightGBM
CatBoost
Machine learning

ABSTRACT

Context: There are many datasets for training and evaluating models to detect web attacks, labeling each request as normal or attack. Web attack protection tools must provide additional information on the type of attack detected, in a clear and simple way.

Objectives: This paper presents a new multi-label dataset for classifying web attacks based on CAPEC classification, a new way of features extraction based on ASCII values, and the evaluation of several combinations of models and algorithms.

Methods: Using a new way to extract features by computing the average of the sum of the ASCII values of each of the characters in each field that compose a web request, several combinations of algorithms (LightGBM and CatBoost) and multi-label classification models are evaluated, to provide a complete CAPEC classification of the web attacks that a system is suffering. The training and test data used for training and evaluating the models come from the new SR-BH 2020 multi-label dataset.

Results: Calculating the average of the sum of the ASCII values of the different characters that make up a web request shows its usefulness for numeric encoding and feature extraction. The new SR-BH 2020 multi-label dataset allows the training and evaluation of multi-label classification models, also allowing the CAPEC classification of the various attacks that a web system is undergoing. The combination of the two-phase model with the MultiOutputClassifier module of the scikit-learn library, together with the CatBoost algorithm shows its superiority in classifying attacks in the different criticality scenarios.

Conclusion: Experimental results indicate that the combination of machine learning algorithms and multi-phase models leads to improved prediction of web attacks. Also, the use of a multi-label dataset is suitable for training learning models that provide information about the type of attack.

1. Introduction & motivation

Every year there are significant increases in the number of attacks against web servers and applications; e-commerce platforms, financial and government institutions, large corporations, etc. are targeted by web attacks for economic or ideological reasons. According to Cisco (Cisco, 2018), 14.5 million DDoS attacks are expected in 2022. Also, SQL Injection (SQLi) and Cross-site Scripting (XSS) attacks are easy and powerful methods for attacking a web site (Johari and Sharma, 2012). The impact of cyber-attacks suffered by companies threatened their viability in 17% of cases, reported specialist insurer Hiscox(Hiscox, 2021), with their website becoming the first point of entry in 29% of cases.

Several technologies and systems exist to prevent and detect attacks on web servers and applications: *misuse detection systems* with large rules and vulnerability signature databases that must be continuously updated, *anomaly detection systems* that interpret deviations based on expected patterns of user and application behavior, taking this behavior as evidence of malicious activity. Recently, there has been a significant increase in scien-

* Corresponding author.
E-mail addresses: tomas.sureda@uah.es (T.S. Riera), juanramon.bermejo@unir.net (J.-R.B. Higuera), javier.bermejo@unir.net (J.B. Higuera), josej.martinez@uah.es (J.-J.M. Herraiz), juanantonio.sicilia@unir.net (J.-A.S. Montalvo).

https://doi.org/10.1016/j.cose.2022.102788

tific interest in anomaly detection techniques applied to web intruder detection (Sureda Riera et al., 2020). To successfully train and evaluate models based on anomaly detection techniques, specific web traffic datasets are needed; a major drawback in the study of web attack prevention and detection is the lack of public datasets to audit and validate the studies performed in this field (Sureda Riera et al., 2020); the DARPA dataset and those belonging to the KDD family have been widely criticized in different studies (Brugger, 2007; Mahoney and Chan, 2003; McHugh, 2000; Tavallaee et al., 2009). The CSIC-2010 dataset has become one of the most popular in recent years for testing protection systems against web attacks. This dataset is generated by synthetic traffic and contains 36,000 requests labeled as normal and more than 25,000 labeled as anomalous (Torrano-Gimenez et al., 2009).

Most datasets are composed of artificially generated traffic and, to our best knowledge, all datasets available for training and/or evaluation of machine learning models provide only labeling of the request in terms of normality or attack, without specifying in any case what type(s) of attack(s) is/are being suffered. The authors strongly believe in the need for datasets that collect this type of additional information and that allow the training of machine learning models that classify attacks based on internationally accepted classification criteria, such as CAPEC. In this way, by providing the incident response team with the CAPEC classification of the attack, response times and effectiveness will be improved, as the specific attack pattern and possible mitigations can be queried.

For this reason, one of the main achievements of this work is the generation of a new dataset (the SR-BH 2020 dataset) that collects different types of attacks, coming from real traffic data (generated by collecting real traffic in a honeypot exposed to the Internet for 12 days), with multi-labels, that report the normality of the request, or the CAPEC classification of the type or types of attack that the web request represents. This dataset, to our knowledge, is the first one that allows the training and evaluation of multi-label machine learning models and algorithms, which can provide the CAPEC classification of the attack(s) that a web application is suffering.

One of the fundamental stages in any data science project is the preprocessing of the data that will be used to train the machine learning models; this stage includes the selection of the relevant characteristics of the dataset that allow a high level of model efficiency to be achieved when making the prediction. In the case of a dataset with web traffic data, it is necessary to numerically encode the various fields of each web request, so that it is possible to apply different statistical techniques to the resulting numerical values. In this study, we present a new form of numerical encoding consisting of calculating the average of the sum of the ASCII values of each of the characters that make up each field of a web request.

Several supervised machine learning algorithms and techniques applied to intrusion detection have been studied over the years, most notably algorithms using ensembles of decision trees in combination with gradient boosting (Tama et al., 2020; Vu et al., 2019); two popular algorithms that provide a gradient boosting framework are LigthGBM (Ke et al., 2017) and CatBoost (Dorogush et al., 2018). In this paper, we will evaluate (using different metrics, depending on the criticality levels of several scenarios) the performance of these algorithms in predicting web attacks, such that they report the normality of the request or the CAPEC classification(s) of the attack. As this is a multi-label classification task since a single request may contain more than one type of attack, the SR-BH 2020 dataset will be used and different multi-label classification models will be combined with the LightGBM and CatBoost algorithms.

This paper makes the following contributions:

- Construction of the SR-BH 2020[1] dataset, a new multi-label dataset for Web attack detection and prediction based on CAPEC attack patterns, suitable for training multi-label classification models.
- A new way of web request data encoding using the mean of the sum of the characters ASCII values.
- Design of one-phase, two-phase, and customized classification multi-label machine learning models.
- Study of the behavior of novel algorithms applying the designed classification multi-label models.
- A ranking is obtained of the different combinations of algorithms/models to be applied in the protection of different scenarios, according to their criticality levels, using different evaluation metrics.

The rest of this work is structured as follows: Section 2 describes background and related work. Section 3 shows in an overview the process followed in this work. In Section 4, the materials and methods used are provided including the dataset used for the experiments and method evaluation, the system followed for feature extraction from the dataset, and a description of the different models applied to algorithms. Section 5, shows the model results and considerations. The conclusions of the study are provided in Section 6.

2. Background and related works

2.1. Background

This section provides an overview of the WAF and RASP web protection tools and the similarities and differences between them, a presentation of the characteristics of the algorithms analyzed in this study, namely LightGBM and CatBoost, as well as a description of the metrics and scenarios of criticality to be used to evaluate the performance of the various models and algorithms.

2.1.1. Web application protection

One of the most widely used methods for Web application protection is the implementation of Web Application Firewall (WAF) tools. A WAF is deployed between the application and the requesting user, inspecting the incoming traffic at the application layer of the OSI model and looking for attack patterns, eventually blocking incoming malicious traffic. WAF devices work by checking that incoming traffic against a database of signatures or rules so that the update of that database is critical. They are independent of the programming language of the web application as they act before the malicious traffic gets to execute the code. Different ways of circumventing the protection provided by a WAF have been proposed (Ristic, 2022), OWASP: Normalization Method, Using HTTP Parameter Pollution (HPP), Using HTTP Parameter Fragmentation (HPF), Using logical requests AND / OR, replacing with their synonyms the SQL functions that get WAF signatures, using comments, case changing, triggering a Buffer Overflow / WAF Crashing. Most of these methods focus on protocol layer exploits that attempt to take advantage of small differences between how WAF, web servers, and backend applications see traffic.

Runtime Application Self-Protection (RASP) tools are defined by Gartner Gartner (2022) as "a security technology that is built or linked into an application or application runtime environment, and is capable of controlling application execution and detecting and preventing real-time attacks". These tools combine real-time contextual awareness of the factors that have led to the application

[1] Available at https://doi.org/10.7910/DVN/OGOIXX.

of current behavior (Dubey, 2016; Steiner et al., 2017). They work via "embedding" the security in the application server to be protected, intercepting all calls to the system to check they are secure so that they depend entirely on the programming language of the application to protect.

Both technologies work in the application layer filtering the HTTP protocol. WAF is a black box technology (it does not need access to the logic of the application; it intercepts the calls and responses to the logic of the application to be protected, performing a syntactic analysis to detect attacks). On the other hand, RASP is a white-box technology that works by installing an agent on the application server, examining how variables in the application's process and code stack get their values to, based on that information, predict whether an attack is taking place.

Several attempts have been made to improve the effectiveness of WAF solutions. The model proposed by Moosa (2010) is based on an Artificial Neural Network (ANN) that defends against SQL injection attacks. While in the training phase the ANN is exposed to a set of normal and harmful data, in the working phase, the trained ANN is embedded into the WAF, thus protecting the entire web server.

In an attempt to improve the effectiveness of WAF rule sets, Auxilia and Tamilselvan (2010) propose a negative security model, which monitors applications for security anomalies, uncommon behaviors, and common web application attacks.

Taint analysis is one of the techniques used in RASP solutions. Haldar et al. Haldar et al. (2005), propose the analysis of externals to the web application data, marking them as untrusted (tainted data). They identify, monitor, and prevent the inappropriate use of this type of data at runtime in the web application to protect, developing a heuristic that instruments the java.lang.String class, to propagate taintedness of strings, as well as to mark certain strings as untainted.

Halfond et al. (2008) propose the use of *positive tainting*, identifying and tracing trusted data at execution time, while conventional tainting is centered on untrusted data. In this way, False Negatives (FN) are completely eliminated, at the cost of producing an increase in False Positives (FP). This approach also makes use of syntax-aware evaluation, so that data usage regulation is applied at development time based on its syntax in the query string, requiring only the deployment of the web application using the MetaSTrings library.

2.1.2. Algorithms

LightGBM LightGradient Boosting Machine, abbreviated as LightGBM, is an open-source library that provides a fast, decentralized, highly performant gradient boosting environment on the basis of a decision tree algorithm. Guolin Ke, et al.Ke et al. (2017), introduced two key concepts:

- Gradient-based One-Side Sampling (GOSS): A modified version of the gradient boosting method that uses only large gradient data instances. GOSS can get fairly accurate information gain estimates with a much smaller data size, which speeds up training and reduces the computational complexity of the method.
- Exclusive Feature Bundling (EFB): This approach groups dispersed (mostly null) features that are mutually exclusive from each other, thus becoming a feature selection method.

In standard decision tree algorithms, such as c4.5 (Ross Quinlan et al., 1994) and CART (Breiman et al., 1984), nodes are expanded in order of depth *(level-wise tree growth)* through of the "divide and conquer" strategy, using a prefixed order (usually from left to right).

LightGBM, on the other hand, is part of a so-called "best-first decision trees" *(leaf tree growth)* where nodes are expanded in the best order rather than in a fixed order. In this case, the best split node at each step is added to the tree. The best node is a node that is not classified as a terminal, that is, a node that minimizes the Gini impurity index among all nodes that can be split (Shi, 2007).

LightGBM uses a histogram-based algorithm. That is, group successive feature values into individual bins, build feature histograms during training, and speeds up this process and reduces memory usage. Following the leaf-wise based approach produces a much more complex tree than the hierarchical-based approach, which is a key factor in achieving higher accuracy. But sometimes this can lead to overfitting. LightGBM aims to reduce the complexity of histogram construction by using GOSS and EFB to reduce sampling data and features.

CatBoost

Yandex developed CatBoost, an open-source software library that provides an innovative categorical feature processing algorithm and a gradient boosting framework that implements *ordering boosting*, a permutation-based alternative to traditional algorithms. CatBoost uses one-time encoding to implement a symmetric tree that handles categorical features and helps reduce prediction times. LightGBM uses *GOSS* to reduce complexity, while the CatBoost algorithm introduces *Minimal Variance Sampling (MVS)*, a weighted version of stochastic gradient boosting sampling used to normalize boosting models. Using MVS reduces the number of examples required for each boosting iteration and greatly improves the quality of the model, making the model more general and less likely to overfit (Dorogush et al., 2018; Prokhorenkova et al., 2018).

Another important concept of CatBoost is the use of Oblivious Decision Trees (ODT's) in the process of building Decision Trees so that a set of ODT's is built. If an ODT has n levels, the set will have 2^n levels, since an ODT is a complete binary tree. The same splitting criterion will be applied to all nodes that are not leaves of the ODT (Hancock and Khoshgoftaar, 2020; Prokhorenkova et al., 2018). According to Prokhorenkova et al. ODTs are balanced, allow speeding up execution, and are less prone to overfitting (Prokhorenkova et al., 2018).

2.1.3. Evaluation metrics and scenarios

Most supervised classification algorithms focus on binary or multi-class classification; in this case, classical classification metrics are adequate, e.g. accuracy, F1 Score, precision, recall, etc. (Sureda Riera et al., 2020). However, when working with datasets in which there are several labels for each observation, it is necessary to complement the classical metrics since the notion of a partially correct prediction of the various labels that make up each observation is introduced (Cheng and Hüllermeier, 2009; Gouk et al., 2016; Read et al., 2011; Zhang and Zhou, 2014).

Also, according to the work of Antunes and Vieira (2015), it is necessary for the metrics to make sense of the model being evaluated and the scenario to which the model is applied; for this reason, their recommendations are followed and four different types of scenarios are defined in which each algorithm/model combination is evaluated, applying the recommended metric in each scenario.

In our case, we have chosen to perform several evaluations: first, we have calculated the Accuracy following the exact-match principle (exact prediction of all the labels); F measure, Recall, Precision, ROC AUC, Informedness, and Markedness of each of the model/algorithm combinations have been calculated on an overall basis. On the other hand, Accuracy and F measure have been evaluated for each of the labels individually, as it may be useful to determine how well the model predictions fit for each type of attack; additional metrics such as Hamming Loss, Hamming Score, and Jaccard Similarity have been introduced to evaluate the models taking into account possible partially correct model predictions.

Metrics

- **Accuracy (for each label):** This is the measure of accuracy that we will use to assess the model's efficiency in predicting each of the labels that make up an observation; in this case, we will apply the classical definition of accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Accuracy - Exact Match (EMR):** This is the measure of accuracy that we will use to evaluate the global model performance; in our case, the exact measure of the prediction for all the labels that compose an observation: We just ignore partially correct predictions (considering them incorrect) and extend the concept of Accuracy used for single label prediction to the multi-label case.

$$EMR = \frac{Number\ of\ records\ with\ exact\ label\ match}{Total\ number\ of\ records}$$

- **Precision:** In our case, to account for label imbalance, we computed the proportion between the correct predicted labels and the total labels averaged over all cases and weighted them by support (the total number of cases for each label).

$$Precision = \frac{TP}{TP + FP}$$

$$Precision_{weighted} = \sum_{classes} weight\ of\ class\ x\ precision\ of\ class$$

- **Recall:** This is the true positive percentage correctly identified, averaged across all instances and support-weighted.

$$Recall = \frac{TP}{TP + FN}$$

$$Recall_{weighted} = \sum_{classes} weight\ of\ class\ x\ recall\ of\ class$$

- **F measure (for each label):** It is a measure of the accuracy of a test, taking into account both precision and recall. Also known as F1-Score or F-Score, this value is a balanced harmonic mean of two metrics: Precision (P) and Recall (R) Bermejo Higuera (2013); Díaz and Bermejo (2013); Van Rijsbergen (1979).

$$F\ measure = \frac{2\ x\ precision\ x\ recall}{precision + recall}$$

- **F measure (averaged):** F measure score averaged across all instances and support-weighted.

$$F\ measure_{weighted} = \sum_{classes} weight\ of\ class\ x\ F\ measure\ of\ class$$

- **Informedness:** Is a measure of how much information the system provides about positive and negative labels,i.e. how informed a predictor is for the desired condition, as opposed to chance, averaged across all instances and support-weighted.

$$Informedness = \frac{TP}{FN + TP} + \frac{TN}{FP + TN} - 1 = \frac{TP}{FN + TP} - \frac{FP}{FP + TN}$$

$$Informedness_{weighted} = \sum_{classes} weight\ of\ class\ x\ Informedness\ of\ class$$

- **Markedness:** A measure of the confidence in the system's positive and negative predictions, it quantifies how consistently the outcome includes a predictor variable as a marker, that is, how the labeled condition for a given predictor compares to chance, averaged across all instances and support-weighted.

$$Markedness = \frac{TP}{TP + FP} + \frac{TN}{FN + TN} - 1 = \frac{TP}{TP + FP} - \frac{FN}{FN + TN}$$

$$Markedness_{weighted} = \sum_{classes} weight\ of\ class\ x\ Markedness\ of\ class$$

- **ROC AUC:** Shows the global efficiency of a classification model at all classification levels, by plotting the true positive rate (TPR) versus the false positive rate (FPR). The AUC is a bidimensional metric of the area under the full ROC curve, which ranges from 0 (100% inaccurate predictions) to 1 (100% accurate predictions), reflecting the separability grades and showing the capacity of a particular model to distinguish among classes (Sureda Riera et al., 2020; Swets, 1996). In our case, ROC AUC is averaged across all instances and support-weighted.
- **Hammming Loss:** Is the proportion of incorrectly predicted labels over the total number of labels. In multi-label classification, the Hamming loss is calculated as the Hamming distance between the true and the predicted values. Its value ranges from 0 to 1. The lower the value, the better the performance of the model.Let D be a multi-label dataset, consisting of $|D|$ multi-label observations (x_i, Y_i), i = 1.|D|, $Y_i \subseteq L$. Let H be a multi-label classifier and $Z_i = H(x_i)$ be the set of labels predicted by H for observation x_i.

$$Hamming\ Loss\ (H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|}$$

  Where Δ represents the symmetric difference between the two sets (Schapire and Singer, 2000; Tsoumakas and Katakis, 2009).
- **Jaccard Similarity:** It measures the degree of similarity between two sets by examining the proportion of correctly predicted positive labels in a potentially positive set (expected positive and real positive).

$$\mathcal{J}(T, P) = \frac{T \cap P}{T \cup P}$$

  where T and P are true labels and predicted labels respectively.

Scenarios

Four different scenarios are defined, based on the classification made by Antunes and Vieira (2015):

- **Very high critical scenario:** Represents the development and evaluation of critical applications that have very high-security requirements because they need to provide their customers with a reliable system. Examples of this scenario are internet bank websites, trade-in equity shares, or massive electronic commerce systems. The top priority in this scenario is the **elimination of the largest number of attacks**, thus assuming the investment of time and resources in remediating the occurrence of non-existent attacks (false positives). In this case, the metric of choice is **recall**, as it maximizes attack detection.
- **Heightened-critical scenario:** In this case, the goal is to achieve a **balance between the priority of detecting and eliminating the maximum number of attacks and preventing the excessive reporting of false positives**, since resources in this type of scenario must be properly managed. Security requirements are high, but lower than Very high critical scenario. Examples of this scenario are government portals, e-commerce web applications, etc. A metric of choice in this scenario could

be the F-measure, but the problem is that it assigns equal importance to precision and recall; **Informedness** appears to be a better alternative, as it is bias-free and does not have the disadvantages of harmonic averaging.

- **Medium-critical scenario:** This scenario features less exposed or less criticals applications, typically with a limited budget, so the resources available for remediation of reported attacks are limited. For this reason, **both finding and eliminating as many attacks as possible and saving resources on remediation of false positives have equal importance**. Examples of this type of scenario are web sites, where attacks result in lower financial losses, or intranet applications that are less prone to external attacks. The optimal solution for this case is to use **F-Measure**.
- **Low-critical scenario:** This scenario represents non-critical applications that are not very exposed to attacks. They are characterized by being implemented with small budgets, so the resources available are limited; the use of these resources should be focused on confirmed attacks. The goal is to **report as few false positives as possible**, increasing confidence in reported attacks. Examples of such scenarios are web portals for small and medium-sized companies. **Markedness** seems to be the best metric. Actually, it is able to give greater preponderance to the accuracy considerations, while at the same time being capable of considering the attacks that are left unnotified.

2.2. Related work

This section details the state of the art related to studies on dataset generation, pattern extraction and feature selection techniques, attack classification and deep learning models.

2.2.1. Datasets

Many datasets have been proposed for the study and evaluation of models and techniques to enable web attack prediction:

- DARPA: The Lincoln Laboratory of the Massachusetts Institute of Technology (MIT) created this dataset in 1998 and updated it in 1999. It consists of about 5 million connection records, which contain raw tcpdump data. It has 244 tagged cases of 58 attacks on four different OS (Lichman, 2000; Tavallaee et al., 2009).
- KDD Cup 99: This is a variant of the DARPA dataset with about 4,900,000 and 2,000,000 entries in the training and test datasets, each of which contains 41 features and is appropriate for training several machine learning algorithms. It may produce inconsistent results, due to the presence of duplicate records (Tavallaee et al., 2009).
- NSL-KDD: In an attempt to deal with one of the problems with the KDD Cup 99 dataset, a series of cleanup operations were performed on the duplicate records. From this processing, the new NSL-KDD dataset was created, which has 175,341 and 82,332 records in the training and test sets, respectively (Devi and Abualkibash, 2019).
- UNSW-NB15: This is a mixed dataset of real normal activities and synthetically generated behaviors of modern attacks, with 47 features and 2 class-labeled features (Moustafa and Slay, 2015).
- Kyoto 2006: This is an exhaustive and representative dataset generated from data obtained from real-time traffic. It has 24 features, 14 of which were retrieved from KDD Cup '99, plus 10 additional ones, but not including those features that contain duplicate records (Protić, 2018).
- ISCX: Emerged from the work of Shiravi et al. Shiravi et al. (2012), this dataset was built by generating simulated traffic for seven days. It contains 11 features and, in addition to the requests being labeled as normal or attack, provides a description of the network traffic.
- CSIC-2010: Based on the work of Torrano-Gimenez, Perez-Villegas and Alvarez Torrano-Gimenez et al. (2009), this dataset was produced at the Consejo Superior de Investigaciones Científicas (CSIC). Originated from simulated web requests to an e-commerce web application, this dataset is composed of 36,000 normal requests and over 25,000 anomalous ones, marked as either normal or anomalous.
- ECML/PKDD 2007: Generated from real traffic and replacing parameter values and names with random values in order to anonymize the data, this dataset includes 35,006 normal records and 15,110 records labeled as attacks (Raïssi et al., 2007).

Although the University of California, specifically the archival authority of the KDD Archive in Irvine discourages its use Brugger (2007); Tavallaee et al. (2009), as well as is considered inadequate and obsolete Mahoney and Chan (2003); McHugh (2000), the KDDD family (DARPA, KDD CUP 99, and NSL_KDD), is widely used in current intrusion detection system evaluation studies (Siddique et al., 2019). With the generation of the new SR-BH 2020 dataset, the state of the art is improved by providing the first dataset, derived from real web traffic data, specifically designed for the training of multi-label web attack prevention and detection models.

2.2.2. Pattern extraction and features selection

Krügel et al. (2002) use a model of distribution of characters to characterize the traffic genuinely generated to the web application. In this work, in contrast to the previous one, a numerical value to each field of a web request is assigned, based on the ASCII value corresponding to each character, in order to train models that can predict the normality or malignity of a web request, as well as the corresponding CAPEC key.

In Kruegel and Vigna (2003), Kruegel and Vigna introduce an anomaly detection system that uses web server log files as input to produce an anomaly score for every web request based on the length and character distribution of the attributes. In the present work, the anomaly score is calculated for each field of interest of a web request extracted from the ModSecurity log.

Kozik et al. (2015) detect anomalies in HTTP traffic, using a pattern extraction method derived from the distribution character model suggested by Kruegel, Toth, and Kirda as well as token detection of a web request, using text segmentation. Our work takes advantage of the log generated by ModSecurity and returns a numeric value for each field of interest.

Resende and Drummond (2018) select characteristics and profile parameters for intrusion detection methods based on anomaly, using an adaptational approach that relies on a genetic algorithm. In our work, features are selected by generating a histogram based on the mean ASCII value obtained by each field of a web request.

In Tan and Hoai (2021), Tan and Hoai propose the HQTN technique that transforms the HTTP request into numeric, focusing on attributes names and values, and query strings and using the CityHash hash function, testing their approach on the CSIC 2010 dataset. In our approach, we transform the full web request to a numeric value by calculating the mean value of the sum of the ASCII values of all the web request fields, which in principle is much easier and gives good results. In addition, they work with a binary label dataset (CSIC 2010) and we work with the SR-BH 2010 dataset which is specific for multi-label classification.

2.2.3. Attacks classification

Dang and François (2018) use relationship inference between various cybersecurity issue repositories: *CAPEC (Common Attack Pattern Enumeration and Classification), CWE (Common Weakness Enumeration), and CVE (Common Vulnerabilities and*

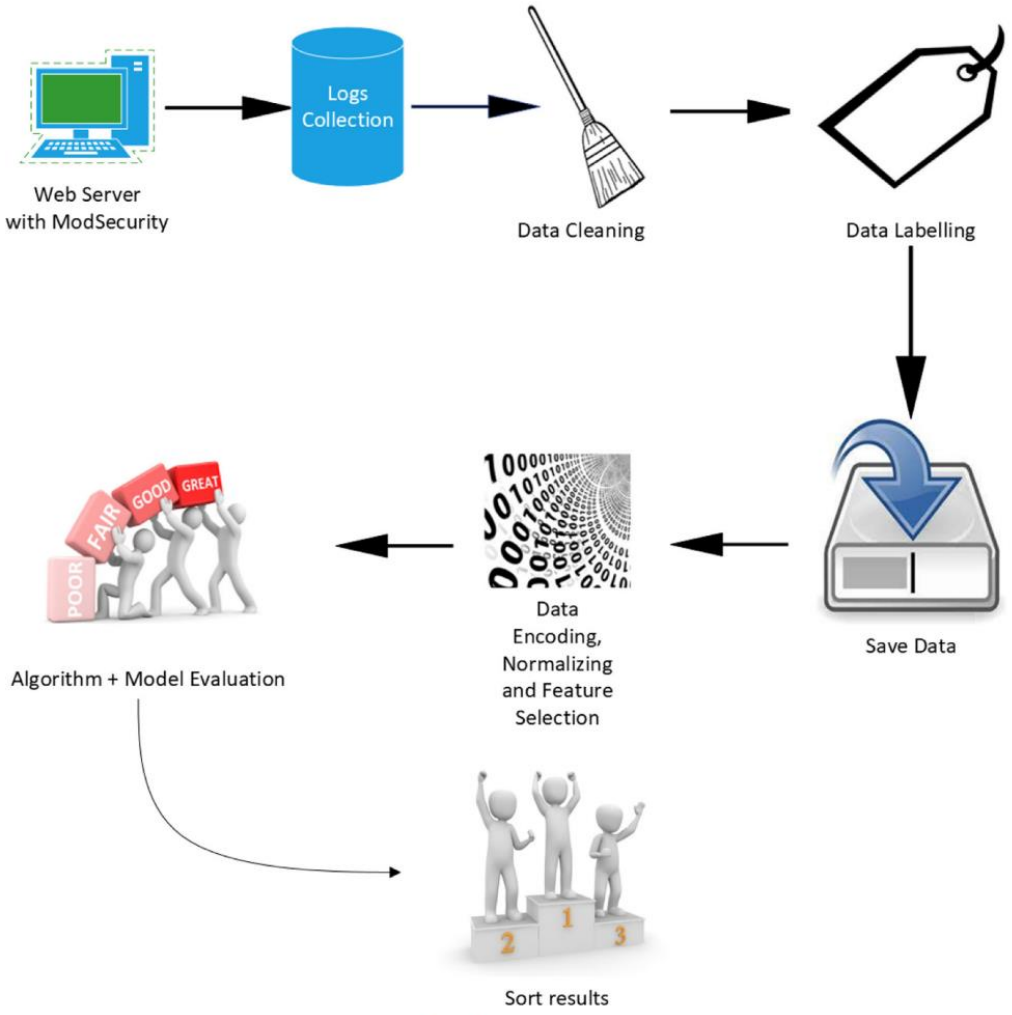


Fig. 1. Process overview.

Exposures) to detect patterns of attacks and weak points in the software associated with SDN/NFV software vulnerabilities. Kanakogi et al. (2021) use three different measures of similarity: TF-IDF, Universal Sentence Encoder (USE), and Sentence-BERT (SBERT) to track correlated CAPEC-IDs based on CVE-IDs. The algorithm/model combinations proposed in our work, return information to the security operator on the CAPEC classifications of the attack the system is undergoing.

2.2.4. Deep learning models

Mac et al. (2018) detect harmful patterns in the HTTP/HTTPS traffic, using an autoencoder. They worked on the raw web request, collecting the absolute path, the method, and the query parameters thanks to a preprocessing of the data by tokenizing the URL, replacing the characters with their corresponding ASCII code. In the present work, a numeric value based on the mean value of the sum of the ASCII score of all the characters that are part of a field in a web request is calculated.

Liang et al. (2017) propose the use of an Autoencoder and a Recurrent Neural Network (RNN) to detect anomalous requests on web servers, tokenizing the URLs to reduce their variability, while Jin et al. (2018) apply AutoEncoder and RNN to identify web attacks based on payloads. Instead of using neural networks, various combinations of models and novel algorithms (LightGBM and CatBoost) in the field of machine learning are evaluated.

Pan et al. (2019) detect runtime intrusions by mining call traces in web applications and learning the correct program execution model through a stacked denoising autoencoder; they call their model Robust Software Modeling Tool (RSMT). Truong et al. (2019) propose detecting anomalous HTTP queries using Sum Rule and Xgboost and combining the related results with several stacked denoising autoencoders (SDAE). In our work, we do not use deep learning techniques, but train models of different phases, using LightGBM and CatBoost.

Tama et al. propose an architecture of stacked ensembles where its base learners are other ensembles learners Tama et al. (2020). Tekerek proposes an anomaly-based Web attack detection architecture that relies on Convolution Neural Network (CNN) (Tekerek, 2021). Our work proposes a two-phase model architecture with CatBoost algorithm.

Montes et al. (2021) use deep learning techniques to enhance Modsecurity performance using a two-phase model: first, using

Table 1
Number of web requests by CAPEC classification.

CAPEC Classification	Number of web requests	% of total requests
000 - Normal	525,195	57.85%
272 - Protocol Manipulation	9153	1.00%
242 - Code Injection	15,827	1.74%
88 - OS Command Injection	7482	0.82%
126 - Path Traversal	20,992	2.31%
66 - SQL Injection	250,311	27.57%
16 - Dictionary-based Password Attack	1847	0.20%
310 - Scanning for vulnerable software	2718	0.30%
153 - Input Data Manipulation	2272	0.25%
274 - HTTP Verb Tampering	5437	0.60%
194 - Fake the source of data	56,145	6.18%
34 - HTTP Response Splitting	19,738	2.17%
33 - HTTP Request Smuggling	1059	0.12%
TOTAL	918,176	

deep learning technology, features are extracted; in a second phase, http requests are treated as raw text to train a one-class supervised model. Our work uses a numerical abnormality value of the fields of interest of a web request to train a two-phase model combined with the CatBoost algorithm.

Oliveira et al. (2021), propose a *multi-class* classification to inform the attack type, comparing a Random Forest (RF), a Multi-Layer Perceptron (ML), and a Long-Short Term Memory (LSTM) algorithm. Although their proposal obtains very good results, it is important to consider that the fact of working with a multi-class classification implies that a request can only be classified under a single type of attack. Our work allows us to properly classify web requests that involve more than one type of simultaneous attack.

Zhang and Zhou (2007), based on the K-nearest neighbor (KNN) algorithm, develop a lazy multi-label learning algorithm. Madjarov et al. (2012) used 11 benchmark data sets on which they experimentally compared 12 multi-label learning methods by using 16 metrics. Zhang and Zhou (2014) review different multi-label learning algorithms and detail different evaluation metrics. Read et al. (2011) perform multi-label classification using a chain of classifiers. Büyükçakir et al. (2018) propose an online stacked ensemble for multi-label stream classification. Wang et al. (2020) propose a collaboration-based multi-label model for e-commerce fraud detection.

3. Process overview

The process followed to complete this work can be summarized as follows:

- First, ModSecurity for Apache with Core Rule Set (CRS) version 3.3.0 is installed on a web server exposed to the Internet. ModSecurity is configured in "Detection Only" mode.
- During the exposure period, the logs generated by the ModSecurity activity are collected on a daily basis.
- The logs are reviewed and cleaned manually and semi-automatically. The correct labeling of each log made by ModSecurity is verified, thus ensuring a proper CAPEC classification.
- The reviewed logs are saved in CSV format, resulting in the SR-BH 2020 dataset.
- Each of the input fields of the dataset is numerically encoded by calculating the mean value of the sum of the ASCII code assignment of each of the characters in each field.
- The values obtained are normalized and a selection of the relevant features of the dataset is made.
- The performance of different combinations of algorithms and multi-label classification models in predicting the CAPEC classification is evaluated.

Table 2
Number of different CAPEC classifications assigned to a web request.

Number of different CAPEC classification	Number of web requests
1	898,576
2	8132
3	1088
4	18

- The results obtained are tabulated and sorted by different metrics according to the level of criticality of the different scenarios chosen.

A graphical overview of the process can be seen in Fig. 1.

4. Materials and methods

4.1. Dataset description

In this study, our new SR-BH 2020 dataset has been developed and implemented to experiment and evaluate the different algorithms and models. The dataset is composed of web requests collected during 12 days of July 2020 by a web server (Wordpress) installed on a virtual machine and exposed to Internet. On this server, Modsecurity version 2.9.2 for Apache, with Core Rule Set (CRS) version 3.3.0 was installed in *"Detection only"* mode, so that all requests (legitimate and malicious) were recorded in the log generated by ModSecurity, but without being blocked. Daily, the logs generated by ModSecurity were collected and the virtual machine was restored to a clean state.

Once the web server exposure period was over, the collected logs were manually and semi-automatically processed by one of the authors to review the web request tagging performed by Modsecurity, correcting where necessary the normal/attack assignment to the corresponding web request and ensuring an appropriate CAPEC classification assignment.

The final result is a multi-label dataset aimed especially at web attack detection and composed of 907,814 requests of which 525,195 are normal requests and 382,619 are anomalous requests, where each record has 24 different features and a set of 13 labels. See Table 1 for detailed information on the number of times a web request is classified under a given CAPEC heading. Note that the sum total of the number of CAPEC classifications in Table 1 is greater than the number of web requests present in the dataset, due to the fact that there are web requests in which more than one type of attack is combined. See Table 2 for details of the number of web requests with multiple CAPEC classifications.

In order to protect the personal data of users accessing the web server, the environment was configured so that all web re-

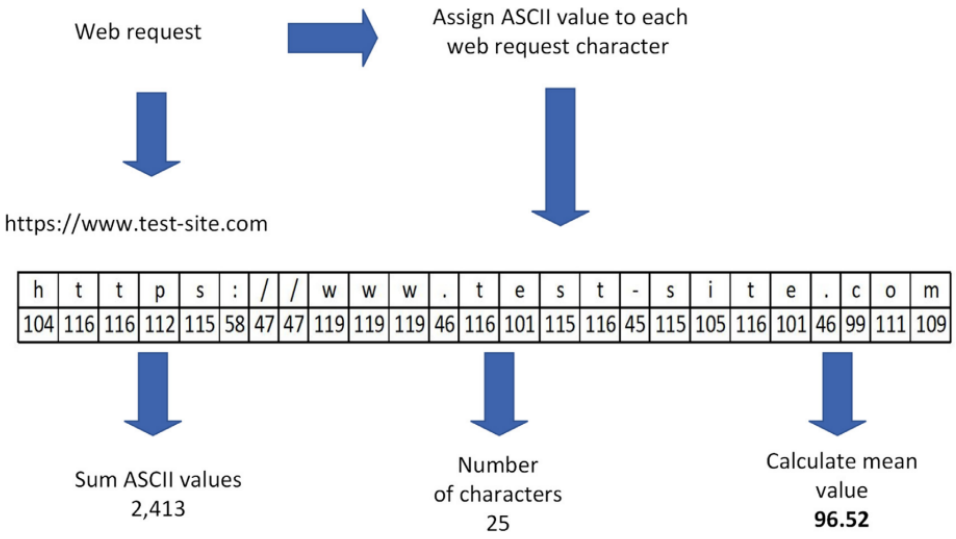


Fig. 2. Example of a web request numerical coding.

quests pass through a router interposed between the web server and the Internet connection: in this way, all requests received by the web server seem to be originated from the local IP address of the router.

4.2. Preprocessing the data, numerical transformation and features selection

Before performing the training of the different machine learning models, a review and preprocessing of the dataset data is necessary to avoid inconsistent and/or duplicated data, error correction and, at the same time, to adapt the data for numerical coding so that they are usable for the machine learning models.

The main objective of the data preprocessing and selection of relevant features of the dataset is to allow the different combinations of algorithms and models evaluated to reach the maximum level of performance and efficiency in their predictions, in addition to reducing the computational cost of modeling.

The numerical transformation of the dataset data was carried out using a procedure inspired by the work of Kozik et al. (2015), Krügel et al. (2002) and Kruegel and Vigna (2003). In our case, the mean of the sum of the ASCII values of the characters (applying a transformation to lowercase) of each of the fields of each web request is calculated: in this way, those fields with a high presence of anomalous characters (such as "./../.", present in a typical "path traversal" attack attempt) will obtain different mean ASCII values than those fields where the web request made is normal. The detailed procedure is provided by Algorithm 1. See Fig. 2 for an example of the proposed numerical coding.

In Table 3, the mean values of different fields of a normal web request and one labeled as a combination of "Protocol Manipulation" and "OS Command Injection" attacks are compared. In Table 4, shows the detail of the fields of the web request in which there are differences in the mean sum of their ASCII values.

Feature selection involves selecting those input variables that have the strongest relationship with the targets variables. Once the mean ASCII values of each feature have been calculated, a histogram is generated for each field in such a way that it is possible to determine and eliminate those characteristics that do not provide differential information. As can be seen in graph A of Fig. 3,

Table 3
Mean field ASCII values of normal and attack web request.

Field	Mean value normal request	Mean value attack request	Difference
method_value	106.667	113.5	Yes
http_request_value	97.1	94.7	Yes
protocol_value	79.875	79.875	No
referer_value	105.667	105.667	No
agent_value	73.9618	98.4444	Yes
host_value	99.6154	48.5556	Yes
origin_value	105.667	105.667	No
cookie_value	105.667	105.667	No
content_type_value	105.667	100.848	Yes
accept_value	43.6667	43.6667	No
accept_language_value	105.667	105.667	No
accept_encoding_value	95.6154	95.6154	No
do_not_track_value	0	0	No
connection_value	99.5	99.5	No
body_value	105.667	92.7102	Yes
response_http_protocol_value	79.875	79.875	No
http_status_code_value	200	404	Yes
http_status_message_value	109	101	Yes
response_content_length_value	51.6667	51.3333	Yes

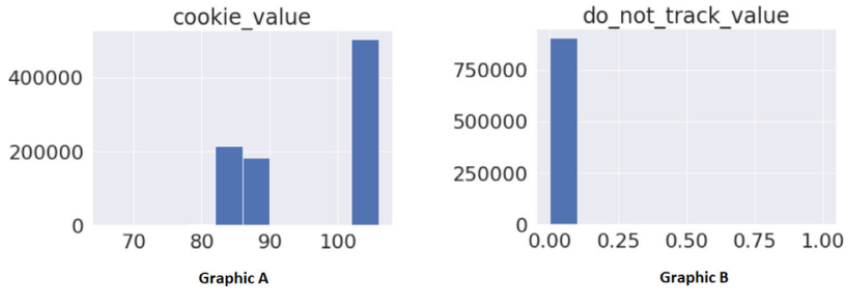


Fig. 3. Histogram of features informational levels.

Data: A field of a web request F
Result: Decimal value D
$L \leftarrow$ *length of* F;
if $L = 0$ **then**
  $D \leftarrow 0$;
else
  $v \leftarrow 0$;
  while *there is data to read in* F **do**
    $c \leftarrow$ read a character;
    $c \leftarrow$ convert c to lowercase;
    $c \leftarrow$ calculate c ASCII value;
    $v \leftarrow v + c$;
  end
  $D \leftarrow v/L$;
end

Algorithm 1: Calculation of the mean of the sum of character ASCII values in a field.

the cookie_value feature provides useful information to allow the differentiation of web requests, since its numerical values are distributed in the 80–90 and 100–110 ranges. However, in graphic B it can be seen that do_not_track_value feature does not provide any useful information since all web requests have the same value.

Once the features that do not provide useful information have been removed, an approximation to the normal distribution of each remaining feature is made by applying the natural logarithm to each value of the remaining features, subsequently standardizing their values using the StandardScaler class of the Scikit-learn library Pedregosa et al. (2011).

Finally, using the random forest classification algorithm and the RFECV class Guyon et al. (2002) of the scikit-learn library Pedregosa et al. (2011), we performed recursive feature removal using cross-validation and selected the final number of features.

When working with a dataset composed of real data, it is common to have imbalanced classes; this different percentage of representation of the classes in a dataset can affect the different evaluation metrics of the learning models. Although there are several methods to generate synthetic data that promote the equal representation of the different classes in a dataset, such as SMOTE Chawla et al. (2002) and MLSMOTE Charte et al. (2015), we have chosen to work only with real data and evaluate the algorithms and models with specific metrics that take in consideration the different percentage of representation of the classes in the dataset: Accuracy, Precision, Recall, F-Score, Hamming Loss, Hamming Score, Jaccard Similarity and ROC AUC. The selected features and labels are detailed in Table 5.

4.3. Models

The SR-BH 2020 dataset has a set of 13 labels. First label indicates whether the web request is considered normal or not, so it is assumed that if its value is 1 (normal request), the rest of the labels of the set should be 0. On the contrary, if the value of the first label is 0 (possible attack), there should be one or more of the labels of the remaining set with its value at 1.

This assumption allows us to establish a division of classifications by phases: Classifications can be established with a single-phase model, in which an attempt is made to predict the entire set of labels independently of the value obtained by the first label, i.e., even if the first label indicates that the request is normal, an attempt will be made to predict the remaining labels in the set. On the other hand, it is possible to generate two-phase prediction models in which the algorithm will only predict the rest of the tags

Table 4
Detail of fields with different mean ASCII values.

Field	Normal web request	Attack web request
method_value	GET	POST
http_request_value	/blog/xmlrpc.php?rsd	/cgi-bin/ViewLog.asp
agent_value	Mozilla/4.0 (compatible;...)	B4ckdoor-owned-you
host_value	test-site.com	127.0.0.1
content_type_value	nan	application/x-www-form-urlencoded
body_value	nan	remote_submit_Flag=1& remote_syslog_Flag=1& RemoteSyslogSupported=1&LogFlag=0& remote_host=%3bcd+/tmp;wget+ http://45.95.168.230/ taevimncorufglbzhwxqpdkjs/Meth.arm7 ;chmod+777+Meth
http_status_code_value	200	404
http_status_message_value	OK	Not Found
response_content_length_value	317	271

Table 5
Final set of selected features and labels.

Feature	Selected
method_value	Yes
http_request_value	Yes
protocol_value	**No**
referer_value	Yes
agent_value	Yes
host_value	**No**
origin_value	**No**
cookie_value	Yes
content_type_value	**No**
accept_value	Yes
accept_language_value	**No**
accept_encoding_value	**No**
do_not_track_value	**No**
connection_value	**No**
body_value	Yes
response_http_protocol_value	**No**
http_status_code_value	**No**
http_status_message_value	Yes
response_content_length_value	Yes
000 - Normal	Yes
272 - Protocol Manipulation	Yes
242 - Code Injection	Yes
88 - OS Command Injection	Yes
126 - Path Traversal	Yes
66 - SQL Injection	Yes
16 - Dictionary-based Password Attack	Yes
310 - Scanning for Vulnerable Software	Yes
153 - Input Data Manipulation	Yes
274 - HTTP Verb Tampering	Yes
194 - Fake the Source of Data	Yes
34 - HTTP Response Splitting	Yes
33 - HTTP Request Smuggling	Yes

if the value of the first tag is 0; if its value is 1 (normal request), it will automatically set all the remaining tags in the set to 0.

A customized model is also generated in which the best hyperparameters for the classification of each of the labels are calculated using GridSearchCV with the LightGBM and CatBoost algorithms. The prediction of each label will be performed with the corresponding algorithm adjusted with the calculated hyperparameters.

In order to obtain a model that provides the best possible results in predicting the normality of the web request, or the resulting CAPEC classification in the case of a web attack, the two algorithms (LightGBM and CatBoost) have been evaluated, using five different models:

i A single-phase model, using the *skmultilearn.problem_transform.BinaryRelevance* class of the *scikit-multilearn* library Szymański and Kajdanowicz (2017), transforming a multi-label classification problem with L labels into L individual binary classification problems with single labels. The prediction result is the union of all label classifiers. Binary Relevance ignores label correlation that exists in the training data, i.e. it assumes independence between labels; due to this loss of information, it would be possible for the prediction sets to contain combinations of labels that never occur in reality.

ii A single-phase model, using the *skmultilearn.problem_transform.ClassifierChain* class of the *scikit-multilearn* library Szymański and Kajdanowicz (2017). This class constructs a conditional sequence of Bayesian label classifiers, as described by Read Read et al. (2009), by generating label classifiers and sorting them into a sequence according to the *Bayesian Chain Rule*. In this case, a chain of binary classifiers is formed in which each classifier in the chain is responsible for learning and predicting the binary association of the label given the feature space, modified by all previous binary predictions on the labels in the chain.

iii A two-phase model in which, in the first phase, it detects the request is normal or not and, in the case of an anomalous request, it passes to the second phase of the model to obtain its CAPEC classification using the *BinaryRelevance* class of the *scikit-multilearn* library (Szymański and Kajdanowicz, 2017)

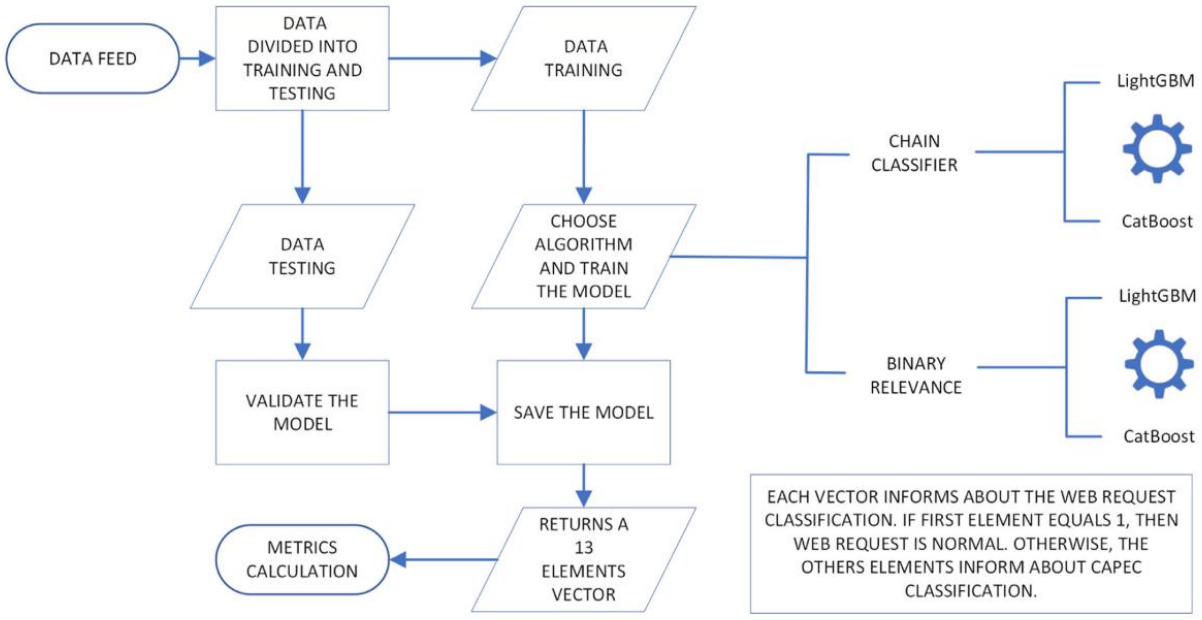


Fig. 4. Single-phase model flowchart.

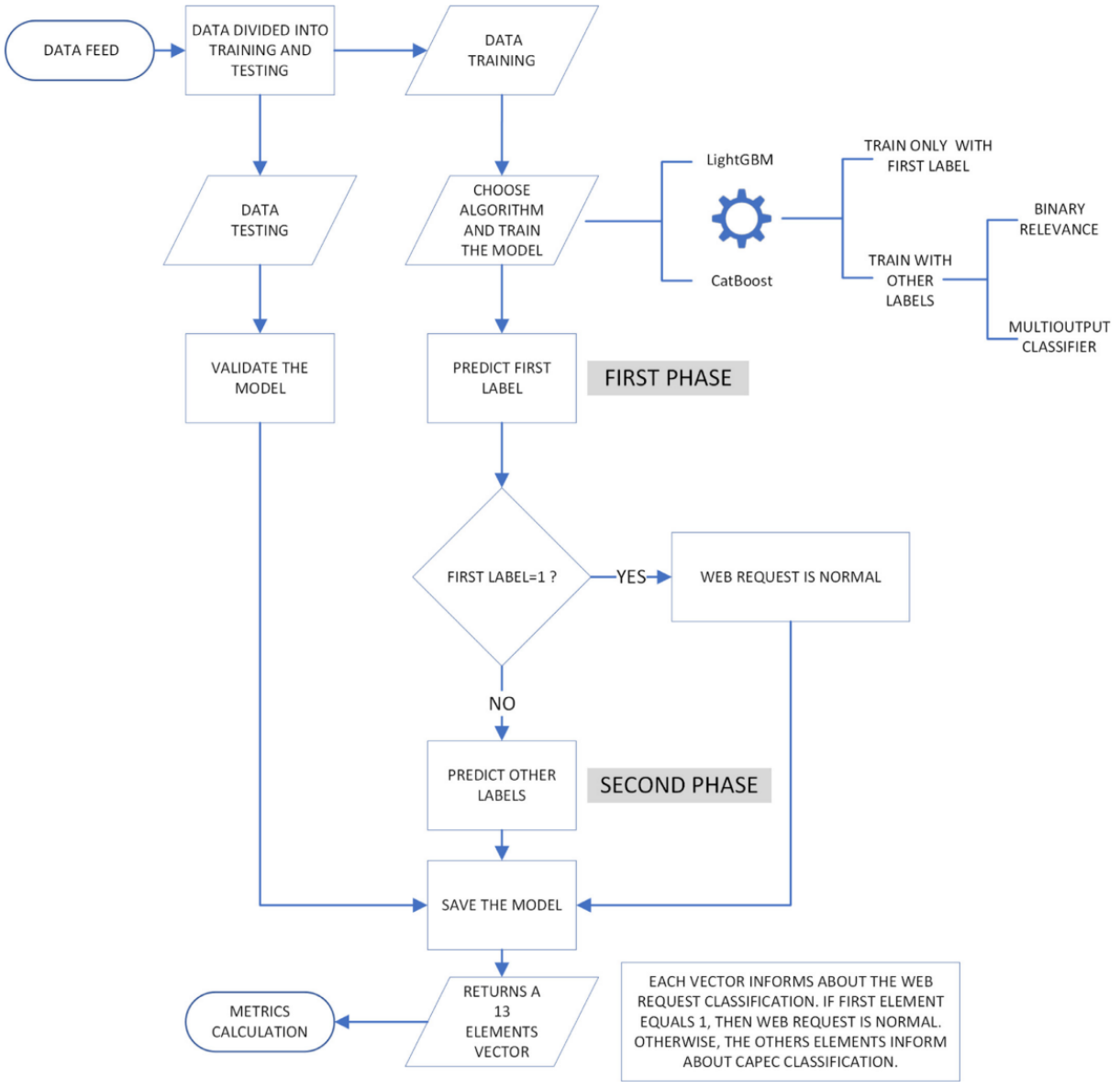


Fig. 5. Two-phase model flowchart.

iv A two-phase model in which, in the first phase, it detects the request is normal or not and, in the case of an anomalous request, it passes to the second phase of the model to obtain its CAPEC classification using the *MultiOutputClassifier* class of the *sklearn.multioutput* module of Scikit-learn library (Pedregosa et al., 2011).

v A customized model in which, the prediction is performed through the specific algorithm(LigthGBM or CatBoost) tuned with the hyperparameters calculated for each CAPEC label.

Figs. 4, 5, and 6 show the flowcharts for the single-phase, two-phase, and customized models respectively.

The combination of models designed and algorithms allows the presentation to the security specialist of information about the attack(s) the system is suffering, including the CAPEC classification(s), as shown in Fig. 7.

5. Results and discussion

The results of the algorithms and model combinations have been evaluated according to the metrics and scenarios discussed in Section 2.1.3. One of the main objectives of the present work is to propose the best combination of algorithms and models able to classify as accurately as possible the different types of attacks in each possible scenario, following the CAPEC classification. Since a web request can be simultaneously classified in more than one CAPEC key, it is necessary to work with multi-label models. The evaluation has been carried out with data from the SR-BH 2020

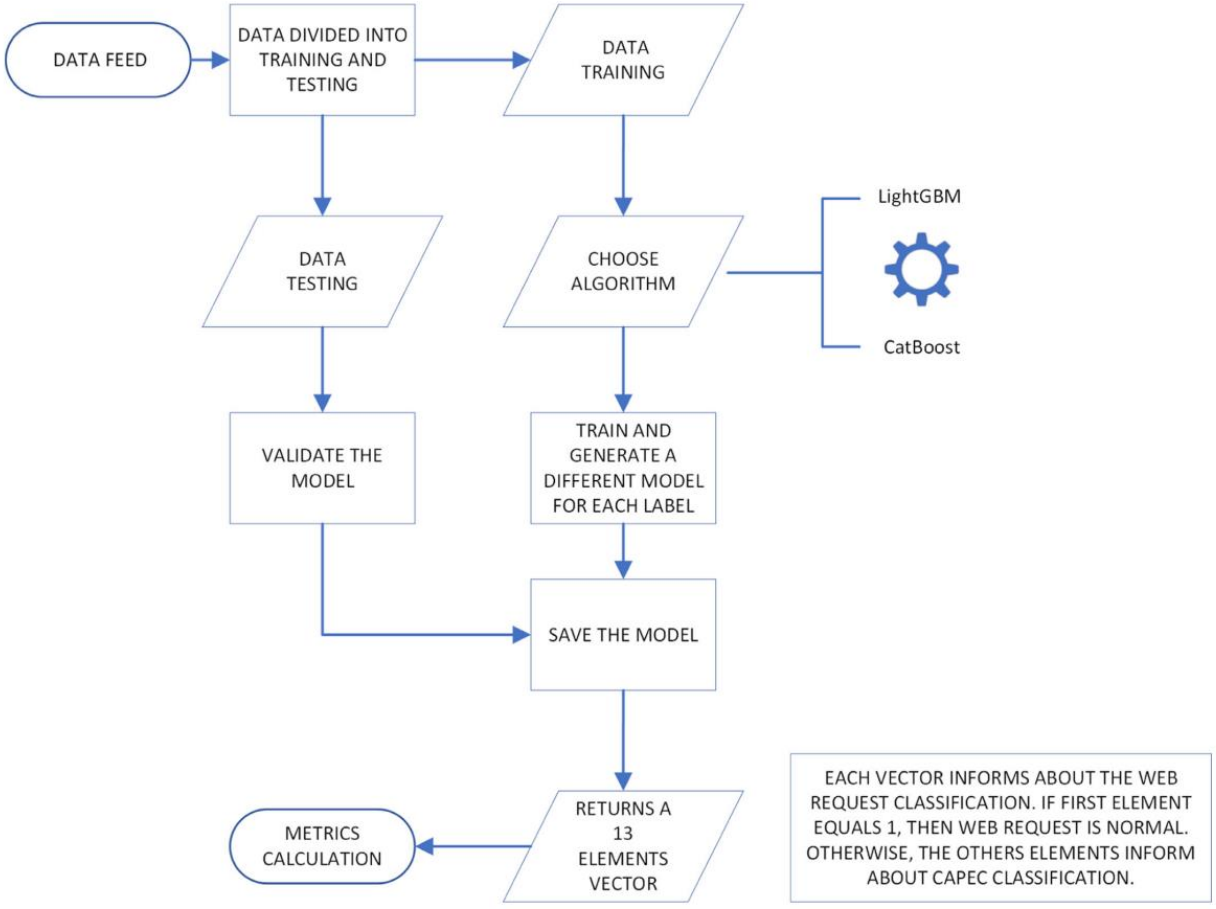


Fig. 6. Customized model flowchart.

```
In [18]: %%time
prediccion([[5.30460883087438,
 1.043082871047808,
 0.623192182505229,
 5.216494102681128,
 0.903506237388453,
 -1.2130669975644175,
 -5.68336728338642,
 -0.9974627487855351,
 -0.5428089419695353]])

CPU times: user 68.6 ms, sys: 0 ns, total: 68.6 ms
Wall time: 21 ms

Out[18]: ([0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 ['272 - Protocol Manipulation', '88 - OS Command Injection'])
```

Fig. 7. Information on attacks, received by the security operator.

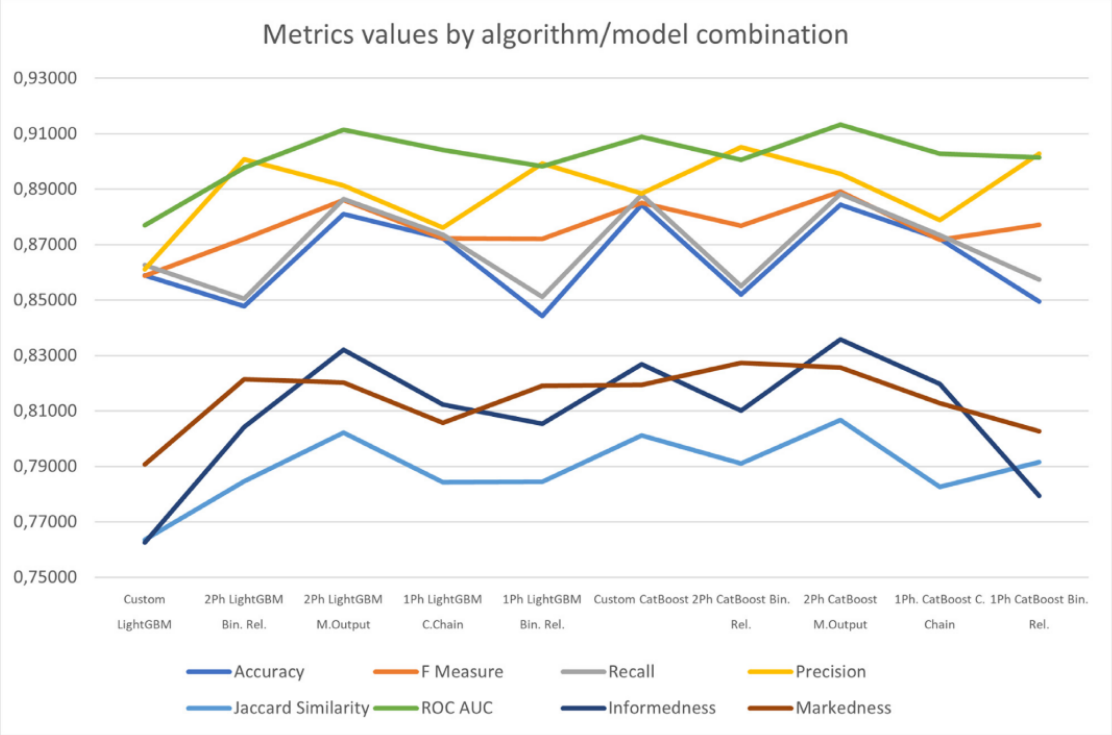


Fig. 8. Metrics by algorithm/model.

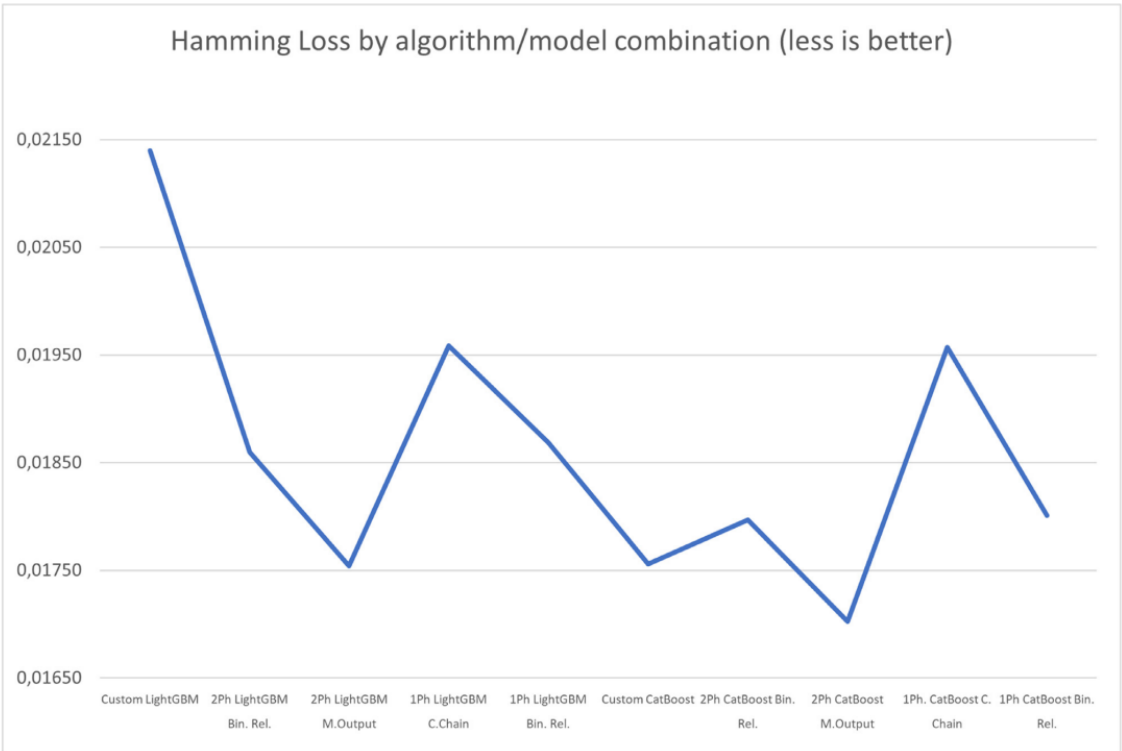


Fig. 9. Hamming Loss by algorithm/model.

Table 6
Summary of metrics by algorithm and model, ordered by recall score.

Model	Accuracy EMR	Precision	Recall	F meas. Avg.	ROC AUC	Hamming Loss	Jaccard Sim.	Inform.	Marked.
Two-phase MultiOutput CatBoost	**0.88445**	0.89557	**0.88829**	**0.88912**	**0.91322**	**0.01703**	**0.80672**	**0.83580**	0.82563
Customized model CatBoost	0.88436	0.88853	0.88790	0.88501	0.90887	0.01756	0.80115	0.82690	0.81941
Two-phase MultiOutput LightGBM	0.88095	0.89137	0.88641	0.88615	0.91139	0.01754	0.80214	0.83200	0.82028
Single-phase Clas.Chain LightGBM	0.87224	0.87610	0.87360	0.87227	0.90421	0.01958	0.78432	0.81238	0.80567
Single-phase Clas.Chain CatBoost	0.87213	0.87876	0.87343	0.87171	0.90281	0.01957	0.78264	0.81973	0.81282
Customized model LightGBM	0.85888	0.86108	0.86270	0.85874	0.87702	0.02140	0.76356	0.76248	0.79066
Single-phase Binary Relevance CatBoost	0.84939	0.90279	0.85734	0.87221	0.90139	0.01801	0.79153	0.77943	0.80261
Two-phase Binary Relevance CatBoost	0.85201	**0.90515**	0.85508	0.87680	0.90055	0.01797	0.79100	0.81007	**0.82741**
Single-phase Binary Relevance LightGBM	0.84419	0.89927	0.85112	0.87204	0.89820	0.01869	0.78439	0.80532	0.81899
Two-phase Binary Relevance LightGBM	0.84782	0.90075	0.85049	0.87216	0.89768	0.01860	0.78468	0.80426	0.82138

Table 7
F measure by CAPEC Classification.

Model	F meas. Avg.	000	272	242	88	126	66	16	310	153	274	194	34	33
Two-phase MultiOutput CatBoost	**0.889**	**0.928**	0.618	**0.862**	**0.884**	**0.855**	**0.855**	**0.999**	0.997	**0.903**	0.999	0.855	**0.509**	**0.905**
Two-phase MultiOutput LightGBM	0.886	0.927	0.600	0.858	0.858	0.852	0.850	**0.999**	**0.999**	0.857	**1.000**	0.849	0.508	0.897
Customized model CatBoost	0.885	0.925	**0.621**	0.860	0.877	0.852	0.853	**0.999**	0.997	0.895	0.999	**0.856**	0.431	0.897
Single-phase Binary Relevance CatBoost	0.877	0.928	0.559	0.790	0.718	0.804	0.843	0.997	0.997	0.842	0.999	0.841	0.359	0.718
Two-phase Binary Relevance CatBoost	0.877	0.928	0.561	0.788	0.716	0.803	0.843	0.997	0.996	0.849	0.998	0.841	0.354	0.718
Single-phase Clas.Chain LightGBM	0.872	0.927	0.560	0.760	0.685	0.790	0.838	0.997	0.996	0.636	**1.000**	0.822	0.362	0.723
Two-phase Binary Relevance LightGBM	0.872	0.927	0.548	0.761	0.677	0.797	0.837	0.998	0.999	0.807	**1.000**	0.827	0.325	0.726
Single-phase Binary Relevance LightGBM	0.872	0.926	0.546	0.762	0.680	0.796	0.838	0.998	0.999	0.806	**1.000**	0.828	0.327	0.726
Single-phase Clas.Chain CatBoost	0.872	0.923	0.549	0.792	0.729	0.797	0.835	**0.999**	0.997	0.834	**1,000**	0.833	0.364	0.724
Customized model LightGBM	0.859	0.907	0.553	0.759	0.671	0.797	0.840	0.996	0.012	0.802	**1.000**	0.825	0.331	0.732

000: Normal 272: Protocol Manipulation 242: Code Injection 88: OS Command Injection 126: Path Traversal 66: SQLi 16: Dictionary based Password Attack 310: Scanning for Vulnerable Software 153: Input Data Manipulation 274: HTTP Verb Tampering 194: Fake the Source of Data 34: HTTP Response Splitting 33 HTTP Request Smuggling.

Table 8
Accuracy by CAPEC Classification.

Model	Acc. global	000	272	242	88	126	66	16	310	153	274	194	34	33
Two-phase MultiOutput CatBoost	**0.884**	**0.919**	**0.994**	**0.995**	**0.998**	**0.994**	**0.914**	**1.000**	**1.000**	**1.000**	**1.000**	**0.982**	**0.983**	**1.000**
Customized model CatBoost	**0.884**	0.914	**0.994**	**0.995**	**0.998**	0.993	0.913	**1.000**	**1.000**	**1.000**	**1.000**	**0.982**	0.982	**1.000**
Two-phase MultiOutput LightGBM	0.881	0.917	0.993	0.995	**0.998**	0.993	0.911	**1.000**	**1.000**	0.999	**1.000**	**0.982**	**0.983**	**1.000**
Single-phase Clas.Chain LightGBM	0.872	0.917	0.993	0.992	0.995	0.990	0.903	**1.000**	**1.000**	0.998	**1.000**	0.978	0.979	0.999
Single-phase Clas.Chain CatBoost	0.872	0.913	0.913	0.993	0.996	0.991	0.900	**1.000**	**1.000**	0.999	**1.000**	0.980	0.980	0.999
Customized model LightGBM	0.859	0.891	0.993	0.992	0.995	0.991	0.911	**1.000**	0.990	0.999	**1.000**	0.979	0.981	0.999
Two-phase Binary Relevance CatBoost	0.852	**0.919**	0.993	0.993	0.996	0.992	0.913	**1.000**	**1.000**	0.999	**1.000**	0.981	0.981	0.999
Single-phase Binary Relevance CatBoost	0.849	**0.919**	0.993	0.993	0.996	0.992	0.913	**1.000**	**1.000**	0.999	**1.000**	0.981	0.981	0.999
Two-phase Binary Relevance LightGBM	0.848	0.917	0.993	0.992	0.995	0.991	0.910	**1.000**	**1.000**	0.999	**1.000**	0.980	0.981	0.999
Single-phase Binary Relevance LightGBM	0.844	0.917	0.993	0.992	0.995	0.991	0.910	**1.000**	**1.000**	0.999	**1.000**	0.980	0.980	0.999

000: Normal 272: Protocol Manipulation 242: Code Injection 88: OS Command Injection 126: Path Traversal 66: SQLi 16: Dictionary based Password Attack 310: Scanning for Vulnerable Software 153: Input Data Manipulation 274: HTTP Verb Tampering 194: Fake the Source of Data 34: HTTP Response Splitting 33 HTTP Request Smuggling.

Table 9
Best scores obtained for LightGBM and CatBoost Algorithms.

Metric	LightGBM	CatBoost
Accuracy EMR	0.88095	**0.88445**
Precision	0.90075	**0.90515**
Recall	0.88641	**0.88829**
F measure	0.88615	**0.88912**
ROC AUC	0.91139	**0.91322**
Hamming Loss	0.01754	**0.01703**
Jaccard Similarity	0.80214	**0.80672**
Informedness	0.83200	**0.83580**
Markedness	0.82138	**0.82741**

dataset, created specifically to allow this type of multi-label classification. 70% of web requests present in the dataset have been used to train the different algorithms, while the remaining 30% of web requests have been used to validate the models generated.

Table 6 presents a summary of the different metrics obtained for each algorithm/model combination, ordered by recall score.

Table 7 details the F measure obtained for each algorithm/model combination in the different CAPEC classifications.

Table 8 details the Accuracy obtained for each algorithm/model combination in the different CAPEC classifications.

As can be seen in tables 6, 7, 8, the combination of the CatBoost algorithm and a two-phase model in which the MultiOutputClassifier class of the Scikit-learn library is applied is the one that obtains the best results in practically all the metrics. This model obtains a strict Accuracy (EMR) of 0.8844, an average F measure of 0.8891, an AUC ROC of 0.9132, and a Hamming Loss of 0.01703.

The two-phase models with the MultiOutputClassifier class are clearly superior to the other models, regardless of the algorithm (LightGBM or CatBoost) included in them: using LightGBM, their EMR is 0.88095, their average F measure is 0.8862, their ROC AUC is 0.9114 and their Hamming Loss is 0.01754.

See Figs. 8, 9 for a detail of the metrics obtained by each algorithm/model combination.

Although all algorithm/model combinations obtain good F-Score and Accuracy scores in each CAPEC classification, it becomes evident that the combination of the CatBoost algorithm and the two-phase model with the MultiOutpoutClassifier class yields the best results, obtaining the highest Accuracy scores in all CAPEC classifications and the highest F-Score in 9 of the 13 CAPEC classifications.

The superiority of the CatBoost algorithm over LightGBM can also be clearly seen, regardless of the model with which they are combined. See Table 9 for a comparison of the best scores obtained by LightGBM and CatBoost in the various metrics evaluated. Note that in the case of the Hamming Loss metric, a lower score is indicative of better performance, as indicated in Section 2.1.3.1.

Following the work of Antunes and Vieira (2015), which recommends the most appropriate metrics according to the type of scenario, the best algorithm/model combination has been selected based on the recommended metric for each different scenario as detailed in 2.1.3.2. Table 10 shows the recommended metric for each of the four scenarios, the top three values obtained in this metric, as well as the algorithm/model combinations that obtained this value.

From the results in Table 10, it is concluded that the combination of the CatBoost algorithm and the two-phase model with the *MultiOutpoutClassifier* class, is the one that obtains the best performance with the recommended metrics in the three most critical scenarios analyzed (*Very high critical scenario, Heightened-critical scenario*, and *Medium-critical scenario*). Only in the *Low-critical scenario*, this algorithm/model combination is outperformed by the combination of CatBoost and the two-phase model with the *Bina-*

Table 10
Recommended metrics for scenario and top three algorithm/model combination.

Scenario	Metric	Best values	Algorithm / Model Combination
Business-critical	Recall	0.88829	Two-phase MultiOutput CatBoost
		0.88790	Customized model CatBoost
		0.88641	Two-phase MultiOutput LightGBM
Heightened-critical applications	Informedness	0.83580	Two-phase MultiOutput CatBoost
		0.83200	Two-phase MultiOutput LightGBM
		0.82690	Customized model CatBoost
Best effort	F-Measure	0.88912	Two-phase MultiOutput CatBoost
		0.88615	Two-phase MultiOutput LightGBM
		0.88501	Customized model CatBoost
Minimum effort	Markedness	0.82741	Two-phase Binary Relevance CatBoost
		0.82563	Two-phase MultiOutput CatBoost
		0.82138	Two-phase Binary Relevance LightGBM

ryRelevance class, obtaining the second-best score. From the review of the scores obtained in the different metrics, it can be concluded that the Two-phase MultiOutput CatBoost model is adequate for all the scenarios considered.

6. Conclusions

In this work we have presented the SR-BH 2020 multi-label dataset, which includes a set of 13 different labels, providing information about the normality of each web request and its possible classification into 12 different CAPEC categories.

A new way to give a numerical value to the alphanumeric strings and symbols that constitute the different fields that conform a web request, by calculating the average of the sum of the ASCII values of each of the characters in each field, is proposed; this numerical value, calculated easily and quickly, allows the extraction of features and the training of the different machine learning models.

We have also designed and evaluated different multi-label classification models, using modules and classes from the scikit-learn and scikit-multilearn libraries. Two leading algorithms in the field of machine learning have been tested with these models: LightGBM and CatBoost. The results obtained by our experiments show a clear superiority of the combination of the CatBoost algorithm and the two-phase model with the MultiOutputClassifier module of the scikit-learn library, in multi-label classification tasks. In future work, the possibility of executing automatic remediation actions, based on CAPEC attack patterns, could be considered.

Consideration should also be given to the possibility of generating a new dataset with data from traffic originating from communications between SOAP or REST web services or even data from web API communication. The labeling of the attacks performed on these services and their CAPEC classification would allow the application of the combinations of algorithms and models generated in this work to new datasets as well as to determine the possible generalization of results to other contexts.

In addition, due to the excellent results obtained in all the scenarios analyzed by the Two-phase MultiOutput CatBoost model, more effort and research could be devoted to the development and improvement of this model to achieve its integration into commercial or open-source web application protection tools that can be used in all types of scenarios with different levels of criticality.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Tomás Sureda Riera: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Juan-Ramón Bermejo Higuera:** Conceptualization, Methodology, Validation, Writing – review & editing, Visualization, Supervision. **Javier Bermejo Higuera:** Conceptualization, Methodology, Validation, Writing – review & editing, Visualization, Supervision. **José-Javier Martínez Herraiz:** Validation, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Juan-Antonio Sicilia Montalvo:** Conceptualization, Methodology, Validation, Writing – review & editing, Visualization, Supervision.

References

Antunes, N., Vieira, M., 2015. On the metrics for benchmarking vulnerability detection tools. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 505–516. doi:10.1109/DSN.2015.30.

Auxilia, M., Tamilselvan, D., 2010. Anomaly detection using negative security model in Web application. In: 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 481–486. doi:10.1109/CISIM.2010.5643461.

Bermejo Higuera, J.R., 2013. Metodología de evaluación de herramientas de análisis automático de seguridad de aplicaciones web para su adaptación en el ciclo de vida de desarrollo. Universidad Nacional Educación a Distancia (UNED). http://e-spacio.uned.es/fez/eserv/tesisuned:IngInd-Jrbermejo/BERMEJO_HIGUERA_Juan_Ramon_Tesis.pdf

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and regression trees.(the wadsworth statistics/probability series), belmont. CA: Wadsworth.

Brugger T.. KDD Cup '99 dataset (Network Intrusion) considered harmful (KDnuggets News 07:18, item 4, Features). 2007. https://www.kdnuggets.com/news/2007/n18/4i.html.

Büyükçakir, A., Bonab, H., Can, F., 2018. A novel online stacked ensemble for multi-label stream classification. In: International Conference on Information and Knowledge Management, Proceedings. Association for Computing Machinery, pp. 1063–1072. doi:10.1145/3269206.3271774.

Charte, F., Rivera, A.J., Del Jesus, M.J., Herrera, F., 2015. MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. Knowl Based Syst 89, 385–397. doi:10.1016/j.knosys.2015.07.019.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., et al., 2002. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16, 321–357. doi:10.1613/jair.953.

Cheng, W., Hüllermeier, E., 2009. Combining instance-based learning and logistic regression for multilabel classification. Mach Learn 76 (2), 211–225. doi:10.1007/s10994-009-5127-5.

Cisco. Cisco Annual Internet Report (2018–2023). 2018. https://bit.ly/3a4a1H4.

Dang, Q.V., François, J., 2018. Utilizing attack enumerations to study SDN/NFV vulnerabilities. In: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 356–361. doi:10.1109/NETSOFT.2018.8459961.

Devi, R., Abualkibash, M., 2019. Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets - a review paper. International Journal of Computer Science and Information Technology 11, 65–80. doi:10.5121/ijcsit.2019.11306.

Díaz, G., Bermejo, J.R., 2013. Static analysis of source code security: assessment of tools against SAMATE tests. Inf Softw Technol 55 (8), 1462–1476. doi:10.1016/j.infsof.2013.02.005.

Dorogush, A.V., Ershov, V., Gulin, A., 2018. Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:181011363.

Dubey, S.K., 2016. An evaluation of java applications using security requirements. International Journal of Recent Trends in Engineering & Research Issue 02 (08). 2455–1457

Gartner. Runtime Application Self-Protection (RASP) - Gartner IT Glossary. 2022. http://www.gartner.com/it-glossary/runtime-application-self-protection-rasp.

Gouk, H., Pfahringer, B., Cree, M., 2016. Learning distance metrics for multi-label classification. In: Durrant, R.J., Kim, K.E. (Eds.), Proceedings of The 8th Asian Conference on Machine Learning. In: Proceedings of Machine Learning Research, volume 63. PMLR, The University of Waikato, Hamilton, New Zealand, pp. 318–333. https://proceedings.mlr.press/v63/Gouk8.html

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., et al., 2002. Gene selection for cancer classification using support vector machines. Mach Learn 46 (1–3), 389–422. doi:10.1023/A:1012487302797.

Haldar, V., Chandra, D., Franz, M., 2005. Dynamic taint propagation for Java. In: 21st Annual Computer Security Applications Conference (ACSAC'05), volume 2005. IEEE, Tucson, Arizona, pp. 303–311. doi:10.1109/CSAC.2005.21.

Halfond, W.G.J., Orso, A., Manolios, P., 2008. WASP: Protecting web applications using positive tainting and syntax-Aware evaluation. IEEE Trans. Software Eng. 34 (1), 65–81. doi:10.1109/TSE.2007.70748.

Hancock, J.T., Khoshgoftaar, T.M., 2020. Catboost for big data: an interdisciplinary review. J Big Data 7 (1), 94. doi:10.1186/s40537-020-00369-8.

Hiscox, 2021. Don't let cyber be a game of chance. Hiscox Cyber Readiness Report 2021. Technical Report. Hiscox. https://bit.ly/3AbiJ0t

Jin, X., Cui, B., Yang, J., Cheng, Z., 2018. Payload-based Web attack detection using deep neural network. In: Lecture Notes on Data Engineering and Communications Technologies, volume 12. Springer, Cham, pp. 482–488. doi:10.1007/978-3-319-69811-3_44.

Johari, R., Sharma, P., 2012. A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection. In: 2012 International Conference on Communication Systems and Network Technologies, pp. 453–458. doi:10.1109/CSNT.2012.104.

Kanakogi, K., Washizaki, H., Fukazawa, Y., Ogata, S., Okubo, T., Kato, T., Kanuka, H., Hazeyama, A., Yoshioka, N., 2021. Tracing CVE vulnerability information to CAPEC attack patterns using natural language processing techniques. Information 12 (8). doi:10.3390/info12080298.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y., 2017. LightGBM: a highly efficient gradient boosting decision tree. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp. 3149–3157.

Kozik, R., Choraś, M., Rafał, R., Witold, H., 2015. Patterns Extraction Method for Anomaly Detection in HTTP Traffic. In: Herrero, A., Baruque, B., Sedano, J., Quintián, H., Corchado, E. (Eds.), International Joint Conference. Springer International Publishing, Cham, pp. 227–236.

Kruegel, C., Vigna, G., 2003. Anomaly detection of Web-based attacks. In: Proceedings of the ACM Conference on Computer and Communication Security (CCS). ACM Press, Washington, DC, pp. 251–261.

Krügel, C., Toth, T., Kirda, E., 2002. Service specific anomaly detection for network intrusion detection. In: Proceedings of the 2002 ACM Symposium on Applied Computing. Association for Computing Machinery, New York, NY, USA, pp. 201–208. doi:10.1145/508791.508835.

Liang, J., Zhao, W., Ye, W., 2017. Anomaly-based Web attack detection: a deep learning approach. In: Proceedings of the 2017 VI International Conference on Network, Communication and Computing. Association for Computing Machinery, New York, NY, USA, pp. 80–85. doi:10.1145/3171592.3171594.

Lichman M.. 1999 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory. 2000. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset.

Mac, H., Truong, D., Nguyen, L., Nguyen, H., Tran, H.A., Tran, D., 2018. Detecting attacks on Web applications using autoencoder. In: Proceedings of the Ninth International Symposium on Information and Communication Technology. Association for Computing Machinery, New York, NY, USA, pp. 416–421. doi:10.1145/3287921.3287946.

Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S., 2012. An extensive experimental comparison of methods for multi-label learning. Pattern Recognit 45 (9), 3084–3104. doi:10.1016/J.PATCOG.2012.03.004.

Mahoney, M.V., Chan, P.K., 2003. An analysis of the 1999 DARPA/lincoln laboratory evaluation data for network anomaly detection. In: Vigna, G., Kruegel, C., Jonsson, E. (Eds.), Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 220–237.

McHugh, J., 2000. Testing intrusion detection systems: A Critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans Inf Syst Secur 3 (4), 262–294. doi:10.1145/382912.382923.

Montes, N., Betarte, G., Martínez, R., Pardo, A., 2021. Web Application Attacks Detection Using Deep Learning. In: Tavares, J.M.R.S., Papa, J.P., González Hidalgo, M. (Eds.), Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer International Publishing, Cham, pp. 227–236.

Moosa, A., 2010. Artificial neural network based web application firewall for SQL injection. International Journal of Information, Control and Computer Sciences 3.0 (4). doi:10.5281/zenodo.1329474.

Moustafa, N., Slay, J., 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6. doi:10.1109/MilCIS.2015.7348942.

Oliveira, N., Praça, I., Maia, E., Sousa, O., 2021. Intelligent cyber attack detection and classification for network-Based intrusion detection systems. Applied Sciences 11 (4). doi:10.3390/app11041674.

OWASP. SQL Injection Bypassing WAF - OWASP. https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF.

Pan, Y., Sun, F., Teng, Z., White, J., Schmidt, D.C., Staples, J., Krause, L., 2019. Detecting web attacks with end-to-end deep learning. Journal of Internet Services and Applications 2019 10:1 10 (1), 1–22. doi:10.1186/S13174-019-0115-X.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A., 2018. CatBoost: Unbiased Boosting with Categorical Features. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp. 6639–6649.

Protić, D., 2018. Review of KDD cup '99, NSL-KDD and Kyoto 2006+ datasets. Vojnotehnicki glasnik 66 (3), 580–596. doi:10.5937/vojtehg66-16670.

Raïssi, C., Brissaud, J., Dray, G., Poncelet, P., Roche, M., Teisseire, M., 2007. Web Analyzing Traffic Challenge: Description and Results. In: ECML/PKDD'2007 Discovery Challenge, France, p. 6. https://hal-lirmm.ccsd.cnrs.fr/lirmm-00168955

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2009. Classifier chains for multi-label classification. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, pp. 254–269.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2011. Classifier chains for multi-label classification. Mach Learn 85 (3), 333–359. doi:10.1007/s10994-011-5256-5.

Resende, P.A.A., Drummond, A.C., 2018. Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. Security and Privacy 1 (4), e36. doi:10.1002/spy2.36.

Ristic I.. Protocol-Level Evasion of Web Application Firewalls – Network Security Blog | Qualys, Inc. 2022. https://blog.qualys.com/ssllabs/2012/07/25/protocol-level-evasion-of-web-application-firewalls.

Ross Quinlan, b.J., Kaufmann Publishers, M., Salzberg, S.L., 1994. C4.5: Programs for machine learning by J. Ross Quinlan. Morgan Kaufmann Publishers, inc., 1993. Machine Learning 1994 16:3 16 (3), 235–240. doi:10.1007/BF00993309.

Schapire, R.E., Singer, Y., 2000. Boostexter: A Boosting-based system for text categorization. Mach Learn 39 (2), 135–168. doi:10.1023/A:1007649029923.

Shi, H., 2007. Best-first Decision Tree Learning. The University of Waikato, Hamilton, New Zealand. https://hdl.handle.net/10289/2317

Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security 31 (3), 357–374. doi:10.1016/j.cose.2011.12.012.

Siddique, K., Akhtar, Z., Khan, F.A., Kim, Y., 2019. KDD Cup 99 data sets: a perspective on the role of data sets in network intrusion detection research. Computer (Long Beach Calif) 52 (2), 41–51. doi:10.1109/MC.2018.2888764.

Steiner, S., de Leon, D.C., Alves-Foss, J., 2017. A structured analysis of SQL injection runtime mitigation techniques. In: Proceedings of the Annual Hawaii International Conference on System Sciences, volume 2017-January, pp. 2887–2895. doi:10.24251/hicss.2017.349.

Sureda Riera, T., Bermejo Higuera, J.R., Bermejo Higuera, J., Martínez Herraiz, J.J., Sicilia Montalvo, J.A., 2020. Prevention and fighting against web attacks through anomaly detection technology. a systematic review. Sustainability 12 (12). doi:10.3390/su12124945.

Swets, J.A., 1996. Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US.

Szymański, P., Kajdanowicz, T., 2017. A scikit-based python environment for performing multi-label classification. ArXiv e-prints.

Tama, B.A., Nkenyereye, L., Islam, S.M.R., Kwak, K.S., 2020. An enhanced anomaly detection in web traffic using a stack of classifier ensemble. IEEE Access 8, 24120–24134. doi:10.1109/ACCESS.2020.2969428.

Tan, H.H., Hoai, T.V., 2021. Web Application Anomaly Detection Based On Converting HTTP Request Parameters To Numeric. In: 2021 15th International Conference on Advanced Computing and Applications (ACOMP), pp. 93–97. doi:10.1109/ACOMP53746.2021.00019.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A Detailed Analysis of the KDD CUP 99 Data Set. In: Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications. IEEE Press, pp. 53–58.

Tekerek, A., 2021. A novel architecture for web-based attack detection using convolutional neural network. Computers and Security 100, 102096. doi:10.1016/j.cose.2020.102096.

Torrano-Gimenez, C., Perez-Villegas, A., Alvarez, G., 2009. A Self-learning Anomaly-Based Web Application Firewall. In: Herrero, A., Gastaldo, P., Zunino, R., Corchado, E. (Eds.), Computational Intelligence in Security for Information Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 85–92.

Truong, D., Tran, D., Nguyen, L., Mac, H., Tran, H.A., Bui, T., 2019. Detecting Web attacks using stacked denoising autoencoder and ensemble learning methods. In: Proceedings of the Tenth International Symposium on Information and Communication Technology. Association for Computing Machinery, New York, NY, USA, pp. 267–272. doi:10.1145/3368926.3369715.

Tsoumakas, G., Katakis, I., 2009. Multi-Label classification: an overview. Int. J. Data Warehouse. Min. 3, 1–13. doi:10.4018/jdwm.2007070101.

Van Rijsbergen, C.J., 1979. Information Retrieval, 2nd Butterworth-Heinemann Newton, MA, USA.

Vu, Q.H., Ruta, D., Cen, L., 2019. Gradient boosting decision trees for cyber security threats detection based on network events logs. In: Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019. Institute of Electrical

and Electronics Engineers Inc., pp. 5921–5928. doi:10.1109/BigData47090.2019.9006061.
Wang, H., Li, Z., Huang, J., Hui, P., Liu, W., Hu, T., Chen, G., 2020. Collaboration based multi-label propagation for fraud detection. In: IJCAI International Joint Conference on Artificial Intelligence, volume 2021-Janua. International Joint Conferences on Artificial Intelligence, pp. 2477–2483. doi:10.24963/ijcai.2020/343.
Zhang, M.L., Zhou, Z.H., 2007. ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognit 40 (7), 2038–2048. doi:10.1016/j.patcog.2006.12.019.
Zhang, M.L., Zhou, Z.H., 2014. A review on multi-label learning algorithms. IEEE Trans Knowl Data Eng 26 (8), 1819–1837. doi:10.1109/TKDE.2013.39.

Tomás Sureda Riera holds a degree in psychology from Universidad Nacional de Educación a Distancia (UNED) and a master's degree in informatics security from the Universidad Internacional de la Rioja (UNIR), in 2002 and 2017, respectively. His master research thesis was qualified with honors and won an ISACA (Valencia Chapter) award for best master research thesis in 2017. His research interests are: security operations online, web and databases security, penetration testing and data science applied to cyber security.

Juan Ramón Bermejo Higuera received the M.Sc. degree in computer engineering and the Ph.D. degree from the Distance Education National University, Spain, in 1998 and 2014, respectively. He is currently the Chief of the Cybersecurity Unit, National Institute of Aerospace Techniques, Spain. He is also a Professor of applications security in the International University of La Rioja and also an Associate Professor of the Ciberdefence Master Science degree with the Alcala de Henares University. He has authored several publications. His research interests include cybersecurity and cyber defense.

Javier Bermejo Higuera received the B.S. degree from Alcala University and the Ph.D. degree from the Army Polytechnic School. He is currently a Professor with the Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja. He has authored several publications. His research interests include the fields of software security, cybersecurity, and malware analysis.

Jose Javier Martínez-Herráiz has been an Associate Professor with the Department of Computer Science (Artificial Intelligence Area), University of Alcalá, Alcalá de Henares, Spain (since 1994), where he is the Rector's Delegate for Electronic Administration and Security. He was with private telecommunication business companies (Spain and Italy) as a Software Analyst, Project Manager, and Consultant between 1988 and 1999. He was the Director of the Computer Science Department between 2008 and 2011. He has collaborated extensively with Spanish law enforcement agencies and cybersecurity companies, and his work was awarded with Order of Merit of the Police Forces (2011 and 2015). He has practical experience in software development, technology and modeling, methodologies for software projects, planning and management, software maintenance, e-learning, gamification technology, and cybersecurity. He received the degree in computer science from the Polytechnic University of Madrid, Madrid, Spain, and the Ph.D. degree from the University of Alcalá in 2004.

Juan Antonio Sicilia Montalvo received the B.S. and Ph.D. degrees from the Universidad de Zaragoza. He is currently a Professor with the Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja. His research interests include combinatorial optimization, computer security, software development based on mathematical algorithms, numerical methods, and heuristic techniques for solving engineering problems.

5. Conclusiones y trabajos futuros

En el artículo 1 *"Prevention and Fighting against Web Attacks through Anomaly Detection Technology. A Systematic Review"* se detectó que una de las principales carencias que impedían la validación y replicación de los resultados experimentales obtenidos en los diferentes estudios realizados en la detección de ataques mediante técnicas de detección de anomalías era la falta de datasets públicamente accesibles y correctamente etiquetados. Únicamente un 29,55% de los resultados experimentales obtenidos en los estudios revisados en esta revisión sistemática de literatura se basaban en datos públicamente accesibles; de este 29,55%, aproximadamente la mitad de ellos se basaban en datasets fuertemente criticados por la comunidad científica. Al mismo tiempo, quedó patente que se obtuvieron altos valores en las métricas de los modelos de detección de anomalías que incorporaban técnicas de aprendizaje automático, si bien faltaba un criterio estándar en la elección de las métricas; además, los valores de las métricas obtenidas disminuían conforme en los estudios se usaban datasets más recientes, por lo que resultaba claro que eran necesario más esfuerzos en la generación de datasets públicos y actualizados.

En el artículo 2 *"Systematic Approach for Web Protection Runtime Tools' Effectiveness Analysis"* se analizan dos tecnologías de protección de aplicaciones web: WAF y RASP, las cuales permiten proteger las partes vulnerables de la aplicación web, hasta la mitigación de las vulnerabilidades mediante la corrección

de código por parte del equipo de desarrolladores. Al mismo tiempo, se desarrolla una nueva metodología experimental que permite la comparación de cualquier herramienta de protección web, mediante el uso de bancos de prueba y la elección de diferentes indicadores (TP,TN,FP,FN) y métricas (F-Score, Precision, Recall, Accuracy). Una de las conclusiones de este trabajo es la necesidad de complementar las herramientas de protección WAF mediante técnicas de aprendizaje automático.

En el artículo 3 *"A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques"* se genera el dataset multietiqueta SR-BH 2020, que permite el entrenamiento de modelos de aprendizaje automático multietiqueta destinados a la detección y predicción de ataques y la asignación de su correspondiente clasificación CAPEC. Se diseña, además, una nueva forma de codificación de los campos que componen las peticiones web y la extracción de características de un dataset, mediante la suma de la media de la asignación de los correspondientes valores ASCII de una petición web. Se han diseñado modelos de clasificación multietiqueta de una fase, dos fases y customizados, que incluyen los algoritmos LightGBM y CatBoost, evaluándolos en función de métricas de evaluación que tienen en cuenta los diferentes niveles de criticidad de los escenarios en los que se instala la aplicación o el servicio web a proteger. El modelo de dos fases fue claramente superior al resto de modelos y el algoritmo CatBoost fue superior a LightGBM en todos los casos.

En futuros trabajos, podría considerarse la posibilidad de ejecutar acciones de remediación automáticas, basadas en los patrones de ataque de CAPEC.

También cabría considerar la posibilidad de generar un nuevo dataset con datos procedentes del tráfico originado por las comunicaciones entre servicios web SOAP o REST, o incluso datos procedentes de la comunicación entre APIs web. El etiquetado de los ataques realizados a estos servicios y su clasificación CAPEC permitiría aplicar las combinaciones de algoritmos y modelos generados en este trabajo a nuevos conjuntos de datos, así como determinar la posible generalización de los resultados a otros contextos.

Además, debido a los excelentes resultados obtenidos en todos los escenarios analizados por el modelo CatBoost Multisalida de dos fases, se podrían dedicar más esfuerzos e investigaciones al desarrollo y mejora de este modelo para conseguir su integración en herramientas de protección de aplicaciones web comerciales o de código abierto que puedan ser utilizadas en todo tipo de escenarios con diferentes niveles de criticidad.

6. Bibliografía

Ahmad, I., Basheri, M., Iqbal, M. J., & Rahim, A. (2018). Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection. *IEEE Access*, *6*, 33789-33795. https://doi.org/10.1109/ACCESS.2018.2841987

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, *46*(3), 175-185. https://doi.org/10.1080/00031305.1992.10475879

Antunes, N., & Vieira, M. (2015). On the Metrics for Benchmarking Vulnerability Detection Tools. *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 505-516. https://doi.org/10.1109/DSN.2015.30

Balduzzi, M., Gimenez, C. T., Balzarotti, D., & Kirda, E. (2011). Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications. *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. https://www.ndss-symposium.org/ndss2011/automated-discovery-of-parameter-pollution-vulnerabilities-in-web-applications

Barnum, S. (2007). *An Introduction to Attack Patterns as a SwA Knowledge Resource*. OMG's First Software Assurance Workshop: Working Together For Confidence, Fairfax, VA USA. https://www.omg.org/news/meetings/workshops/SWA_2007_Presentations/00-T4_Barnum.pdf

Bisht, P., Hinrichs, T., Skrupsky, N., & Venkatakrishnan, V. N. (2011). WAPTEC: Whitebox analysis of web applications for parameter tampering exploit construction. *Proceedings*

of the 18th ACM conference on Computer and communications security, 575-586. https://doi.org/10.1145/2046707.2046774

Bravenboer, M., Dolstra, E., & Visser, E. (2010). Preventing injection attacks with syntax embeddings. *Science of Computer Programming*, *75*(7), 473-495. https://doi.org/10.1016/j.scico.2009.05.004

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 93-104. https://doi.org/10.1145/342009.335388

Buehrer, G., Weide, B. W., & Sivilotti, P. A. G. (2005). Using parse tree validation to prevent SQL injection attacks. *Proceedings of the 5th international workshop on Software engineering and middleware*, 106-113. https://doi.org/10.1145/1108473.1108496

Coifman, R. R., & Lafon, S. (2006). Diffusion maps. *Applied and Computational Harmonic Analysis*, *21*(1), 5-30. https://doi.org/10.1016/j.acha.2006.04.006

de Carvalho, A. C. P. L. F., & Freitas, A. A. (2009). A Tutorial on Multi-label Classification Techniques. En A. Abraham, A.-E. Hassanien, & V. Snášel (Eds.), *Foundations of Computational Intelligence Volume 5: Function Approximation and Classification* (pp. 177-195). Springer. https://doi.org/10.1007/978-3-642-01536-6_8

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1-38.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis / Richard O. Duda, Peter E. Hart* (P. E. Hart, Ed.). Wiley.

Eltanbouly, S., Bashendy, M., AlNaimi, N., Chkirbene, Z., & Erbad, A. (2020). Machine Learning Techniques for Network Anomaly Detection: A Survey. *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 156-162. https://doi.org/10.1109/ICIoT48696.2020.9089465

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters a Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 226-231.

Fernandes, G., Rodrigues, J. J. P. C., Carvalho, L. F., Al-Muhtadi, J. F., & Proença, M. L. (2019). A comprehensive survey on network anomaly detection. *Telecommunication Systems*, *70*(3), 447-489. https://doi.org/10.1007/s11235-018-0475-8

Franzese, M., & Iuliano, A. (2019). *Hidden Markov Models* (S. Ranganathan, M. Gribskov, K. Nakai, & C. B. T.-E. of B. and C. B. Schönbach, Eds.; pp. 753-762). Academic Press. https://doi.org/10.1016/B978-0-12-809633-8.20488-3

Frey, B. J., & Dueck, D. (2007). Clustering by Passing Messages Between Data Points. *Science*, *315*(5814), 972-976. https://doi.org/10.1126/science.1136800

Gamma, E., Helm, Richard, Johnson, Ralph, & Vlissides, John. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc.

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security*, *28*(1-2), 18-28. https://doi.org/10.1016/j.cose.2008.08.003

Gouk, H., Pfahringer, B., & Cree, M. (2016). Learning Distance Metrics for Multi-Label Classification. En R. J. Durrant & K.-E. Kim (Eds.), *Proceedings of The 8th Asian Conference on Machine Learning* (Vol. 63, pp. 318-333). PMLR. https://proceedings.mlr.press/v63/Gouk8.html

Gupta, M. R., & Chen, Y. (2011). Theory and Use of the EM Algorithm. *Found. Trends Signal Process.*, *4*(3), 223-296. https://doi.org/10.1561/2000000034

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*(1-3), 389-422. https://doi.org/10.1023/A:1012487302797

Heiderich, M., Frosch, T., Jensen, M., & Holz, T. (2011). Crouching tiger - hidden payload: Security risks of scalable vectors graphics. *Proceedings of the 18th ACM conference on Computer and communications security*, 239-250. https://doi.org/10.1145/2046707.2046735

Hodge, V. J., & Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, *22*(2), 85-126. https://doi.org/10.1007/s10462-004-4304-y

Hoglund, G., & McGraw, G. (2004). *Exploiting Software: How to Break Code*. Addison-Wesley Professional.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554-2558. https://doi.org/10.1073/pnas.79.8.2554

Jolliffe, I. T. (2002). Principal Component Analysis. En *Principal Component Analysis*. Springer-Verlag. https://doi.org/10.1007/b98835

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3149-3157.

Kim, A., Park, M., & Lee, D. H. (2020). AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. *IEEE Access*, *8*, 70245-70261. https://doi.org/10.1109/ACCESS.2020.2986882

Kotu, V., & Deshpande, B. (2019). *Chapter 13—Anomaly Detection* (V. Kotu & B. B. T.-D. S. (Second E. Deshpande, Eds.; pp. 447-465). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-814761-0.00013-7

Kozik, R., Choraś, M., Rafał, R., & Witold, H. (2015). Patterns Extraction Method for Anomaly Detection in HTTP Traffic. En Á. Herrero, B. Baruque, J. Sedano, H. Quintián, & E. Corchado (Eds.), *International Joint Conference* (pp. 227-236). Springer International Publishing.

Kruegel, C., Toth, T., & Kirda, E. (2002). Service Specific Anomaly Detection for Network Intrusion Detection. *Proceedings of the 2002 ACM Symposium on Applied Computing*, 201-208. https://doi.org/10.1145/508791.508835

Kruegel, C., & Vigna, G. (2003). Anomaly detection of Web-based attacks. *Proceedings of the ACM Conference on Computer and Communications Security*, 251-261. https://doi.org/10.1145/948143.948144

Kruegel, C., Vigna, G., & Robertson, W. (2005). A multi-model approach to the detection of web-based attacks. *Computer Networks*, *48*(5), 717-738. https://doi.org/10.1016/j.comnet.2005.01.009

Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16-24. https://doi.org/10.1016/J.JNCA.2012.09.004

Livshits, B., Nori, A. V., Rajamani, S. K., & Banerjee, A. (2009). Merlin: Specification inference for explicit information flow problems. *ACM SIGPLAN Notices*, *44*(6), 75-86. https://doi.org/10.1145/1543135.1542485

Luo, C., Tan, Z., Min, G., Gan, J., Shi, W., & Tian, Z. (2021). A Novel Web Attack Detection System for Internet of Things via Ensemble Classification. *IEEE Transactions on Industrial Informatics*, *17*(8), 5810-5818. https://doi.org/10.1109/TII.2020.3038761

M. Lichman. (2000). *1999 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory*. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset

Manning, C. D., Schütze, H., & Raghavan, P. (Eds.). (2008). Scoring, term weighting, and the vector space model. En *Introduction to Information Retrieval* (pp. 100-123). Cambridge University Press. https://doi.org/DOI: 10.1017/CBO9780511809071.007

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., & Foozy, C. F. M. (2021). Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset. *IEEE Access*, *9*, 22351-22370. https://doi.org/10.1109/ACCESS.2021.3056614

McLachlan, G. J. (2004). *Discriminant analysis and statistical pattern recognition* (Vol. 544). John Wiley & Sons.

MITRE Corporation. (2019). *CVE-2020-1938* [Available from MITRE, CVE-ID CVE-2020-1938]. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-1938

MITRE Corporation. (2020). *CVE-2020-14882* [Available from MITRE, CVE-ID CVE-2020-14882.]. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-14882

MITRE Corporation. (2022a). *CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC)*. https://capec.mitre.org/index.html

MITRE Corporation. (2022b). *CVE - Common Vulnerabilities and Exposures*. https://www.cve.org/

MITRE Corporation. (2022c). *CWE - Common Weakness Enumeration*. CWE - Common Weakness Enumeration. https://cwe.mitre.org/

Moore, A. P., Ellison, R. J., & Linger, R. C. (2001). *Attack Modeling for Information Security and Survivability*. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST. https://apps.dtic.mil/sti/citations/ADA388771

Moustafa, N., Hu, J., & Slay, J. (2019). A holistic review of Network Anomaly Detection Systems: A comprehensive survey. *Journal of Network and Computer Applications, 128*, 33-55. https://doi.org/10.1016/j.jnca.2018.12.006

Nadji, Y., Saxena, P., & Song, D. (2009). Document structure integrity: A robust basis for cross-site scripting defense. *In NDSS*.

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics, 7*. https://www.frontiersin.org/article/10.3389/fnbot.2013.00021

Payne & Meisel. (1977). An Algorithm for Constructing Optimal Binary Decision Trees. *IEEE Transactions on Computers*, *C-26*(9), 905-916. https://doi.org/10.1109/TC.1977.1674938

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,

Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825-2830.

Prokhorenko, V., Choo, K.-K. R., & Ashman, H. (2016). Web application protection techniques: A taxonomy. *Journal of Network and Computer Applications*, *60*, 95-112. https://doi.org/10.1016/j.jnca.2015.11.017

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased Boosting with Categorical Features. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6639-6649.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333-359. https://doi.org/10.1007/s10994-011-5256-5

Rokach, L., & Maimon, O. (2005). *Clustering Methods BT - Data Mining and Knowledge Discovery Handbook* (O. Maimon & L. Rokach, Eds.; pp. 321-352). Springer US. https://doi.org/10.1007/0-387-25465-X_15

Roweis, S. T., & Saul, L. K. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, *290*(5500), 2323 LP - 2326. https://doi.org/10.1126/science.290.5500.2323

Schapire, R. E., & Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, *39*(2), 135-168. https://doi.org/10.1023/A:1007649029923

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., & Platt, J. (1999). Support Vector Method for Novelty Detection. *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 582-588.

Sethi, A., & Barnum, S. (2013, mayo 14). *Introduction to Attack Patterns | CISA*. Introduction to Attack Patterns | CISA. https://www.cisa.gov/uscert/bsi/articles/knowledge/attack-patterns/introduction-to-attack-patterns

Skrupsky, N., Bisht, P., Hinrichs, T., Venkatakrishnan, V. N., & Zuck, L. (2013). TamperProof: A server-agnostic defense for parameter tampering attacks on web applications. *Proceedings of the third ACM conference on Data and application security and privacy*, 129-140. https://doi.org/10.1145/2435349.2435365

Sommer, R. (2008). *Viable Network Intrusion Detection: Trade-Offs in High-Performance Environments*. VDM Verlag.

Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *2010 IEEE Symposium on Security and Privacy*, 305-316. https://doi.org/10.1109/SP.2010.25

Stephen, R. (2004). Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation*, *60*(5), 503-520. https://doi.org/10.1108/00220410410560582

Su, Z., & Wassermann, G. (2006). The essence of command injection attacks in web applications. *ACM SIGPLAN Notices*, *41*(1), 372-382. https://doi.org/10.1145/1111320.1111070

Sun, F., Xu, L., & Su, Z. (2011). *Static Detection of Access Control Vulnerabilities in Web Applications*. 20th USENIX Security Symposium (USENIX Security 11). https://www.usenix.org/conference/usenix-security-11/static-detection-access-control-vulnerabilities-web-applications

Sureda Riera, T., Bermejo Higuera, J. R., Bermejo Higuera, J., Sicilia Montalvo, J. A., & Martínez Herráiz, J. J. (2022). *SR-BH 2020 multi-label dataset* [Data set]. Harvard Dataverse. https://doi.org/10.7910/DVN/OGOIXX

Sureda Riera, T., Bermejo Higuera, J.-R., Bermejo Higuera, J., Martínez Herraiz, J.-J., & Sicilia Montalvo, J.-A. (2020). Prevention and Fighting against Web Attacks through Anomaly Detection Technology. A Systematic Review. *Sustainability*, *12*(12). https://doi.org/10.3390/su12124945

Swets, J. A. (1996). Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers. En *Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers.* Lawrence Erlbaum Associates, Inc.

Szymański, P., & Kajdanowicz, T. (2018). *A scikit-based Python environment for performing multi-label classification* (arXiv:1702.01460). arXiv. https://doi.org/10.48550/arXiv.1702.01460

Tekerek, A. (2021). A novel architecture for web-based attack detection using convolutional neural network. *Computers and Security*, *100*, 102096. https://doi.org/10.1016/j.cose.2020.102096

Tian, Z., Luo, C., Qiu, J., Du, X., & Guizani, M. (2020). A Distributed Deep Learning System for Web Attack Detection on Edge Devices. *IEEE Transactions on Industrial Informatics*, *16*(3), 1963-1971. https://doi.org/10.1109/TII.2019.2938778

Tomović, A., Janičić, P., & Kešelj, V. (2006). N-Gram-based classification and unsupervised hierarchical clustering of genome sequences. *Computer Methods and Programs in Biomedicine*, *81*(2), 137-153. https://doi.org/10.1016/j.cmpb.2005.11.007

Tonella, P., & Ricca, F. (2004). A 2-layer model for the white-box testing of Web applications. *Sixth IEEE International Workshop on Web Site Evolution Proceedings*, 11-19. https://doi.org/10.1109/WSE.2004.10012

Tripp, O., Pistoia, M., Fink, S. J., Sridharan, M., & Weisman, O. (2009). TAJ: Effective taint analysis of web applications. *ACM SIGPLAN Notices*, *44*(6), 87-97. https://doi.org/10.1145/1543135.1542486

Van Gundy, M., & Chen, H. (2012). Noncespaces: Using randomization to defeat cross-site scripting attacks. *Computers & Security*, *31*(4), 612-628. https://doi.org/10.1016/j.cose.2011.12.004

W3Techs. (2022). *Usage Statistics and Market Share of Content Management Systems for Websites*. W3Techs. https://w3techs.com/technologies/overview/content_management

Webb, G. I., Boughton, J. R., & Wang, Z. (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, *58*(1), 5-24. https://doi.org/10.1007/s10994-005-4258-6

Weinberger, J., Saxena, P., Akhawe, D., Finifter, M., Shin, R., & Song, D. (2011). A Systematic Analysis of XSS Sanitization in Web Application Frameworks. En V. Atluri & C. Diaz (Eds.), *Computer Security – ESORICS 2011* (pp. 150-171). Springer. https://doi.org/10.1007/978-3-642-23822-2_9

Yu, F., Alkhalaf, M., & Bultan, T. (2010). Stranger: An Automata-Based String Analysis Tool for PHP. En J. Esparza & R. Majumdar (Eds.), *Tools and Algorithms for the Construction and Analysis of Systems* (pp. 154-157). Springer. https://doi.org/10.1007/978-3-642-12002-2_13

Zhang, M. L., & Zhou, Z. H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819-1837. https://doi.org/10.1109/TKDE.2013.39

Zheng, Y., & Zhang, X. (2013). Path sensitive static analysis of web applications for remote code execution vulnerability detection. *2013 35th International Conference on Software Engineering (ICSE)*, 652-661. https://doi.org/10.1109/ICSE.2013.6606611